UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

EDER JACQUES PORFÍRIO FARIAS

AUTOEVAL-CT: UM ARTEFATO DE AVALIAÇÃO SEMI-AUTOMATIZADA DO PENSAMENTO COMPUTACIONAL

FORTALEZA

2025

EDER JACQUES PORFÍRIO FARIAS

AUTOEVAL-CT: UM ARTEFATO DE AVALIAÇÃO SEMI-AUTOMATIZADA DO PENSAMENTO COMPUTACIONAL

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universi- dade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Computação. Área de Concentração: Sistemas de Informação.

Orientador: Prof. Dr. Windson Viana de Carvalho.
Coorientadora: Prof. Dra. Jacqueline Ramos Macedo Antunes de Souza.

FORTALEZA

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F238a Farias, Eder Jacques Porfírio.
Autoeval-ct : um artefato de avaliação semi-automatizada do pensamento computacional / Eder Jacques Porfírio Farias. – 2025.
225 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2025.
Orientação: Prof. Dr. Windson Viana de Carvalho.
Coorientação: Prof. Dr. Jacqueline Ramos Macedo Antunes de Souza.

1. Pensamento computacional. 2. Educação de computação. 3. Avaliação da aprendizagem. 4. Inteligência artificial. 5. Ensino médio. I. Título.

CDD 005

EDER JACQUES PORFÍRIO FARIAS

AUTOEVAL-CT: UM ARTEFATO DE AVALIAÇÃO SEMI-AUTOMATIZADA DO PENSAMENTO COMPUTACIONAL

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requi- sito parcial à obtenção do título de doutor em Computação. Área de Concentração: Sistemas de Informação.

Aprovada em: 27/08/2025.

BANCA EXAMINADORA

Prof. Dr. Windson Viana de Carvalho (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dra. Jacqueline Ramos Macedo Antunes de Souza (Coorientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Alysson Diniz dos Santos
Universidade Federal do Ceará (UFC)

Prof. Dr. Roberto Pereira
Universidade Federal do Paraná (UFPR)

Prof. Dr. Joaquim Bento Cavalcante Neto
Universidade Federal do Ceará (UFC)

A Deus, pelo dom imerecido da vida; à minha família — minha esposa e meus filhos — por serem o alicerce da minha existência; e aos meus pais, cujo cuidado e dedicação sustentaram os primeiros passos da minha caminhada, expresso minha mais profunda gratidão.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, minha rocha e refúgio, por ser a fonte inesgotável de força, inspiração e sabedoria em todos os momentos desta jornada. Foi nos dias mais difíceis que senti Seu cuidado mais de perto, e nas conquistas, Sua graça me sustentando.

À minha amada esposa, Emanuela Mesquita Porfírio, pelo amor, companheirismo e apoio incondicional, fundamentais para o sucesso desta jornada. Sua presença constante ao meu lado, mesmo nos momentos mais desafiadores, foi fonte de força, equilíbrio e inspiração. Obrigado por acreditar em mim quando até eu duvidei, por compreender minhas ausências, por ouvir meus desabafos e por celebrar comigo cada pequena conquista ao longo deste percurso. Esta vitória também é sua, pois sem você, ela não teria sido possível.

Ao professor Dr. Windson Viana de Carvalho, pela excelente orientação, pela paciência e pela parceria ao longo de toda esta jornada. Suas contribuições foram inestimáveis para o meu desenvolvimento acadêmico e pessoal. Mais do que um orientador, foi um verdadeiro amigo, cuja escuta atenta, conselhos generosos e incentivo constante fizeram toda a diferença. É alguém que levo comigo não apenas como referência intelectual, mas como pessoa que quero manter por perto ao longo da vida.

Aos meus coorientadores, Dra. Jacqueline Ramos Macedo Antunes de Souza e Dr. Alysson Diniz dos Santos, pelas valiosas contribuições no desenvolvimento desta tese.

À Universidade Federal do Ceará (UFC), pelo suporte institucional e pela infraestrutura oferecida ao longo do curso de doutorado.

À Universidade Estadual Vale do Acaraú (UVA), pelo apoio institucional e pela liberação de carga horária, que me permitiu dedicar o tempo necessário à elaboração desta tese.

À Escola de Ensino Estadual de Educação Profissional Monsenhor José Aloysio Pinto, em nome do professor Augusto Braz e do Diretor Flávio Loiola Frota, pela autorização e pelo suporte na aplicação do Experimento Controlado.

Aos colegas da turma do Doutorado Interinstitucional (DINTER), pelas reflexões, críticas construtivas e sugestões recebidas ao longo do percurso.

Aos professores especialistas em pensamento computacional que participaram das entrevistas, pela generosidade em compartilhar seu tempo e conhecimento.

Aos professores da banca examinadora, pela disponibilidade, valiosas colaborações e sugestões pertinentes.

“Porque o Senhor dá a sabedoria; da sua boca procedem o conhecimento e o entendimento.” (Provérbios 2:6)

RESUMO

Este trabalho apresenta uma tese de doutorado cujo objetivo é o desenvolvimento e a análise de uma abordagem avaliativa e de um artefato destinados a auxiliar professores do Ensino Médio na avaliação da aprendizagem do Pensamento Computacional (PC). O ensino e a avaliação do PC tornaram-se aspectos fundamentais na educação contemporânea, especialmente após sua inclusão na Base Nacional Comum Curricular (BNCC). Entretanto, a avaliação dessa competência ainda enfrenta desafios significativos, sobretudo pela escassez de propostas e ferramentas que possibilitem um processo eficiente e sistemático. Para enfrentar esse problema, propõe-se o desenvolvimento e a avaliação do *Automatic Evaluation for Computational Thinking* (AutoEval-CT), um artefato capaz de analisar automaticamente códigos de programação produzidos por estudantes, identificando evidências de PC nas habilidades de abstração, decomposição, reconhecimento de padrões e pensamento algorítmico. O artefato emprega modelos de *Large Language Model* (LLM) para realizar essa análise e gerar relatórios detalhados que apoiam os professores no processo avaliativo. A avaliação da ferramenta foi conduzida segundo os princípios do *Design Science Research* (DSR), em múltiplas etapas, incluindo testes com especialistas, comparações entre avaliações humanas e semi-automatizadas, análises de diferentes modelos de LLM e um experimento controlado com uma turma do Ensino Médio. A hipótese central desta pesquisa é que a adoção de uma abordagem avaliativa hierárquica-quali-quantidativa e de um artefato baseado na análise semi-automatizada de códigos de programação proporciona suporte aos professores no processo de avaliação do PC em alunos do Ensino Médio. Espera-se que os resultados desta pesquisa contribuam no suporte aos professores do Ensino Médio, ao fornecer uma solução inovadora para a avaliação do PC, além de oferecer evidências empíricas para o desenvolvimento de novas pesquisas.

Palavras-chave: pensamento computacional; educação de computação; avaliação da aprendizagem; inteligência artificial; ensino médio.

ABSTRACT

This work presents a doctoral thesis whose objective is the development and analysis of an evaluative approach and an artifact designed to assist high school teachers in assessing students' learning of *Computational Thinking* (CT). The teaching and assessment of CT have become fundamental aspects of contemporary education, especially after its inclusion in the BNCC. However, the assessment of this competence still faces significant challenges, particularly due to the scarcity of proposals and tools that enable an efficient and systematic process. To address this problem, the development and evaluation of AutoEval-CT is proposed, an artifact capable of automatically analyzing programming code produced by students, identifying evidence of CT in the skills of abstraction, decomposition, pattern recognition, and algorithmic thinking. The artifact employs LLM models to carry out this analysis and generate detailed reports that support teachers in the assessment process. The evaluation of the tool was conducted according to the principles of DSR, in multiple stages, including tests with experts, comparisons between human and semi-automated assessments, analyses of different LLM models, and a controlled experiment with a high school class. The central hypothesis of this research is that the adoption of a hierarchical quali-quantitative evaluative approach, together with an artifact based on the semi-automated analysis of programming code, provides support for teachers in assessing CT in high school students. It is expected that the results of this research will contribute to supporting high school teachers by providing an innovative solution for the assessment of CT, as well as offering empirical evidence for the development of future studies.

Keywords: computational thinking; computing education; learning assessment; artificial intelligence; high school.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
API	*Application Programming Interface*
AutoEval-CT	*Automatic Evaluation for Computational Thinking*
BNCC	Base Nacional Comum Curricular
CAAE	Certificado de Apresentação de Apreciação Ética
CT	*Computational Thinking*
DINTER	Doutorado Interinstitucional
DSR	*Design Science Research*
GPU	*Graphics Processing Unit*
IEEE	*Institute of Electrical and Electronics Engineers*
LLM	*Large Language Model*
MSL	Mapeamento Sistemático da Literatura
PC	Pensamento Computacional
PLN	Processamento de Linguagem Natural
SBIE	Simpósio Brasileiro de Informática na Educação
UFC	Universidade Federal do Ceará
UVA	Universidade Estadual Vale do Acaraú

SUMÁRIO

1 INTRODUÇÃO

Esta tese de doutorado tem como objetivo desenvolver e investigar a eficácia de uma abordagem de avaliação e um artefato computacional destinados a apoiar professores do Ensino Médio na avaliação da aprendizagem em Pensamento Computacional (PC). A abordagem avaliativa adotada nesta tese integra uma abordagem hierárquica combinando análises qualitativas (quali) e quantitativas (quanti). O artefato proposto realiza a análise automática de códigos de programação elaborados por estudantes, identificando evidências e indícios relacionados às habilidades de abstração, decomposição, reconhecimento de padrões e pensamento algorítmico. Para isso, são empregados modelos baseados em LLM, os quais possibilitam a geração de relatórios detalhados que subsidiam o processo avaliativo conduzido pelos docentes.

Este capítulo apresenta uma visão introdutória da tese, destacando elementos importantes para a sua compreensão. A Seção 1.1 expõe o contexto que fundamenta o estudo, enquanto a Seção 1.2 descreve as principais motivações que justificam o desenvolvimento desta pesquisa. Em seguida, a Seção 1.3 apresenta a hipótese investigada, a Seção 1.4 delineia as questões de pesquisa que orientam o trabalho, e a Seção 1.5 apresenta a solução proposta de forma geral. Por fim, a Seção 1.6 descreve o roteiro geral da tese, situando o leitor em relação à sua estrutura e desenvolvimento.

1.1 Contextualização

O PC é compreendido como uma abordagem prática para a resolução de problemas, envolvendo a habilidade de projetar sistemas e compreender a relação entre o pensamento humano e os conceitos fundamentais da Ciência da Computação (WING; STANZIONE, 2016; ROMÁN-GONZÁLEZ *et al.*, 2017). Observa-se que a Ciência da Computação e consequentemente o PC estão presentes em praticamente todas as áreas. O uso de seus conceitos, métodos e ferramentas pode transformar o comportamento de diversas disciplinas, profissões e setores (WING; STANZIONE, 2016). Os benefícios educacionais de "pensar computacionalmente" vão além das ciências exatas, contribuindo para o desenvolvimento de habilidades intelectuais que podem ser aplicadas a qualquer domínio.

Embora frequentemente associados, ensino de programação e pensamento computacional não são sinônimos. O ensino de programação concentra-se, sobretudo, em transmitir conhecimentos sobre lógica de programação, técnicas de desenvolvimento, estruturas de dados,

além da sintaxe e semântica específicas de linguagens de programação. Já o PC configura-se como uma competência que pode ser desenvolvida em qualquer pessoa, embora encontre terreno fértil entre profissionais da área de computação, pois está intimamente relacionado à capacidade de resolver problemas de forma estruturada e eficiente. Essa competência envolve, principalmente, habilidades como pensamento algorítmico, abstração, reconhecimento de padrões e decomposição de problemas, sem se restringir, no entanto, ao domínio exclusivo da Ciência da Computação. O ensino de programação é, portanto, uma das diversas estratégias possíveis para promover o PC, mas este transcende a codificação, não se limitando nem se reduzindo unicamente ao universo computacional (WANGENHEIM *et al.*, 2014; SCHOEFFEL *et al.*, 2015; GERALDES *et al.*, 2017).

Nesse contexto, o PC também se apresenta como uma abordagem eficaz para a resolução de problemas em diferentes áreas (FRANÇA; TEDESCO, 2017). Essa perspectiva faz do PC uma alternativa promissora para desenvolver, em profissionais de outras áreas do conhecimento, competências tipicamente associadas aos cientistas da computação. Assim, o ensino do PC para estudantes de diferentes idades emerge como uma estratégia relevante para aprimorar a capacidade de reconhecer, analisar, compreender e solucionar problemas.

Para aprofundar a definição do termo, Barr e Stephenson relataram em (BARR; STEPHENSON, 2011a) a iniciativa de um grupo interdisciplinar composto por 26 pesquisadores, que se reuniu para discutir o conceito de PC. Embora o objetivo não fosse criar um conceito definitivo, buscou-se uma visão compartilhada sobre o significado do PC, especialmente no ensino fundamental e médio. Entre as definições operacionais mais aceitas, destaca-se a subdivisão do PC em quatro habilidades principais: abstração, algoritmos, reconhecimento de padrões e decomposição.

Além de sua definição formal, o PC é reconhecido como um termo acessível e amigável (ROMÁN-GONZÁLEZ *et al.*, 2019), contrastando com conceitos mais técnicos da Ciência da Computação, como “análise de algoritmos”, “lógica de programação” e “estrutura de dados”. Essa característica tem facilitado sua integração com professores do ensino fundamental e médio, viabilizando a continuidade de projetos educacionais nas escolas. Ademais, o PC desempenha um papel importante na redução de barreiras relacionadas à programação, popularizando ferramentas visuais que priorizam a resolução de problemas e minimizam a complexidade da sintaxe tradicional.

Embora o entendimento mais amplo do PC tenha favorecido sua popularização, ele

também trouxe desafios relacionados à definição de processos consistentes para o ensino e a avaliação dessa competência (ROMÁN-GONZÁLEZ *et al.*, 2019). A ausência de formação técnica em tecnologia da informação entre professores, especialmente no ensino fundamental e médio (BELLONI, 2002), representa um obstáculo significativo para implementar práticas que desenvolvam o PC. Assim, é essencial desenvolver métodos e ferramentas que ofereçam suporte tanto ao ensino quanto à avaliação do PC, alinhados às demandas contemporâneas da educação.

Compreende-se que, no cenário educacional contemporâneo, a avaliação deve ultrapassar a mera aferição de conteúdos memorizados, direcionando-se, prioritariamente, à análise do desenvolvimento de competências e habilidades nos estudantes. Nessa perspectiva, ao se adotar os pressupostos de Perrenoud (PERRENOUD, 2015) — para quem avaliar competências implica mais do que atribuir notas, exigindo a observação de como o estudante mobiliza, de forma integrada, saberes, habilidades e atitudes na resolução de situações-problema significativas — e os referenciais da taxonomia revisada de Bloom (ANDERSON; KRATHWOHL, 2001), que propõe uma estrutura hierarquizada e abrangente para a avaliação de habilidades cognitivas, torna-se possível a construção de instrumentos avaliativos mais alinhados à complexidade inerente ao desenvolvimento de competências. No contexto específico do PC, tal abordagem favorece a identificação de evidências concretas da capacidade do aluno em empregar processos de abstração, decomposição, reconhecimento de padrões e formulação de algoritmos para solucionar desafios computacionais.

Segundo (PERRENOUD, 2015), avaliar competências vai além da simples atribuição de notas; implica observar como o estudante mobiliza, de forma articulada, conhecimentos, habilidades e atitudes na resolução de problemas reais e significativos. Sob essa ótica, o Pensamento Computacional pode ser compreendido como uma competência complexa, que integra saberes conceituais sobre lógica e estruturas computacionais (conhecimentos), a capacidade de aplicar tais saberes por meio da abstração, decomposição, reconhecimento de padrões e formulação de algoritmos (habilidades), e também atitudes como a persistência, a criatividade e a disposição para resolver problemas de forma sistemática. Avaliar o desenvolvimento do PC, portanto, requer instrumentos capazes de captar essa mobilização integrada dos diferentes componentes que caracterizam uma competência.

1.2 Motivação

O desenvolvimento de habilidades relacionadas ao PC tem ganhado crescente relevância no cenário educacional, sobretudo após sua incorporação nas diretrizes da BNCC. Esse reconhecimento institucional tem impulsionado diversas iniciativas voltadas à promoção do pensamento computacional desde as etapas iniciais da educação básica (BARBOSA; MALTEMPI, 2020). No entanto, à medida que sua presença se consolida no currículo, emergem novos desafios — especialmente no que se refere à avaliação das habilidades associadas ao PC (ROMÁN-GONZÁLEZ *et al.*, 2019).

Entre os principais entraves identificados está a complexidade inerente à avaliação de habilidades como abstração, decomposição, reconhecimento de padrões e elaboração de algoritmos. Tais competências, por sua natureza multidimensional e processual, exigem métodos avaliativos mais estruturados, criteriosos e contínuos — características que nem sempre são atendidas pelas abordagens tradicionais focadas em exames e provas para verificar mais o quanto o estudante memorizou ou domina o conteúdo ensinado do que em avaliar quais habilidades ou competências os estudantes desenvolveram. Por exemplo, a abstração requer domínio de habilidades intelectuais superiores; por isso, a Taxonomia de Bloom propõe estágios de aquisição dessas habilidades, que vão das mais básicas às mais complexas.

Adicionalmente, muitos docentes da educação básica, embora engajados com práticas pedagógicas inovadoras, não possuem formação específica em Ciência da Computação. Essa lacuna dificulta tanto a apropriação conceitual quanto a operacionalização didática dos elementos técnicos do PC (BELLONI, 2002). A granularidade presente nos componentes avaliativos do PC (ARAUJO *et al.*, 2016), aliada à limitada familiaridade com linguagens de programação, contribui para que a avaliação se configure como um dos principais gargalos na consolidação do ensino de pensamento computacional. Diante desse cenário, a automatização da avaliação desponta como uma estratégia promissora. Ela possibilita a padronização do processo avaliativo (SANTOS *et al.*, 2017).

Assim, a presente tese fundamenta-se em motivações de natureza pedagógica e tecnológica, com o objetivo de superar lacunas metodológicas. Ao propor um sistema automatizado para a avaliação do pensamento computacional, busca-se não apenas apoiar o trabalho docente, mas também contribuir para a consolidação do ensino de PC na educação básica.

1.3 Hipótese

A hipótese desta pesquisa é que a adoção de uma abordagem avaliativa hierárquica-quali-quantitativa e de um artefato baseado na análise semi-automatizada de códigos de programação proporciona suporte aos professores no processo de avaliação do PC em alunos do Ensino Médio.

1.4 Questões de Pesquisa

A avaliação semi-automatizada de códigos de programação melhora o processo de avaliação do Pensamento Computacional?

Além disso, as seguintes questões complementares norteiam essa tese de doutorado:

1. De que forma um artefato de avaliação baseado na análise de códigos de programação pode auxiliar o professor no processo de avaliação do PC em alunos do Ensino Médio?
   Objetivo: Investigar de que forma um artefato tecnológico, fundamentado na análise semi-automatizada de códigos de programação, pode apoiar o trabalho docente no processo de avaliação do PC em alunos do Ensino Médio.
2. Como os professores percebem as contribuições advindas do uso de uma plataforma de suporte à avaliação do Pensamento Computacional?
   Objetivo: Investigar as percepções dos professores acerca das contribuições proporcionadas pelo uso de um artefato tecnológico de suporte à avaliação do Pensamento Computacional, especialmente no contexto dos processos de ensino e aprendizagem.
3. De que forma a análise semi-automatizada de códigos contribui para uma avaliação mais focada dos diferentes aspectos ensino do PC (por exemplo, decomposição, abstração, reconhecimento de padrões, algoritmo)?
   Objetivo: Verificar se a plataforma consegue capturar múltiplos indicadores ou dimensões do PC, não se restringindo apenas à correção sintática.
4. Qual o grau de precisão ou concordância das avaliações semi-automatizadas em relação às avaliações realizadas por especialistas ou pelos próprios professores?
   Objetivo: Medir a assertividade das notas e *feedbacks* gerados pela plataforma, comparando-os ao julgamento humano.
5. Como diferentes modelos de linguagem de grande escala (LLM) se comportam no processo de avaliação do Pensamento Computacional?

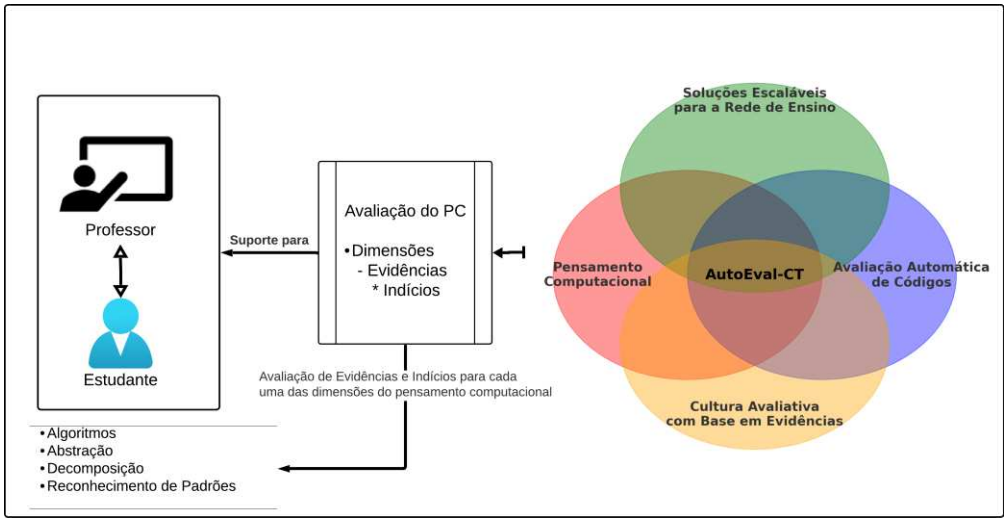
Objetivo: Analisar e comparar o desempenho de distintos modelos de linguagem de grande escala (LLM) para auxiliar o processo de avaliação do PC.

1.5 Abordagem Avaliativa e Artefato

Diante do exposto, foram desenvolvidos uma abordagem avaliativa que integra uma abordagem avaliativa hierárquica combinando análises qualitativas (quali) e quantitativas (quanti) e um artefato que configura-se como um instrumento para a avaliação semi-automatizada do PC, fundamentado na análise de códigos de programação produzidos por estudantes. Na abordagem avaliativa, cada uma das quatro habilidades do PC — algoritmos, abstração, decomposição e reconhecimento de padrões (BARR; STEPHENSON, 2011a) — é subdividida em evidências, e cada evidência, por sua vez, é detalhada em indícios específicos. A automação da análise oferece suporte efetivo ao professor no processo de avaliação do PC, considerando que essa tarefa é inerentemente complexa e se torna ainda mais desafiadora diante do elevado volume de códigos e do número expressivo de estudantes a serem avaliados.

Além disso, o instrumento viabiliza a realização de avaliações em maior escala e apresenta flexibilidade para a aplicação de uma multiplicidade maior de processos avaliativos (SCRIVEN, 1966; LUCKESI, 2011). A Figura 1 apresenta uma visão holística do funcionamento do artefato, que se posiciona na intersecção entre Pensamento Computacional, Avaliação Semi-automatizada de Códigos, Soluções Escaláveis para Redes de Ensino e Cultura Avaliativa Baseada em Evidências.

Figura 1 – Visão geral da abordagem avaliativa



Fonte: elaborada pelo autor (2025).

A utilização de um artefato semi-automatizado no processo de avaliação do PC não tem como objetivo eliminar a subjetividade ou a autonomia do professor na interpretação pedagógica, mas sim oferecer subsídios concretos e confiáveis que possibilitem um retorno mais objetivo e seguro sobre o desempenho dos estudantes. Ao apoiar-se em evidências geradas de forma sistemática, o docente pode concentrar seus esforços na análise qualitativa das aprendizagens e na adoção de estratégias didáticas mais eficazes, promovendo um processo formativo mais centrado no desenvolvimento das habilidades em PC.

Mais do que a obtenção de resultados imediatos ou classificatórios, o que realmente importa é a consolidação de uma aprendizagem segura, que assegure ao estudante a internalização significativa dos conhecimentos e o desenvolvimento das competências propostas.

1.6 Estrutura do Documento

O restante desta de tese está organizada em oito capítulos, cujos conteúdos são brevemente descritos a seguir:

- **Capítulo 2 — Fundamentação Teórica**
  Expõe os conceitos, modelos e pressupostos que embasam a investigação, oferecendo o referencial necessário para compreender o escopo do estudo.
- **Capítulo 3 — Trabalhos Relacionados**
  Apresenta o estado da arte por meio de um levantamento sistemático de pesquisas e ferramentas correlatas, situando esta tese no panorama atual do campo.
- **Capítulo 4 — Metodologia**
  Descreve a abordagem metodológica adotada, detalhando procedimentos, instrumentos, participantes e critérios de análise empregados na condução do trabalho.
- **Capítulo 5 - Perspectivas, desafios e oportunidades na avaliação do Pensamento Computacional**
  Este capítulo apresenta uma investigação com foco na avaliação do PC que se desenvolveu em duas fases: um Mapeamento Sistemático da Literatura (MSL) e um *survey* aplicado com pesquisadores cujos trabalhos foram incluídos no MSL.
- **Capítulo 6 — Avaliação Quali-Quanti do Pensamento Computacional**
  Detalha a abordagem avaliativa utilizada, contemplando as fases qualitativa e quantitativa, bem como os indicadores empregados para mensuração do PC.
- **Capítulo 7 — Projeto e Desenvolvimento do Artefato**

Documenta o projeto e a implementação do artefato concebido, abrangendo requisitos, arquitetura, tecnologias e funcionalidades.

- **Capítulo 8 — Revisão por Especialistas**

  Apresenta os resultados da revisão conduzida com especialistas em PC, destacando evidências de confiabilidade e sugestões de aprimoramento.

- **Capítulo 9 — Resultados**

  Apresenta os resultados obtidos na Etapa de Experimentação. Os dados aqui expostos compreendem os experimentos realizados com diferentes *Large Language Models* (LLMs), a comparação entre a avaliação do artefato AutoEval-CT e a de um especialista humano, e, por fim, o experimento controlado conduzido em uma turma do Ensino Médio.

- **Capítulo 10 — Considerações Finais**

  Resume as principais conclusões, apresenta as limitações do estudo. Sistemiza as contribuições teóricas e propõe direções de continuidade da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

O desenvolvimento desta pesquisa demanda a construção de uma base conceitual que permita compreender os princípios, desafios e possibilidades relacionados ao ensino e à avaliação do Pensamento Computacional (PC) no contexto educacional. Este capítulo tem como objetivo apresentar a fundamentação teórica que sustenta esta tese, abordando os principais conceitos, definições e interpretações sobre o PC, sua inserção no ambiente educacional e as estratégias existentes para sua avaliação. Além disso, discute-se o desenvolvimento e a estrutura do artefato "Computacionalidade", proposto como instrumento de suporte à análise e à avaliação de evidências de PC em atividades educacionais. Por fim, são exploradas as abordagens atuais de correção automática de códigos de programação, destacando seus benefícios, desafios e contribuições para a prática pedagógica no desenvolvimento de habilidades associadas ao PC. Essa fundamentação é essencial para contextualizar as decisões metodológicas e orientar a análise dos resultados apresentados nos capítulos subsequentes.

2.1 Definição do Pensamento Computacional

O PC é uma abordagem cognitiva para a formulação e resolução sistemática de problemas que utiliza conceitos da Ciência da Computação. O termo foi popularizado por Jeannette Wing (2006), que argumentou que essa habilidade é essencial não apenas para cientistas da computação, mas para qualquer pessoa (WING, 2006a).

Em sua definição mais difundida e aceita ele pode ser subdividido em quatro pilares principais (WING, 2006a; WING, 2011):

- **Decomposição:** Divisão de um problema complexo em partes menores e mais manejáveis.
- **Reconhecimento de padrões:** Identificação de similaridades dentro dos problemas e dados.
- **Abstração:** Redução de detalhes irrelevantes para focar nos aspectos essenciais do problema.
- **Algoritmos:** Criação de sequências de passos bem definidas para resolver problemas.

2.1.1 Ampliação do Conceito

Em 2010, foi realizado um esforço multidisciplinar visando estabelecer uma compreensão mais clara a respeito do conceito de PC. Nesse contexto, Barr e Stephenson (BARR;

STEPHENSON, 2011a) relatam a reunião de 26 estudiosos de diversas áreas com o propósito de discutir e aprofundar o significado desse termo. Embora não pretendesse criar uma definição definitiva, o grupo buscou chegar a um entendimento consistente do PC, sobretudo aplicado aos níveis de ensino fundamental e médio.

Esse projeto propôs-se uma definição de PC como um processo de resolução de problemas que abrange, mas não se limita, aos seguintes aspectos:

1. Formular problemas de modo que seja possível utilizar o computador e outras ferramentas para auxiliá-los na resolução;
2. Organizar e analisar dados de forma lógica;
3. Representar dados por meio de abstrações, tais como modelos e simulações;
4. Automatizar soluções por meio do pensamento algorítmico;
5. Identificar, analisar e implementar soluções com foco na obtenção da combinação mais eficiente e eficaz de etapas e recursos;
6. Generalizar e transferir esse processo de resolução de problemas para uma ampla variedade de situações.

A definição acima ressalta que o PC se fundamenta não apenas no uso de recursos computacionais para resolver problemas, mas também na habilidade de abstrair, automatizar e transferir soluções para diferentes contextos, ampliando o alcance das aplicações educacionais e científicas. Nesse sentido, a proposta demonstra a relevância de se refletir sobre os processos cognitivos envolvidos, ao mesmo tempo em que fomenta a integração desses conhecimentos no currículo escolar.

2.1.2 Outras Definições

Brackmann (BRACKMANN, 2017) ressalta que “O PC é uma distinta capacidade criativa, crítica e estratégica humana de saber utilizar os fundamentos da computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, por meio de passos claros, de tal forma que uma pessoa ou uma máquina possam executá-los eficazmente”.

Ainda segundo a compilação de definições apresentada por Brackmann (BRACKMANN, 2017), outros autores destacam diferentes dimensões do PC. Na literatura, por exemplo, encontram-se os seguintes entendimentos:

“São habilidades comumente utilizadas na criação de programas computacionais

> como uma metodologia para resolver problemas específicos nas mais diversas áreas". (BUNDY, 2007a; NUNES, 2011)
>
> "É o processo de reconhecer aspectos da computação em um mundo que nos cerca e aplicar ferramentas e técnicas da Ciência da Computação para entender e argumentar sobre sistemas e processos naturais e artificiais". (FURBER, 2012)
>
> "Pensar nos problemas de forma que um computador consiga solucioná-los. O Pensamento Computacional é executado por pessoas e não por computadores. Ele inclui o pensamento lógico, a habilidade de reconhecimento de padrões, raciocinar através de algoritmos, decompor e abstrair um problema". (LIUKAS, 2015)

É amplamente aceito que PC não se limita à Ciência da Computação, sendo aplicado em diversas áreas, como:

- **Ciências e Engenharia**: Modelagem matemática e análise de grandes volumes de dados (Big Data).
- **Medicina:** Desenvolvimento de diagnósticos assistidos por inteligência artificial.
- **Ciências Sociais:** Análise de tendências sociais e modelagem de comportamento.
- **Economia e Finanças:** Algoritmos para previsão de mercado e gestão de riscos.

A aplicação do PC tem impactado positivamente o desenvolvimento de tecnologias inovadoras em diversas áreas (SINGH *et al.*, 2024).

Observa-se, portanto, que cada definição realça aspectos específicos do PC — como criatividade, criticidade, uso de recursos tecnológicos, lógica e capacidade de abstração — evidenciando a complexidade do tema e a importância de se considerar as diferentes abordagens ao aplicá-lo em contextos educacionais e de pesquisa. Desse modo, o PC tornou-se um elemento-chave para o desenvolvimento de habilidades cognitivas avançadas, promovendo uma abordagem estruturada para a solução de desafios complexos em diversas áreas do conhecimento (HAMIDI, 2024).

2.2 O Pensamento Computacional na Educação

Seguindo uma tendência mundial, em dezembro de 2018, o Ministério da Educação homologou, no Brasil, a Base Nacional Comum Curricular BNCC (BNCC, 2018). Esse documento estabelece diretrizes para a educação básica e, dentre outras disposições, inclui a tecnologia da informação, a codificação e a resolução de problemas em sua quinta competência (de um total de dez).

Denominada **Literacia Digital**, essa competência está descrita na BNCC da seguinte forma:

> *"5 - Literacia Digital: Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva."*

Além de estar explicitamente definido na quinta competência, o PC também auxilia no desenvolvimento de outras competências gerais ali descritas, como:

- Competência 1: Compreensão do mundo;
- Competência 2: Pensamento científico, criativo e crítico;
- Competência 4: Comunicação;
- Competência 7: Argumentação;
- Competência 9: Empatia e cooperação.

Dessa forma, o PC emerge como uma ferramenta relevante para a consolidação das práticas e objetivos propostos pela BNCC, ampliando o potencial dos estudantes para atuarem de forma crítica, criativa e participativa na sociedade. A implementação do PC na educação tem sido defendida como uma forma de desenvolver o raciocínio lógico e preparar os alunos para o mercado de trabalho digital (SANTIAGO *et al.*, 2024).

Algumas estratégias educacionais incluem:

- **Uso de Programação e Robótica:** Plataformas como Scratch e Python permitem o desenvolvimento das habilidades de PC desde a infância.
- **Gamificação:** Jogos educativos estimulam o pensamento lógico e a criatividade na solução de problemas.
- **Atividades Desplugadas:** Exercícios sem o uso de computadores ajudam no desenvolvimento do raciocínio lógico.

A introdução dessas estratégias melhora o desempenho acadêmico e a capacidade de abstração dos alunos (COOPER *et al.*, 2010).

2.3 Avaliação do Pensamento Computacional

Um dos desafios fundamentais do PC é sua avaliação, pois envolve múltiplos aspectos, desde a decomposição de problemas até a formulação de algoritmos. De acordo com (SANDRI *et al.*, 2025), a avaliação do PC exige abordagens multidimensionais que vão além das provas tradicionais, incorporando métricas qualitativas e quantitativas.

Nos últimos anos, pesquisadores têm desenvolvido diferentes métodos para tentar medir o "nível" de PC em estudantes, abrangendo desde questionários e testes padronizados até avaliações baseadas em observação de comportamento e desempenho prático em atividades de programação e resolução de problemas (BORELA *et al.*, 2025).

A avaliação do PC pode ser realizada de diversas maneiras, cada uma com seus benefícios e limitações. Os métodos mais comuns incluem:

- **Avaliação Baseada em Testes:** Testes estruturados e padronizados são frequentemente usados para medir habilidades computacionais. Testes podem incluir questões de lógica, decomposição de problemas e pseudocódigo para avaliar o nível de compreensão do aluno sobre os conceitos fundamentais do PC (BORELA *et al.*, 2025).
- **Avaliação Baseada em Projetos:** A análise de projetos desenvolvidos por estudantes, como jogos, programas ou simulações, tem sido uma abordagem amplamente utilizada. O autor de (SAWADA *et al.*, 2025), destaca que a avaliação de projetos permite uma visão mais holística da aplicação do PC em contextos práticos.
- **Observação e Análise de Processos:** A observação do processo de resolução de problemas é uma técnica qualitativa útil para medir habilidades de PC. Estudos como o de (ZINN, 2025) sugere que o registro de ações do aluno ao resolver problemas computacionais fornece informações valiosas sobre seu raciocínio lógico.
- **Avaliação Baseada em Inteligência Artificial:** a integração de IA na avaliação é uma área emergente e promissora (FREITAS *et al.*, 2025). Sistemas de IA podem potencialmente analisar padrões de resolução de problemas, fornecer *feedback* personalizado e identificar dificuldades específicas dos alunos, contribuindo para uma avaliação mais eficaz e adaptativa do PC (NASCIMENTO *et al.*, 2024).

A avaliação do PC ainda enfrenta desafios significativos:

- **Falta de Padronização:** Não há um consenso global sobre a melhor forma de avaliar o PC.
- **Dificuldade em Medir Processos Mentais:** Muitas habilidades computacionais envolvem

processos internos que são difíceis de quantificar com precisão.

- **Desafios Tecnológicos:** A implementação de ferramentas avançadas de avaliação requer infraestrutura e formação específica e adequada para os professores.

2.3.1 Classificação de Ferramentas de Avaliação do Pensamento Computacional

A literatura apresenta diferentes abordagens para avaliar o PC, que podem ser categorizadas em seis principais tipos de ferramentas: diagnósticas, somativas, formativas-iterativas, mineração de dados, transferência de habilidades e percepção-atitude (ROMÁN-GONZÁLEZ *et al.*, 2019).

2.3.1.1 Ferramentas Diagnósticas (CT Diagnostic Tools)

As ferramentas diagnósticas têm como principal objetivo medir o nível de aptidão do sujeito em PC antes ou depois de uma intervenção educacional. Elas são utilizadas em condições de pré-teste para avaliar o conhecimento prévio e em condições de pós-teste para verificar a evolução do aprendizado. Exemplos de ferramentas dessa categoria incluem:

- Computational Thinking Test (GONZÁLEZ, 2015)
- Test for Measuring Basic Programming Abilities (MÜHLING *et al.*, 2015)

2.3.1.2 Ferramentas Somativas (CT Summative Tools)

As ferramentas somativas têm como propósito avaliar se o aluno adquiriu conhecimento suficiente após um processo de ensino. Essas ferramentas são utilizadas em pós-testes para medir a aprendizagem e compreensão dos conceitos de PC. Entre os exemplos, destacam-se:

- Avaliação no contexto do Scratch (MEERBAUM-SALANT; BEN-ARI, 2013)
- Quizly (App Inventor) (MAIORANA *et al.*, 2015)

2.3.1.3 Ferramentas Formativas-Iterativas (CT Formative-Iterative Tools)

Esse tipo de ferramenta tem como foco fornecer feedback automatizado ao aluno durante o processo de aprendizado, visando a melhoria contínua das habilidades de PC. Diferente das ferramentas somativas, elas não avaliam o indivíduo diretamente, mas sim seus produtos de aprendizagem, como projetos de programação. Algumas ferramentas dessa categoria incluem:

- Dr. Scratch (MORENO-LEÓN; ROBLES, 2015)

– Ninja Code Village (Scratch) (OTA *et al.*, 2016)

2.3.1.4 Ferramentas de Mineração de Dados (CT Data-Mining Tools)

Diferentemente das ferramentas formativas-iterativas, as ferramentas de mineração de dados registram e analisam a atividade do aluno em tempo real, fornecendo insights sobre os processos cognitivos envolvidos na aquisição do PC. Essas ferramentas permitem detectar lacunas e dificuldades no aprendizado por meio de métricas e análises de big data. Exemplos incluem:

– Blockly environment (GROVER *et al.*, 2017)
– Kodetu (EGUíLUZ *et al.*, 2020)

2.3.1.5 Ferramentas de Transferência de Habilidades (CT Skill Transfer Tools)

Essas ferramentas têm como objetivo avaliar a capacidade do aluno de aplicar habilidades de PC em diferentes contextos. Elas são úteis para medir a retenção do conhecimento e a transferência de competências após um período de aprendizado. Alguns exemplos incluem:

– Bebras Tasks (DAGIENĖ; FUTSCHEK, 2008)
– CTP-Quis (BASAWAPATNA *et al.*, 2011)

2.3.1.6 Ferramentas de Avaliação de Percepções e Atitudes (CT Perceptions–Attitudes Scales)

Além de medir habilidades técnicas, é fundamental avaliar as percepções e atitudes dos indivíduos em relação ao PC, pois essas variáveis influenciam diretamente a motivação e o desempenho no aprendizado. Algumas escalas reconhecidas para essa avaliação incluem:

– Computational Thinking Scales (CTS) (KORKMAZ *et al.*, 2017)
– Computational Thinking Skills Scale (CTSS) (DURAK; SARITEPECI, 2018)

2.4 Correção Automática de Exercícios de Programação

Os sistemas de correção automática podem ser classificados em diferentes categorias, dependendo da abordagem adotada para avaliar os códigos dos alunos. Algumas das principais técnicas incluem:

2.4.1 Testes Unitários e Programação Orientada a Aspectos

Uma das abordagens para a correção automática de código em linguagens como C é baseada em testes unitários e programação orientada a aspectos. Sterbini e Temperini (STERBINI; TEMPERINI, 2004) apresentaram um sistema onde os professores criam testes unitários para funções específicas e fornecem uma implementação de referência. O código dos alunos é testado contra esses critérios e recebe feedback automático sobre os erros identificados. Esse método permite um alto grau de precisão na correção e auxilia os alunos a compreenderem os erros em seus códigos.

2.4.2 Correção Baseada em Análise de Código e Feedback Imediato

Alguns sistemas utilizam análise estática de código para verificar boas práticas de programação, qualidade do código e até mesmo a presença de erros semânticos. Esses sistemas fornecem feedback imediato, ajudando os alunos a corrigirem seus erros antes de submeterem suas soluções. Essa abordagem é especialmente útil para iniciantes, pois permite um aprendizado iterativo e uma melhor compreensão das regras da linguagem de programação (VANACORE *et al.*, 2024).

2.4.3 Correção com IA e Aprendizado de Máquina

Com o avanço da inteligência artificial, novas abordagens têm surgido para a correção automática de exercícios de programação. Modelos baseados em aprendizado de máquina podem identificar padrões de erro comuns e sugerir correções adaptadas ao perfil do aluno. Além disso, redes neurais têm sido exploradas para entender o estilo de programação do aluno e oferecer sugestões mais personalizadas (TERRAGNI; GIACAMAN, 2025).

2.4.4 Benefícios da Correção Automática

A incorporação de sistemas automáticos para a correção de códigos de programação proporciona uma série de benefícios significativos tanto para os docentes quanto para os discentes. Dentre os principais benefícios, destacam-se:

- **Otimização do tempo docente**: A automação libera o professor de tarefas avaliativas repetitivas, permitindo que ele dedique mais tempo a atividades analíticas e pedagógicas de maior complexidade, como o acompanhamento individualizado dos estudantes e o

planejamento de intervenções didáticas mais eficazes.

- **Superação de limitações na formação do avaliador**: Muitos professores não recebem uma formação sistemática e aprofundada sobre avaliação educacional, o que pode comprometer a precisão e a confiabilidade dos processos avaliativos (VIANNA, 2014). Nesse sentido, a utilização de sistemas automáticos auxilia na padronização da correção e no fornecimento de parâmetros consistentes, especialmente em tarefas com critérios bem definidos, contribuindo para a redução da subjetividade e para a qualificação do julgamento pedagógico.
- **Feedback imediato ao estudante**: A disponibilização de resultados logo após a submissão do código permite que os alunos identifiquem e corrijam seus erros de forma autônoma e em tempo hábil, promovendo um aprendizado mais ativo e significativo.
- **Redução de vieses e inconsistências**: (VIANNA, 2014) também aponta que o avaliador pode ser influenciado por fatores externos à aprendizagem, como expectativas prévias, fadiga ou pressões institucionais. A correção automatizada, ao aplicar critérios pré-estabelecidos de forma objetiva, contribui para minimizar esses vieses, assegurando maior equidade entre os avaliados.
- **Escalabilidade da avaliação**: A utilização de sistemas automáticos torna viável a correção de um grande volume de códigos, possibilitando que instituições de ensino atendam a turmas numerosas sem comprometer a qualidade da avaliação ou sobrecarregar o corpo docente.

2.4.5 Desafios na Implementação

Apesar das vantagens, a correção automática de exercícios de programação também apresenta desafios, como:

- **Detecção de soluções alternativas**: Alunos podem chegar à resposta correta de maneiras diferentes, o que pode ser difícil para alguns sistemas de avaliação reconhecerem.
- **Análise semântica e lógica**: A mera verificação da sintaxe do código pode não ser suficiente para avaliar corretamente a lógica aplicada pelo aluno.
- **Personalização do feedback**: Fornecer feedback útil e personalizado ainda é um grande desafio, especialmente para sistemas baseados em regras fixas.

2.5 Tipos de Avaliação: Diagnóstica, Formativa e Somativa

A avaliação educacional é um processo essencial para a compreensão e o aprimoramento do ensino e da aprendizagem. Entre as distintas classificações possíveis, destaca-se a tipologia proposta por (SCRIVEN, 1966), que introduziu conceitos fundamentais para distinguir as funções avaliativas no contexto educacional, diferenciando a avaliação formativa da somativa. Complementarmente, (LUCKESI, 2022) contribui significativamente para a discussão, ao enfatizar o caráter emancipador da avaliação e seu papel na promoção da aprendizagem.

2.5.1 Avaliação Diagnóstica

A avaliação diagnóstica ocorre, em geral, antes do início de um processo educativo ou de um novo conteúdo, tendo como objetivo identificar conhecimentos prévios, habilidades já consolidadas e possíveis lacunas ou dificuldades apresentadas pelos estudantes. Para Luckesi, essa etapa é essencial, pois permite ao educador planejar intervenções adequadas e construir estratégias pedagógicas compatíveis com as necessidades reais da turma. Assim, a avaliação diagnóstica não visa atribuir notas, mas subsidiar o ensino de forma mais eficaz e personalizada (LUCKESI, 2022).

2.5.2 Avaliação Formativa

(SCRIVEN, 1966) cunhou o termo *formative evaluation* para designar a avaliação realizada durante o processo, com a finalidade de monitorar e aprimorar o ensino em curso. Seu foco não é simplesmente registrar desempenhos, mas fornecer informações que retroalimentem o processo de aprendizagem, permitindo ajustes pedagógicos em tempo hábil. Para (LUCKESI, 2022), a avaliação formativa é eminentemente mediadora, pois busca orientar o estudante em sua trajetória, identificando avanços e dificuldades de forma contínua. Dessa forma, assume caráter eminentemente processual e qualitativo, sendo instrumento de intervenção pedagógica e não mero julgamento.

Segundo (PERRENOUD, 2015), a avaliação formativa deve ser essencialmente pragmática e sistemática, baseada em uma observação contínua das ações e produções dos alunos. Isso implica ir além de instrumentos formais ou momentos específicos de verificação: exige sensibilidade para captar evidências cotidianas de aprendizagem, capacidade de interpretar indícios e compromisso com a melhoria do processo educacional. O professor, nesse contexto,

atua como um observador atento e analítico, que recolhe dados significativos em pequenos gestos, falas e produções dos estudantes, utilizando esses elementos para construir um retrato mais fiel das aprendizagens.

2.5.3 Avaliação Somativa

Em contrapartida, (SCRIVEN, 1966) define a *summative evaluation* como aquela destinada a julgar o valor ou mérito de um programa, curso ou processo, geralmente ao seu término. Na educação, corresponde à avaliação que ocorre ao final de um ciclo, disciplina ou etapa, com o objetivo de verificar se as metas educacionais foram alcançadas. Para (LUCKESI, 2022), a avaliação somativa tende a assumir caráter classificatório, estando frequentemente ligada à atribuição de notas, certificados ou decisões de promoção ou retenção. Contudo, ele adverte que, mesmo sendo somativa, essa avaliação não deve perder de vista o compromisso com o desenvolvimento do estudante e a justiça pedagógica.

2.6 Avaliação de Habilidades e Competências

A avaliação no contexto educacional contemporâneo vai além da verificação de conteúdos memorizados; ela se volta, sobretudo, para a compreensão do desenvolvimento de habilidades e competências por parte dos estudantes. Essa mudança de paradigma é impulsionada tanto por diretrizes curriculares atuais, como a BNCC, quanto por teóricos da educação que defendem abordagens avaliativas mais formativas, diagnósticas e integradas ao processo de aprendizagem.

Para (PERRENOUD, 2015), avaliar competências é mais do que atribuir notas; trata-se de observar como o aluno mobiliza, de forma articulada, conhecimentos, habilidades e atitudes para resolver problemas reais e significativos. Segundo o autor, a avaliação deve ser uma prática contínua e contextualizada, capaz de captar evidências da aprendizagem em situações concretas e de apoiar a regulação do ensino. Perrenoud argumenta que uma avaliação centrada nas competências exige do professor a capacidade de interpretar processos, analisar desempenhos e oferecer *feedbacks* que favoreçam o avanço do estudante.

Nesse mesmo sentido, a Taxonomia de Bloom revisada (ANDERSON; KRATHWOHL, 2001) oferece uma estrutura útil para pensar a avaliação de habilidades cognitivas de forma hierárquica e integrada. A versão revisada atualiza a proposta original de Bloom, deslocando o

foco de substantivos para verbos, e reorganiza os níveis cognitivos em: lembrar, compreender, aplicar, analisar, avaliar e criar. Essa estrutura permite ao professor planejar avaliações que vão desde tarefas de memorização até a resolução de problemas complexos e criativos, o que está diretamente alinhado com a proposta de desenvolver o PC.

2.7 Lógicas de Avaliação

A avaliação da aprendizagem, longe de ser um ato neutro ou puramente técnico, carrega em si diferentes concepções e finalidades que influenciam diretamente a prática pedagógica. (PERRENOUD, 2015), em sua obra *Dez Novas Competências para Ensinar*, distingue duas lógicas fundamentais que orientam as práticas avaliativas no contexto escolar: a lógica da excelência e da seleção e a lógica da regulação das aprendizagens.

A primeira lógica, predominante na tradição escolar, tem como finalidade hierarquizar, classificar e selecionar os estudantes a partir de seus desempenhos. Nesse modelo, prevalecem práticas que se apoiam em provas pontuais, notas e critérios normativos, frequentemente desconectados do processo de aprendizagem e das necessidades reais dos alunos. Essa lógica reforça desigualdades e tende a valorizar o produto final em detrimento do percurso formativo.

Em contraposição, Perrenoud defende uma segunda lógica: a da regulação das aprendizagens, sustentada por uma abordagem formativa da avaliação. Essa perspectiva compreende a avaliação como parte integrante e contínua do processo pedagógico, cuja função principal é diagnosticar dificuldades, identificar avanços e ajustar as estratégias de ensino para favorecer o desenvolvimento de cada estudante. Trata-se de uma avaliação comprometida com a aprendizagem, que visa não apenas medir, mas intervir e orientar.

Assim, a avaliação, na visão de Perrenoud, deixa de ser um fim em si mesma para tornar-se um mecanismo de regulação didática. Ela compõe o próprio ato pedagógico, integrando-se ao processo de ensino-aprendizagem como um instrumento de apoio à construção do conhecimento. Ao adotar essa abordagem, o professor transforma a avaliação em uma prática intencional e reflexiva, voltada para a promoção da aprendizagem de forma equitativa, contínua e significativa.

2.8 Considerações Finais

O aprofundamento teórico desenvolvido neste capítulo evidencia que o PC constitui uma competência transversal, essencial para a formação de sujeitos aptos de enfrentar desafios complexos em uma sociedade cada vez mais mediada pela tecnologia. As diferentes definições e abordagens apresentadas reforçam a importância de compreender o PC não apenas como um conjunto de habilidades técnicas, mas como uma forma de raciocínio aplicável em múltiplos contextos. No campo educacional, sua incorporação é impulsionada por diretrizes como a BNCC, que reconhecem a necessidade de preparar os estudantes para um mundo digital, crítico e criativo. Entretanto, a avaliação do PC permanece como um desafio relevante, especialmente diante da ausência de instrumentos padronizados e da dificuldade em mensurar processos cognitivos complexos.

3 TRABALHOS RELACIONADOS

Este capítulo tem como objetivo apresentar e discutir os principais trabalhos relacionados à avaliação do PC, com foco nas iniciativas que propõem ferramentas, modelos ou plataformas tecnológicas. São analisadas, particularmente, as contribuições dos artefatos Computacionalidade, CodeMaster e Dr. Scratch, bem como de estudos que, embora não proponham ferramentas específicas, oferecem reflexões e fundamentos relevantes sobre estratégias avaliativas no contexto do PC. A partir dessa análise, busca-se evidenciar os avanços, limitações e oportunidades existentes, que fundamentam e justificam a abordagem desenvolvida nesta tese.

A identificação dos trabalhos relacionados que fundamentam este capítulo foi realizada a partir dos MSL conduzidos no decorrer desta pesquisa.

3.1 Computacionalidade

O artefato "Computacionalidade" foi desenvolvido com o propósito de apoiar docentes no processo de análise e avaliação das soluções desenvolvidas por estudantes em atividades cujo objetivo é exercitar o PC. Conforme descrito em (OLIVEIRA; PEREIRA, 2023), esse artefato busca fornecer uma estrutura clara e rigorosa para conduzir a avaliação de diferentes aspectos do PC, indo além da verificação do resultado final da atividade.

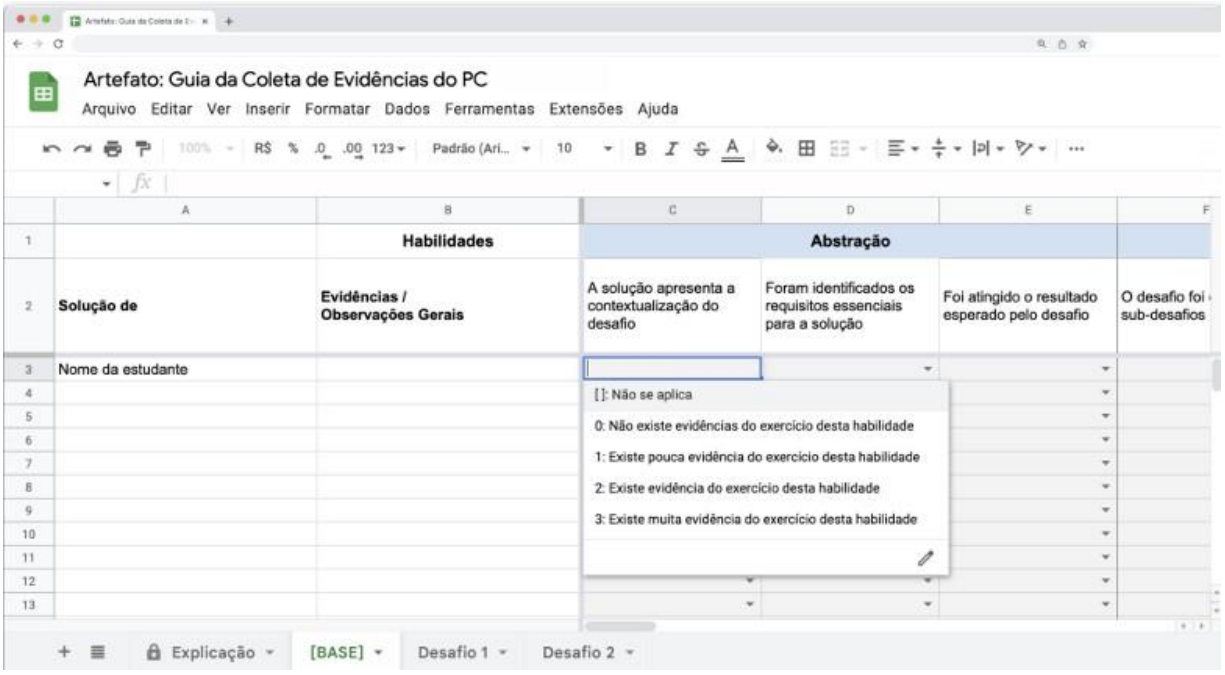
Os autores propõem um artefato voltado à coleta de evidências do exercício do PC no Ensino Superior em Computação, denominado Computacionalidade. A ferramenta concentra-se na análise de atividades práticas, como exercícios de programação, com o objetivo de identificar traços do raciocínio computacional manifestados pelos estudantes, fornecendo subsídios para intervenções pedagógicas mais direcionadas.

Embora desenvolvido para o contexto universitário, o estudo apresenta abordagens e métricas passíveis de adaptação a diferentes níveis de ensino, o que evidencia a importância de ferramentas sistemáticas para a avaliação do PC em situações reais de aprendizagem. No entanto, o artefato descrito por (OLIVEIRA; PEREIRA, 2023) depende da análise humana dos dados coletados, não realizando a avaliação de forma automatizada. Esse aspecto reforça uma lacuna significativa na área: a ausência de soluções capazes de promover avaliações automáticas e fornecer feedback imediato — exatamente o foco da abordagem apresentada neste trabalho.

A Figura 2 ilustra a interface do artefato Computacionalidade.

O artefato, é composto por: (i) um conjunto de habilidades e suas definições; (ii) um

Figura 2 – *Print* de tela do artefato computacionalidade



Fonte: Oliveira e Pereira (2023).

conjunto de evidências que devem ser analisadas; e (iii) uma escala simplificada para mensurar as evidências identificadas.

3.1.1 Habilidades e Evidências

O artefato "Computacionalidade" identifica um conjunto de habilidades relacionadas ao PC, que podem ser consultadas na Tabela 1 do artefato original. Para cada uma dessas habilidades, descrevem-se critérios que ajudam a identificar evidências de seu exercício. Dessa forma, o artefato orienta o professor sobre:

1. **O que coletar:** Quais informações ou características do código ou da solução devem ser observadas para verificar a presença da habilidade.
2. **Como coletar:** Instruções sobre como analisar e registrar cada evidência, de modo a garantir maior consistência na avaliação.

As evidências de cada habilidade foram detalhadas no artefato original, apresentando orientações sobre como reconhecê-las no processo de resolução das atividades.

3.1.2 Escala de Avaliação

Para mensurar o grau de exercício de cada habilidade, o artefato propõe uma escala simplificada, inspirada em (FRANÇA; SILVA, 2020). Cada habilidade pode ser classificada de acordo com o nível de evidência observado, seguindo as opções:

Não se aplica: A habilidade não é relevante para a atividade em questão.

0 **Não existe evidência:** Não há indícios de que a habilidade tenha sido exercitada.

1 **Pouca evidência:** Há indícios mínimos, porém não suficientemente claros ou consistentes.

2 **Evidência presente:** A habilidade é identificável na solução de forma satisfatória.

3 **Muita evidência:** A habilidade está fortemente presente, com múltiplos exemplos de aplicação.

Esta escala facilita a verificação de cada habilidade e permite um retorno mais objetivo quanto ao grau de proficiência demonstrado pelo estudante no contexto de PC.

3.1.3 Suporte Ferramental

Foi utilizada a plataforma Google Sheets[1]. A escolha se deveu à facilidade de uso, ampla familiaridade por parte de professores e estudantes, disponibilidade via web e recursos colaborativos.

Nesse ambiente, cada planilha do artefato contempla:

- Uma visão geral, apresentando a definição das habilidades e as evidências a serem coletadas.
- Uma planilha base para cada atividade (desafio), onde as linhas representam estudantes e as colunas contêm caixas de seleção para cada evidência.

Ao adotar essa estrutura, "Computacionalidade" possibilita o registro das avaliações e a organização dos dados, fornecendo maior visibilidade sobre o progresso de cada estudante.

3.1.4 Benefícios e Perspectivas

A principal contribuição do artefato "Computacionalidade" está em detalhar as habilidades do PC e em propor uma forma sistemática de coleta de evidências de seu exercício. Por meio da enumeração de evidências e do uso da escala de avaliação, o professor direciona a

[1] Qualquer outra ferramenta de planilha eletrônica, ou mesmo base de dados, poderia ser utilizada com a mesma finalidade.

atenção para diferentes aspectos relevantes, observando tanto o produto (solução final) quanto o processo de construção da solução.

Esse nível de detalhamento incentiva a reflexão na resolução de problemas, além de reforçar boas práticas de programação.

Em síntese, o artefato “Computacionalidade” oferece uma abordagem estruturada para avaliar o PC, simplificando a verificação de evidências e promovendo maior consistência nos feedbacks ao longo do processo de ensino-aprendizagem.

3.2 CodeMaster

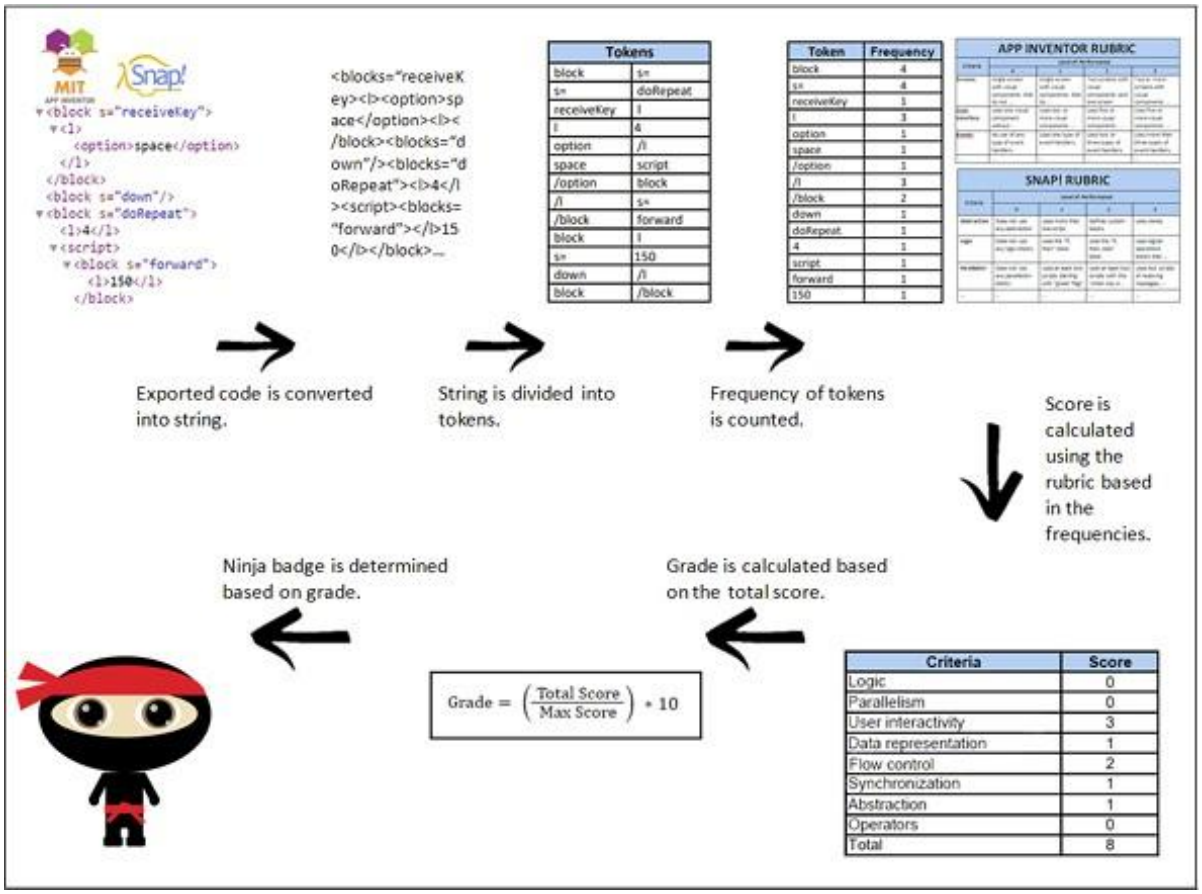
Von Wangenheim et al. (WANGENHEIM *et al.*, 2018) apresentam o CodeMaster, uma aplicação web que permite avaliar e classificar automaticamente projetos programados em App Inventor e Snap!, atribuindo pontuação com base em rubricas pré-definidas. A solução faz uso de uma análise estática de código, identificando padrões, blocos e estruturas específicas para gerar *feedback* imediato aos estudantes, enquanto oferece aos professores a possibilidade de avaliar turmas inteiras de forma mais ágil. Embora apresente benefícios significativos ao automatizar o processo de correção, o CodeMaster não realiza uma análise detalhada das habilidades de PC, pois seu foco principal recai na conformidade a padrões de implementação estipulados pela rubrica. Dessa forma, ainda que eficaz em contextos de linguagens de programação baseadas em blocos, a ferramenta prioriza critérios de codificação predefinidos, diferentemente da abordagem deste trabalho, que visa caracterizar e aferir dimensões específicas do PC.

A Figura 3 apresenta uma visão geral do processo de análise, avaliação e classificação do CodeMaster.

3.3 DrScratch

Moreno-León e Robles (MORENO-LEÓN; ROBLES, 2015) apresentam o Dr. Scratch, uma ferramenta que avalia automaticamente projetos desenvolvidos na plataforma Scratch, atribuindo um nível de Pensamento Computacional com base em métricas relacionadas à utilização de blocos e estruturas específicas (por exemplo, paralelização, abstração, sincronização e controle de fluxo). Embora Dr. Scratch forneça uma análise automatizada, centrada em atributos que refletem certas dimensões do PC, sua aplicação se restringe ao ambiente de programação em blocos, fazendo uso de uma rubrica sustentada na presença ou ausência de

Figura 3 – Processo de avaliação do CodeMaster



Fonte: Wangenheim *et al*. (2018).

padrões de código característicos do Scratch. Dessa forma, a ferramenta se mostra eficaz para introduzir e medir elementos do PC em linguagens visuais, mas não contempla, de forma direta, a avaliação de habilidades em linguagens textuais, como Python, nem uma análise mais detalhada dos diferentes aspectos que podem compor o PC em contextos de programação avançada.

A Figura 4 apresenta a interface de feedback gerado pela avaliação automática realizada pela ferramenta Dr. Scratch.

3.4 Outros Trabalhos

Além dos estudos que apresentam artefatos específicos para avaliação do PC, há investigações que, embora não proponham ferramentas propriamente ditas, dialogam diretamente com os objetivos desta pesquisa. Entre elas destaca-se o trabalho de (GREIFENSTEIN, 2021), que apresenta um projeto de pesquisa voltado à concepção de uma plataforma de *feedback* eficaz para programas Scratch desenvolvidos por alunos do ensino fundamental. O autor fundamenta sua proposta em três questões norteadoras: (i) que tipos de *feedback* sobre programas Scratch

Figura 4 – *Print* de tela de *feedback* do Dr. Scratch

Fonte: Moreno-León e Robles (2015).

podem ser oferecidos às crianças e se mostram efetivos; (ii) quais competências — em termos de conhecimento de conteúdo e conhecimento pedagógico — os docentes necessitam para oferecer *feedback* de qualidade; e (iii) em que medida ferramentas de análise automatizada podem auxiliar os professores nesse processo.

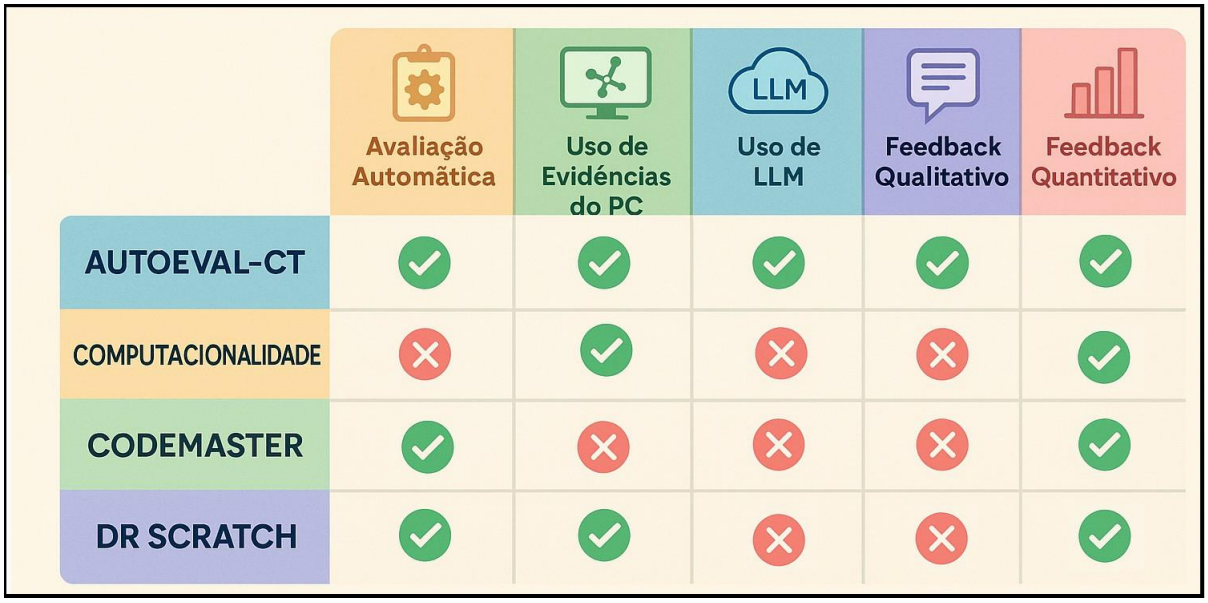
Outra contribuição relevante é o levantamento integrativo de literatura conduzido por (LIU *et al.*, 2021), cujo objetivo foi mapear estratégias de avaliação de PC. O estudo foi organizado em duas fases. Na primeira, realizou-se uma análise crítica dos métodos existentes, constatando-se que questionários e entrevistas constituem as abordagens mais recorrentes no contexto do ensino fundamental, com foco predominante nos artefatos computacionais produzidos pelos alunos e em seu desempenho em testes específicos de PC. Na segunda fase, os autores exploraram a literatura sobre técnicas de rastreamento ocular e protocolos de pensar-em-voz-alta para compreensão de processos cognitivos. A partir dessa síntese, recomendaram a inclusão combinada dessas metodologias aos instrumentos de avaliação, argumentando que tal integração pode oferecer novos *insights* sobre a aprendizagem de PC.

3.5 Análise Comparativa

Na análise dos trabalhos relacionados observa-se que as soluções atualmente disponíveis apresentam limitações quanto à capacidade de avaliar, de forma automatizada, dimensões mais complexas do PC, sobretudo no contexto de linguagens textuais. Além disso, poucas ferramentas incorporam abordagens baseadas em evidências cognitivas, nem oferecem feedback qualitativo robusto que auxilie efetivamente o professor no processo formativo.

A Figura 5 sintetiza, em um quadro comparativo, de que maneira distintas ferramentas direcionadas à avaliação do Pensamento Computacional atendem aos seguintes critérios: (i) avaliação automática; (ii) utilização de evidências de PC; (iii) emprego de LLM; (iv) oferta de *feedback* qualitativo; e (v) oferta de *feedback* quantitativo. O quadro inclui a ferramenta AUTOEVAL-CT — objeto central desta tese — e, para fins de contraste, as soluções Computacionalidade (OLIVEIRA; PEREIRA, 2023), CodeMaster (WANGENHEIM *et al.*, 2018) e Dr. Scratch (MORENO-LEÓN; ROBLES, 2015).

Figura 5 – Quadro comparativo de ferramentas de avaliação do pc



Fonte: elaborada pelo autor (2025).

4 METODOLOGIA

Este capítulo descreve os procedimentos metodológicos adotados para conduzir esta pesquisa, que tem como objetivo principal o desenvolvimento e a avaliação de um artefato computacional para a avaliação do PC no contexto do Ensino Médio.

4.1 Metodologias que Nortearam esta Pesquisa

Esta pesquisa adota uma abordagem metodológica que combina princípios da pesquisa científica em Computação com métodos de investigação empírica aplicados à Educação. A condução do estudo foi orientada principalmente pela metodologia *Design Science Research* (DSR), complementada por *Mapeamentos Sistemáticos da Literatura* (MSL), *Pesquisa-Ação*, além da realização de *Experimento Controlado* e de pesquisas *Survey*. Esta combinação visa garantir robustez tanto na construção do artefato quanto na avaliação de sua aplicabilidade no contexto educacional.

Os experimentos que envolveram esta pesquisa foram submetidos e aprovados pelo Comitê de Ética em Pesquisa da Universidade Estadual Vale do Acaraú, sob o número Certificado de Apresentação de Apreciação Ética (CAAE) 88541225.6.0000.5053 e parecer 7.722.336. A Carta de Anuência da direção da escola e o parecer do CEP encontram-se no Anexo A.

4.1.1 Design Science Research (DSR)

O DSR constitui o eixo central desta pesquisa, uma vez que seu objetivo é projetar, desenvolver e avaliar um artefato tecnológico — o *AutoEval-CT* — destinado à avaliação semi-automatizada do Pensamento Computacional (PC) no Ensino Médio. Segundo (HEVNER *et al.*, 2004), o DSR é uma abordagem metodológica que busca gerar conhecimento científico a partir da criação e avaliação de artefatos que solucionam problemas práticos.

A aplicação do DSR nesta tese se deu por meio das seguintes etapas:

- **Exploração**: Investigação do problema, identificação das necessidades educacionais e compreensão dos desafios na avaliação do PC.
- **Aprofundamento**: Análise detalhada das soluções existentes, levantamento de requisitos e definição das dimensões de avaliação do PC.
- **Desenvolvimento**: Projeto, construção e implementação do artefato *AutoEval-CT*, incluindo suas funcionalidades, arquitetura e integração com modelos de *Large Language*

Models (LLMs).

- **Experimentação**: Avaliação do artefato por meio de testes com especialistas, análise de concordância com avaliações humanas, comparação entre diferentes modelos de LLM e experimento controlado aplicado em turmas de Ensino Médio.

Essa estrutura garante que o desenvolvimento do artefato seja conduzido de forma científica, iterativa e orientada a resultados práticos, mantendo o rigor acadêmico e a relevância social.

4.1.2 Mapeamento Sistemático da Literatura (MSL)

Os MSL desempenharam papel fundamental na fundamentação teórica e na identificação de lacunas de pesquisa. Foram conduzidos três MSL, com objetivos distintos:

- **MSL 1**: Investigação sobre o uso de programação em blocos aplicada ao ensino do Pensamento Computacional.
- **MSL 2**: Análise da adoção do Pensamento Computacional no contexto escolar brasileiro.
- **MSL 3**: Levantamento dos desafios, perspectivas e oportunidades na avaliação do Pensamento Computacional.

Cada mapeamento seguiu rigorosamente protocolos previamente definidos, que incluíram:

- Definição de perguntas de pesquisa;
- Estabelecimento de critérios de inclusão e exclusão;
- Estruturação de estratégias de busca em bases reconhecidas (SBC, IEEE, ACM, Scopus, entre outras);
- Análise e categorização dos dados extraídos.

Os resultados dos mapeamentos forneceram suporte teórico robusto para orientar tanto a modelagem do artefato quanto os critérios de avaliação utilizados na pesquisa.

4.1.3 Pesquisa-Ação

Além do DSR, a pesquisa incorpora os princípios da *Pesquisa-Ação*, uma abordagem metodológica que promove a intervenção direta no contexto estudado, com participação ativa dos sujeitos envolvidos no processo. Segundo (THIOLLENT; COLETTE, 2020), a Pesquisa-Ação é caracterizada por um processo cíclico e colaborativo que visa, simultaneamente, a geração de conhecimento científico e a transformação da prática.

Sua aplicação nesta tese ocorreu na etapa de desenvolvimento e experimentação de uma proposta de ensino de PC, estruturada a partir da integração entre Ação Computacional e *Design Thinking*. Essa proposta foi implementada em um curso ofertado a estudantes do primeiro semestre de Ciência da Computação, conduzido no contexto do Ensino Remoto Emergencial. Durante o curso, cada aluno foi desafiado a conceber e desenvolver um aplicativo para *smartphone*.

Esse processo permitiu:

- A constatação de que os conhecimentos dos participantes em relação ao PC foram alterados positivamente;
- A identificação de que os estudantes demonstraram motivação em aprofundar seus conhecimentos sobre PC após a experiência do curso;
- A verificação de que os aplicativos desenvolvidos apresentaram, segundo os instrumentos de avaliação utilizados, boa qualidade de código e adequação às propostas de projeto.

4.1.4 Experimento Controlado

A avaliação prática do artefato foi realizada também por meio de um *experimento controlado* aplicado em uma escola de Ensino Médio, envolvendo professores e alunos em um ambiente real de aprendizagem. Essa estratégia metodológica possibilitou:

- Avaliar a usabilidade e a eficácia do artefato no contexto educacional;
- Observar o impacto da automação na prática docente, especialmente no que tange à redução do tempo gasto na avaliação, aumento da escalabilidade e aprimoramento da qualidade do feedback;
- Coletar dados empíricos para comparação entre a avaliação semi-automatizada e a avaliação realizada por especialistas humanos.

O experimento seguiu um protocolo estruturado, com definição de objetivos, coleta sistemática de dados, aplicação de instrumentos (questionários, entrevistas e análises dos códigos produzidos pelos alunos) e análise dos resultados.

4.1.5 Survey

Adicionalmente, foi aplicado um *survey* junto a especialistas da área de Educação em Computação e Pensamento Computacional. O objetivo foi:

- Levantar percepções, desafios e expectativas quanto aos processos de avaliação do PC;

- Avaliar as características desejáveis para ferramentas de avaliação semi-automatizada;
- Obter feedback sobre a aplicabilidade do artefato desenvolvido.

O *survey* contribuiu para o refinamento do artefato e para a avaliação dos dados qualitativos e quantitativos obtidos ao longo da pesquisa.

4.1.6 Análises Qualitativas e Quantitativas

O processo de avaliação do artefato envolveu tanto análises qualitativas quanto quantitativas. As principais abordagens utilizadas foram:

- **Análise Qualitativa**: Utilização de análise de conteúdo para interpretar dados oriundos de entrevistas, feedbacks dos especialistas e registros observacionais.
- **Análise Quantitativa**: Aplicação de testes estatísticos (correlação de Pearson, regressão linear, teste de Friedm, entre outros) para comparar resultados, além da análise de respostas aos questionários.

4.2 Organização da Tese

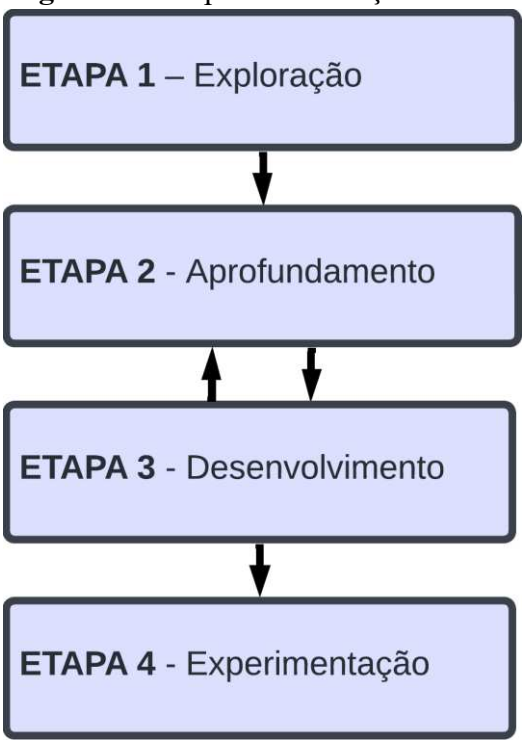
A execução desta tese de doutorado está organizada em quatro etapas interdependentes, conforme ilustrado na Figura 6. Em cada uma delas, emprega-se um desenho metodológico que combina abordagens qualitativas e quantitativas, possibilitando tanto um exame aprofundado dos fenômenos em estudo quanto uma avaliação mensurável dos resultados.

Esse arranjo tem por objetivo assegurar uma especificação detalhada das etapas de desenvolvimento, bem como promover uma evolução incremental e consistente da solução apresentada ao longo de todo o processo.

A **primeira etapa**, denominada **Exploração**, tem como principal objetivo a construção de um panorama consolidado do estado da arte sobre o tema, bem como a identificação de problemáticas emergentes que afetam o desenvolvimento e a aplicação do PC. Esse mapeamento inicial fornece bases sólidas para as investigações posteriores, permitindo compreender lacunas, tendências e oportunidades na área.

Na **segunda etapa**, voltada para o **Aprofundamento**, busca-se identificar desafios e oportunidades de pesquisa especificamente relacionados à avaliação da aprendizagem em PC. Nessa fase, delineiam-se questões mais específicas, definindo os requisitos e indicadores necessários para analisar a eficácia de abordagens, metodologias ou ferramentas voltadas ao

Figura 6 – Etapas da condução da tese



Fonte: elaborada pelo autor (2025).

ensino e à compreensão do PC, foi a partir dela que foram definidas os requisitos para o desenvolvimento do artefato.

A **terceira etapa** concentra-se no **Desenvolvimento**. Consiste na concepção e implementação de uma solução ou ferramenta, construída com base nos insights obtidos nas fases anteriores. Sempre que necessário, realizam-se revisões e refinamentos pautados nas descobertas da etapa de Aprofundamento, assegurando que o artefato atenda aos requisitos identificados e esteja pronto para a próxima fase.

Por fim, a **quarta etapa** corresponde à **Experimentação**, na qual o artefato é submetido a testes e avaliações em contextos simulados e reais. Nessa fase, colhem-se dados empíricos para verificar sua eficácia, identificar possíveis limitações e propor ajustes, concluindo o ciclo de desenvolvimento e fornecendo evidências sobre a contribuição efetiva do artefato para a pesquisa em PC.

Nas próximas seções, cada etapa será descrita em detalhes, com ênfase nos procedimentos metodológicos empregados e na forma como se integram ao processo de desenvolvimento da pesquisa.

4.2.1 Exploração

Esta etapa da pesquisa, de caráter essencialmente exploratório, foi desenvolvida ao longo de 2020, com o objetivo principal de aprofundar a compreensão do PC e identificar desafios e lacunas emergentes na literatura acadêmica. Para isso, foram realizadas duas atividades-chave: (i) um MSL, voltado para a investigação de abordagens e metodologias empregadas no ensino de PC; e (ii) uma pesquisa-ação, focada na aplicação prática e na análise do ensino de PC por meio de programação em blocos.

A realização e combinação dessas duas atividades possibilitou a construção de um panorama abrangente e consolidado do estado da arte sobre o tema, além de revelar problemáticas que impactam diretamente o desenvolvimento e a implementação do ensino de PC. Em especial, os achados dessa fase evidenciaram, entre outros aspectos, que a avaliação do PC se apresenta como um dos desafios mais significativos no campo da educação, ressaltando a necessidade de investigações mais aprofundadas sobre estratégias e ferramentas que possam apoiar esse processo avaliativo. Esses resultados forneceram subsídios teóricos e práticos para as etapas subsequentes desta investigação.

1. **Pesquisa-Ação:** Nesta vertente, propôs-se uma abordagem de ensino de PC que mescla *Computational Action* (TISSENBAUM *et al.*, 2019) e *Design Thinking* (AMBROSE; HARRIS, 2016), aplicada em uma disciplina para ingressantes em Ciência da Computação. Por meio de programação em blocos, os estudantes desenvolveram aplicativos para *smartphones*. A análise dos resultados demonstrou um aumento significativo tanto no conhecimento sobre PC quanto na motivação dos participantes. Esse estudo foi relatado em detalhes no artigo *Pensamento Computacional e a Ação Computacional por Ensino Remoto: Um relato de experiência de uso do AppInventor em meio a pandemia de COVID-19* (FARIAS *et al.*, 2020), publicado no Simpósio Brasileiro de Informática na Educação (Simpósio Brasileiro de Informática na Educação (SBIE)) de 2020, na trilha de Pensamento Computacional. Uma versão resumida deste artigo encontra-se no Apêndice A.
2. **Mapeamento Sistemático da Literatura (MSL):** Com o intuito de obter uma visão holística da área e identificar oportunidades de pesquisa, realizou-se um mapeamento sistemático voltado ao uso de programação em blocos no ensino de PC. Esse levantamento forneceu informações relevantes sobre os níveis de ensino mais contemplados, as ferramentas mais empregadas, as estratégias de avaliação utilizadas e a concentração geográfica

das pesquisas. Os resultados foram publicados no artigo *Programação em Blocos Aplicada no Ensino do Pensamento Computacional: Um Mapeamento Sistemático* (SOUSA *et al.*, 2020), também apresentado no SBIE de 2020, na trilha de Pensamento Computacional. O artigo está disponível no Apêndice B.

Em síntese, as reflexões e evidências obtidas nesta etapa estabelecem os alicerces para as fases subsequentes do trabalho, permitindo direcionar os esforços de pesquisa para a proposição de soluções que respondam, de forma fundamentada, aos principais desafios inerentes ao ensino e à avaliação do Pensamento Computacional

4.2.2 Aprofundamento

Esta etapa da pesquisa foi conduzida entre os anos de 2022 e 2023, com o objetivo central de identificar desafios e oportunidades de pesquisa relacionados à avaliação da aprendizagem do PC. Para alcançar esse propósito, adotou-se uma abordagem de métodos mistos, combinando survey e análise bibliográfica, possibilitando uma investigação abrangente e fundamentada sobre o tema.

Os resultados desta fase evidenciaram que a avaliação do PC ainda representa um desafio significativo no contexto educacional, especialmente no ensino médio, onde professores relatam uma carência substancial de ferramentas apropriadas para suporte avaliativo. Além disso, foi possível mapear as principais linguagens de programação utilizadas no ensino de PC, fornecendo insumos para compreender quais delas são mais adotadas e suas implicações no processo de ensino-aprendizagem.

Outro achado relevante desta investigação foi a necessidade de uma ferramenta específica para avaliar os pilares do Pensamento Computacional, tais como decomposição, reconhecimento de padrões, abstração e algoritmos. Essa lacuna aponta para a importância do desenvolvimento de um artefato tecnológico capaz de automatizar e sistematizar a avaliação do PC, contribuindo para a superação dos desafios encontrados e proporcionando maior suporte aos docentes no acompanhamento do progresso dos alunos.

As atividades realizadas nessa etapa incluem:

1. Reanálise dos trabalhos selecionados no MSL descrito na subseção 4.2.1, com foco desta vez na avaliação da aprendizagem do PC;
2. Aplicação de um *survey* com o questionário sendo submetido aos autores dos artigos que foram investigados em (SOUSA *et al.*, 2020).

3. Análise da bibliográfica adicional de trabalhos que utilizavam aplicações específicas para realizar a avaliação da aprendizagem do pensamento computacional;
4. Análise de funcionamento das principais ferramentas de avaliação do pensamento computacional. A escolha das ferramentas a serem analisadas foi feita com base na frequência que cada uma delas foi citada no mapeamento, *survey* e análise bibliográfica;

Como resultado dessa etapa, tivemos:

1. **Mapeamento Sistemático da Literatura:** com o objetivo identificar e apresentar um mapeamento sistemático da literatura sobre estudos que relatam a adoção do Pensamento Computacional no contexto escolar. Foram analisadas as principais características dos estudos como, localidade, níveis de escolaridade, teorias pedagógicas, metodologias e ferramentas de aprendizagem. Os resultados deste estudo foram divulgados no artigo intitulado Análise da Adoção de Pensamento Computacional no Contexto Escolar Brasileiro: Um Mapeamento Sistemático da Literatura (FARIAS *et al.*, 2023), publicado nos anais do Simpósio Brasileiro de Informática na Educação (SBIE) de 2023, na trilha temática de Pensamento Computacional. O referido artigo está disponível no Apêndice C desta tese.
2. **MSL e Survey:** Esta etapa consistiu na realização de um MSL, complementado pela aplicação de uma survey direcionada a pesquisadores que desenvolvem estudos na área. A análise abrangeu 81 artigos selecionados no MSL, cujos resultados são apresentados em detalhes no Capítulo 5.

4.2.3 Desenvolvimento

Seguindo os princípios metodológicos do *Design Science Research* (DSR) (DRESCH *et al.*, 2020), esta etapa do trabalho encontra-se organizada em três fases principais, cada qual direcionada a aspectos específicos da concepção e evolução do artefato proposto. A adoção do DSR permite uma abordagem iterativa e rigorosa, pois integra tanto a produção de conhecimento teórico quanto a geração de soluções práticas para problemas reais (HEVNER *et al.*, 2004).

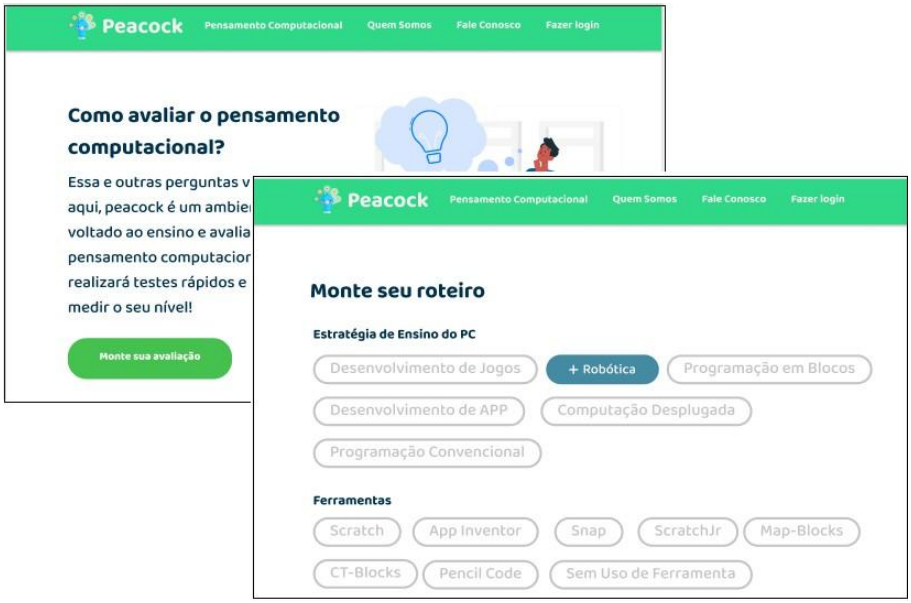
1. **Projeto do Artefato:** Nesta primeira fase, concentra-se a definição inicial do escopo, requisitos e funcionalidades desejáveis, tomando como base os achados das fases anteriores da pesquisa. A elaboração de um plano conceitual sólido — que inclua objetivos claros, elementos necessários para sua concretização e restrições impostas pelo contexto — é fundamental para orientar o desenvolvimento subsequente.
2. **Desenvolvimento do Artefato:** Com o planejamento delineado, passa-se à implementação

das funcionalidades previstas. A escolha de tecnologias, frameworks e abordagens de desenvolvimento deve alinhar-se aos objetivos traçados, buscando garantir a coerência entre as necessidades mapeadas e as soluções adotadas. Este momento também envolve constantes iterações e refinamentos, já que cada melhoria ou descoberta pode demandar ajustes no projeto inicial.

3. **Revisão do Artefato:** Após a implementação inicial, procede-se a uma avaliação preliminar junto a especialistas e potenciais usuários, visando identificar eventuais problemas de usabilidade, lacunas funcionais e oportunidades de aperfeiçoamento. Os insights advindos desta avaliação subsidiarão as próximas fases de refinamento, de forma a garantir que o artefato evolua em sintonia com as demandas práticas e teóricas da pesquisa.

É relevante destacar que, em 2023, quando o escopo do software encontrava-se ainda mais focado na elaboração de um roteiro para auxiliar professores na avaliação do PC, foi desenvolvido um protótipo de alta fidelidade (Figura 7). Apesar disso, optou-se por não seguir com sua implementação efetiva, pois as investigações realizadas na Etapa 3 desta tese evidenciaram a necessidade de um artefato mais direcionado e conciso. Dessa forma, os resultados obtidos serviram como insumos críticos para reorientar o projeto e assegurar que o desenvolvimento futuro atendesse com maior precisão às lacunas identificadas ao longo do processo, reforçando o caráter iterativo e reflexivo que caracteriza a abordagem DSR.

Figura 7 – Protótipo da ferramenta de avaliação do pc



Fonte: elaborada pelo autor (2025).

O desenvolvimento do artefato final ocorreu em 2024, tendo como propósito principal

apoiar professores, especialmente do ensino médio, na avaliação da aprendizagem de PC. Nesse processo, o artefato foi concebido para realizar a correção semi-automatizada de códigos em Python, baseando-se em desafios previamente propostos pelos docentes, o que torna o processo avaliativo mais eficiente, consistente e alinhado às demandas pedagógicas contemporâneas. Ao fornecer feedback imediato e mensurável, espera-se que esse mecanismo promova maior engajamento dos estudantes e o desenvolvimento das habilidades que formam o PC.

As especificações técnicas, funcionalidades detalhadas e os diversos aspectos inerentes ao projeto e ao desenvolvimento do artefato são discutidos em profundidade no Capítulo 7. É nessa seção que se apresentam, por exemplo, as escolhas de arquitetura de software, as estratégias de implementação adotadas e as decisões de design tomadas ao longo do processo, assegurando a coerência entre os objetivos educacionais e a proposta metodológica.

Em consonância com os fundamentos do *Design Science Research* (DSR), a avaliação inicial do artefato foi planejada de maneira experimental (DRESCH *et al.*, 2020). Conforme mencionado previamente, a avaliação final será conduzida por meio de um experimento controlado em sala de aula, permitindo analisar o impacto do artefato em um ambiente real de ensino e aprendizagem. Entretanto, antes dessa aplicação em contexto autêntico, o artefato passou por uma fase de revisão, na qual um painel de especialistas em PC avaliou sua adequação quanto às funcionalidades previstas e à capacidade de atender às demandas identificadas na etapa de levantamento de requisitos.

O principal objetivo dessa revisão foi verificar se as funcionalidades implementadas seriam, de fato, capazes de apoiar o trabalho docente, certificando-se de que o artefato estivesse alinhado a princípios pedagógicos e às expectativas dos professores. A operacionalização dessa etapa e os resultados obtidos encontram-se descritos detalhadamente na Capítulo 8, que apresenta as análises realizadas pelos especialistas, as contribuições recebidas e as melhorias implementadas no artefato. Dessa forma, esse processo de revisão reforça o rigor metodológico do DSR, assegurando que o artefato seja aprimorado com base em evidências concretas e *feedback* fundamentado, antes de sua aplicação em um contexto educacional real.

4.2.4 Experimentação

Conforme apresentado na Seção 4.2.3, a avaliação do artefato fundamenta-se nos princípios do DSR (DRESCH *et al.*, 2020; PIMENTEL *et al.*, 2019), abordagem que concebe o desenvolvimento de artefatos como uma atividade científica orientada à criação de soluções

inovadoras para problemas bem definidos. Ao adotar o DSR, busca-se avaliar a eficácia do artefato de maneira sistemática, testando hipóteses específicas e promovendo seu aperfeiçoamento contínuo com base nos resultados obtidos.

Além disso, o DSR contribui para assegurar que o artefato seja funcional e adequado ao contexto educacional. Espera-se, assim, que a abordagem ofereça uma solução eficaz e acessível para apoiar professores do Ensino Médio na avaliação do PC, ampliando a efetividade do desenvolvimento das habilidades que compõem o PC entre estudantes desse nível de ensino.

Concluídos o desenvolvimento e a revisão do artefato de software, a etapa de experimentação foi estruturada em três ações principais:

1. Comparação das respostas fornecidas por diferentes modelos de LLMs;
2. Comparação entre as avaliações realizadas por um especialista humano e aquelas geradas pelo artefato;
3. Realização de um experimento controlado em uma turma de Ensino Médio.

4.2.4.1 Comparação entre Modelos de LLMs

Nesta fase, foi conduzida uma análise comparativa entre as respostas fornecidas por três *Large Language Models* (LLMs) disponíveis no mercado: ChatGPT, Google Gemini e deepSeek. O objetivo consistiu em avaliar o desempenho desses modelos no contexto do artefato proposto, identificando possíveis lacunas e oportunidades de aprimoramento.

O critério utilizado para avaliação das LLMs foi o desempenho e a eficiência, mensurados pelo tempo médio de resposta (em segundos) para cada código submetido. O procedimento foi realizado da seguinte forma:

- Foi utilizado um cronômetro do navegador para mensurar o tempo de resposta;
- Os códigos e desafios foram categorizados em níveis de complexidade (simples, intermediário e complexo);
- A contagem do tempo se iniciava no momento do envio da requisição ao modelo e era interrompida quando a resposta estava completamente exibida no artefato;
- O tempo gasto foi registrado em uma planilha para cada código submetido;
- Ao final, calculou-se a média dos tempos coletados para cada modelo.

Durante os testes, não foram observadas instabilidades significativas, como *timeouts*, quedas de conexão ou falhas técnicas. Os experimentos foram realizados tanto em horários de pico quanto em períodos noturnos, não havendo variações expressivas no desempenho.

4.2.4.2 Comparação com Avaliação de Especialistas

Foi conduzido um estudo aplicado em uma turma do Ensino Superior, no qual um professor especialista avaliou manualmente o PC, embasando-se em uma adaptação do artefato proposto por (OLIVEIRA; PEREIRA, 2023). O objetivo foi comparar as avaliações humanas às geradas pelo artefato, a fim de verificar se este apresenta níveis de precisão e confiabilidade compatíveis com a análise especializada.

Destaca-se que, no artefato utilizado pelo professor, as habilidades do Pensamento Computacional estão subdivididas até o nível de evidências. Dessa forma, a comparação entre a avaliação do professor e a do artefato foi realizada com base na média das notas atribuídas aos indícios que compõem cada evidência específica.

O professor especialista realizou apenas a avaliação quantitativa dos códigos fornecidos. Assim, foram comparados exclusivamente os conceitos (notas) atribuídos pelo especialista com aqueles fornecidos pelo artefato, considerando os modelos de LLM ChatGPT, Google Gemini e deepSeek.

O processo foi conduzido da seguinte forma:

- Os mesmos códigos Python, previamente avaliados pelo professor especialista, foram submetidos a cada uma das LLM;
- Foram registradas as notas atribuídas por cada modelo às evidências analisadas;
- Foi calculado o coeficiente de concordância entre as notas humanas e as geradas pelo artefato (acurácia simples).

Adicionalmente, foi verificado se os modelos foram capazes de retornar adequadamente a indicação de "não se aplica" nos casos em que determinada evidência estava ausente, conforme identificado pelo especialista.

4.2.4.3 Experimento Controlado

Por fim, foi realizado um experimento controlado em uma turma do Ensino Médio da Escola Estadual de Educação Profissional Monsenhor José Aloysio Pinto, localizada no estado do Ceará. O objetivo principal do estudo foi avaliar, sob perspectiva científica, a capacidade do artefato em fornecer suporte ao professor do Ensino Médio na avaliação do Pensamento Computacional.

Trata-se de um estudo de caso com abordagem quali-quantitativa, conduzido com 23

alunos, devidamente submetido e aprovado pelo Comitê de Ética em Pesquisa da Universidade Estadual Vale do Acaraú (UVA). A Carta de Anuência da direção da escola e o parecer do CEP encontram-se no Anexo A.

Durante o experimento, os alunos foram convidados a resolver desafios de programação em linguagem Python, no ambiente escolar regular, sob supervisão do professor responsável. As soluções desenvolvidas foram analisadas pelo artefato AutoEval-CT.

A participação dos estudantes foi voluntária, condicionada à assinatura do Termo de Consentimento Livre e Esclarecido (TCLE) pelos responsáveis legais, uma vez que se tratava de participantes menores de idade. O estudo respeitou integralmente os princípios éticos aplicáveis à pesquisa com seres humanos, garantindo anonimato, confidencialidade e uso exclusivo dos dados para fins científicos, em conformidade com a Lei Geral de Proteção de Dados (LGPD).

A coleta de dados ocorreu por meio de dois procedimentos complementares:

1. **Exportação e análise dos registros gerados pelo artefato de avaliação do Pensamento Computacional:** Inclui métricas de desempenho, indicadores de processo e demais evidências produzidas automaticamente pela ferramenta;
2. **Entrevista semiestruturada com o(a) professor(a) da disciplina:** Aplicada após o período de uso do artefato, utilizando roteiro previamente validado para explorar aspectos pedagógicos, técnicos e éticos envolvidos. O roteiro da entrevista pode está disponível no Apêndice D.

Ao término da entrevista, todas as falas foram transcritas e submetidas à técnica de Análise de Conteúdo proposta por (BARDIN, 2016), visando extrair categorias e padrões relevantes que subsidiem as conclusões deste estudo.

4.3 Considerações Finais

Este capítulo apresentou os procedimentos metodológicos adotados nesta tese. A combinação de diferentes abordagens — *Design Science Research*, Mapeamentos Sistemáticos da Literatura, Pesquisa-Ação, Experimento Controlado e *Survey* — garantiu a integração entre rigor científico e aplicabilidade prática, permitindo que o artefato fosse desenvolvido de forma iterativa, fundamentada e orientada às necessidades reais do contexto educacional.

O arranjo metodológico aqui descrito conferiu consistência e legitimidade ao processo investigativo, assegurando que os resultados desta tese estejam alicerçados em evidências empíricas e respaldo teórico.

5 PERSPECTIVAS, DESAFIOS E OPORTUNIDADES NA AVALIAÇÃO DO PENSAMENTO COMPUTACIONAL

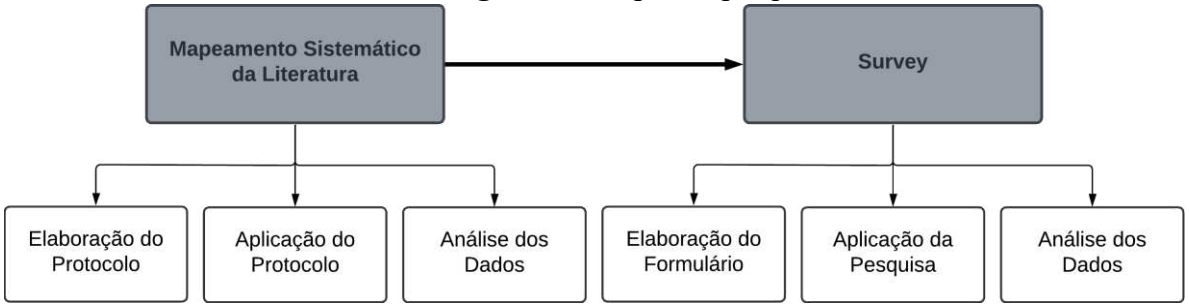
Este capítulo tem como objetivo apresentar uma análise abrangente do estado da arte sobre a avaliação do PC, destacando os principais conhecimentos, estratégias, dificuldades e oportunidades identificadas na literatura. Para isso, foi conduzido um MSL, complementado pela aplicação de uma survey junto a pesquisadores que desenvolvem estudos na área. A análise dos 81 artigos selecionados no MSL, associada às respostas de 41 especialistas participantes da pesquisa, permitiu identificar lacunas relevantes, entre elas a carência de instrumentos padronizados e eficientes para apoiar os professores no processo de avaliação do PC. Os achados reforçam a necessidade de avançar na construção de metodologias e ferramentas que tornem esse processo mais preciso, objetivo e alinhado às demandas educacionais contemporâneas.

5.1 Metodologia

Para esta etapa do trabalho, o problema de pesquisa consiste em caracterizar o estado da arte da avaliação do PC e identificar suas principais características, estratégias, dificuldades, desafios e soluções.

Dividimos nossa investigação em duas fases para responder a essa questão central, conforme ilustrado na Figura 8. Na primeira fase, realizamos um MSL com foco específico na avaliação de PC. Na segunda fase — visando compreender com mais profundidade os desafios e oportunidades do tema — conduzimos uma *survey* com os pesquisadores cujos trabalhos compuseram o MSL.

Figura 8 – Etapas da pesquisa



Fonte: elaborada pelo autor (2025).

Questões de pesquisa:

RQ1. Quais são as principais dificuldades, características e aspectos apontados pelos

autores na avaliação de PC?

RQ2. Quais ferramentas são atualmente utilizadas para avaliar PC?

RQ3. Quais estratégias e instrumentos são empregados na avaliação de PC?

5.1.1 Mapeamento Sistemático da Literatura

Esta etapa do trabalho segue as diretrizes propostas por (KITCHENHAM *et al.*, 2007). Utilizou-se a ferramenta StArt (State of the Art through Systematic Review), desenvolvida na Universidade Federal de São Carlos (UFSCar), para apoiar o mapeamento.

5.1.1.1 Protocolo de Revisão

Um protocolo de revisão é um conjunto de instruções que descreve as etapas necessárias para o mapeamento, reduzindo possíveis vieses dos pesquisadores (KITCHENHAM *et al.*, 2007).

Empregou-se uma busca automatizada nas seguintes bibliotecas digitais: *Institute of Electrical and Electronics Engineers* (IEEE), *Association for Computing Machinery* (ACM) e Science Direct. Aceitaram-se apenas artigos em inglês publicados entre 2017 e 2022.

Seguiu-se o protocolo PICOC sugerido por (KITCHENHAM *et al.*, 2007) para criar a string de busca, dividindo os termos em cinco blocos: População, Intervenção, Comparação, Resultado e Contexto. A string gerada foi: **(“*Computational thinking*”) *AND* (“*Assessment*” *OR* “*Test*” *OR* “*Evaluations*” *OR* “*Examinations*”)**.

Critérios de Inclusão

1. Estudos que contenham as palavras-chave;
2. Estudos que apresentem informações relacionadas ao pensamento computacional e sua aplicação;
3. Estudos sobre avaliação de pensamento computacional;
4. Estudos em inglês;
5. Estudos publicados entre 2017 e 2022.

Critérios de Exclusão

1. Publicações fora do período ou idioma estabelecidos;
2. Trabalhos sem versão completa disponível;
3. Trabalhos que não tratem de pensamento computacional;
4. Trabalhos que não tratem da avaliação de pensamento computacional.

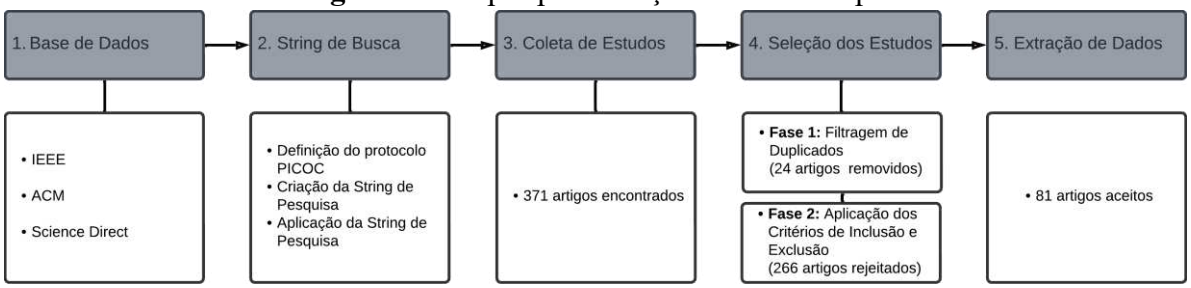
Formulário de Extração de Dados

1. Título do artigo;
2. Autores;
3. Ano de publicação;
4. País onde o estudo foi realizado;
5. Data de execução do estudo;
6. Metodologia de ensino;
7. Faixa etária;
8. Nível educacional;
9. Tipo de avaliação de PC;
10. Ferramentas de avaliação;
11. Instrumentos de avaliação.

5.1.1.2 Aplicação do Protocolo

Na 9, ilustra-se o processo de busca, seleção e extração de dados dos estudos. Esse processo é dividido em cinco fases: repositórios (bases de dados), string de busca, coleta de estudos, seleção de estudos e extração de dados.

Figura 9 – Etapas para seleção dos estudos primários



Fonte: elaborada pelo autor (2025).

A Tabela 1 resume os resultados da busca automática. Recuperaram-se 371 estudos; a base IEEE concentrou 91,11 % deles, enquanto a ACM respondeu por apenas 2,7 %.

Após definir as bases de dados e aplicar a string de busca, os 338 estudos primários passaram por duas fases de seleção. Primeiramente, removeram-se 24 duplicatas, restando 347 trabalhos. Em seguida, aplicaram-se os critérios de seleção, resultando na aceitação de 81 estudos e rejeição de 266. Na quinta etapa, os dados necessários às perguntas de pesquisa foram extraídos.

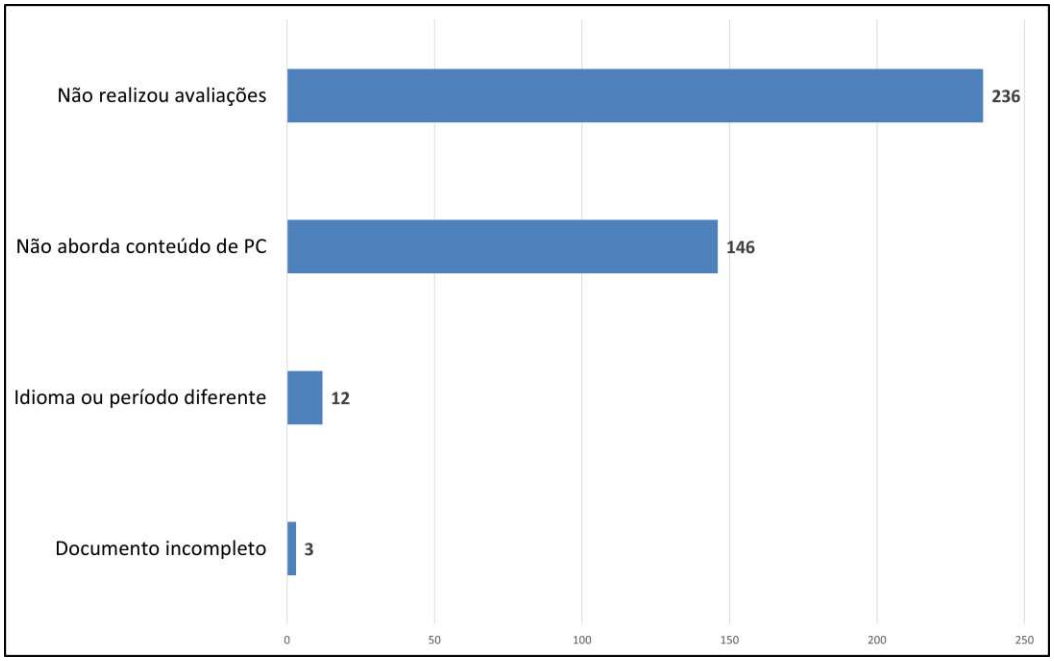
Tabela 1 – Busca automatizada

Repositórios	Encontrados	%	Duplicados	Rejeitados	Aceitos	% Aceitos
IEEE	338	91,11%	2	260	76	93,83%
Science Direct	23	6,20%	22	1	0	0%
ACM	10	2,70%	0	5	5	6,17%
Total	**371**	**100%**	**24**	**266**	**81**	**100%**

Fonte: elaborada pelo autor (2025).

Avaliar PC é uma tarefa não trivial, como mostra a 10. Dos 266 artigos rejeitados, 236 (88,72 %) não realizaram nenhuma avaliação de PC.

Figura 10 – Critérios de exclusão aplicados



Fonte: elaborada pelo autor (2025).

Os títulos dos 81 trabalhos selecionados podem ser encontrados no Apêndice E

5.1.2 Pesquisa por Survey

5.1.2.1 Escopo

Desenvolvemos o survey desta pesquisa de acordo com as orientações de (KITCHENHAM *et al.*, 2015). Pesquisas de opinião com especialistas podem cumprir diversos propósitos, como auxiliar na identificação de problemas, prever mudanças e esclarecer questões relevantes sobre um tema específico (ROWE; WRIGHT, 2001).

Propusemos realizar um survey on-line com todos os autores dos 81 trabalhos

selecionados no MSL, a fim de analisar a avaliação de PC de forma mais aprofundada. Dessa maneira, buscamos detectar desafios e oportunidades de pesquisa na área (ROWE; WRIGHT, 2001).

5.1.2.2 Instrumento

O questionário foi composto por três seções e 16 perguntas, incluindo cinco questões abertas. O conteúdo está dividido em: (a) Perfil do Pesquisador, (b) Pensamento Computacional e (c) Avaliação do Pensamento Computacional.

Utilizamos a ferramenta Google Forms® para apoiar a aplicação e o gerenciamento do survey.

5.1.2.3 Procedimento

No processo de busca por endereços de e-mail, foram identificados 300 autores. Verificou-se, contudo, que sete endereços não puderam ser localizados e 26 estavam duplicados, resultando em 267 destinatários finais convidados a participar. Dos 267 e-mails enviados, 41 autores responderam; 201 não responderam, 21 retornaram erro e quatro foram bloqueados.

A seleção dos participantes ocorreu por amostragem não probabilística por conveniência (WOHLIN *et al.*, 2012), pois escolhemos pesquisadores provenientes do MSL.

5.1.2.4 Validade da Amostra

Apresentamos uma visão geral dos resultados da pesquisa de opinião com os autores dos 81 artigos mapeados. Quarenta e um autores responderam ao questionário. Esse número é significativo e estatisticamente válido, já que a literatura indica ser adequado utilizar grupos de 5 a 20 especialistas (ROWE; WRIGHT, 2001). Além disso, a diversidade de experiências dos participantes e o fato de terem respondido de forma independente reduzem a possibilidade de viés nas respostas.

Como as perguntas eram opcionais, algumas delas receberam menos de 41 respostas na apresentação dos resultados.

5.1.2.5 Análise dos Dados

Para as questões de múltipla escolha, empregamos duas abordagens. Utilizamos estatística descritiva nas perguntas destinadas à coleta de dados e, nas que solicitavam opinião dos especialistas, adotamos uma escala de Likert de 1 a 5, em que 1 corresponde a "Mais fácil" e 5 a "Mais difícil" (LUNA, 2007).

Realizamos uma análise qualitativa das respostas às questões abertas e uma análise quantitativa das respostas às questões fechadas. Os resultados de ambas as análises foram combinados para revelar tópicos essenciais. A interpretação das cinco questões abertas utilizou a técnica de análise de conteúdo, que identifica, analisa e interpreta padrões de significado em dados qualitativos (BARDIN, 2010).

5.2 Resultados e Discussões

Nesta seção, apresentamos os principais resultados que emergiram a partir da tabulação dos dados do MSL, descrito na Seção 5.1.1, e dos respostas provenientes do survey detalhado na Seção 5.1.2.

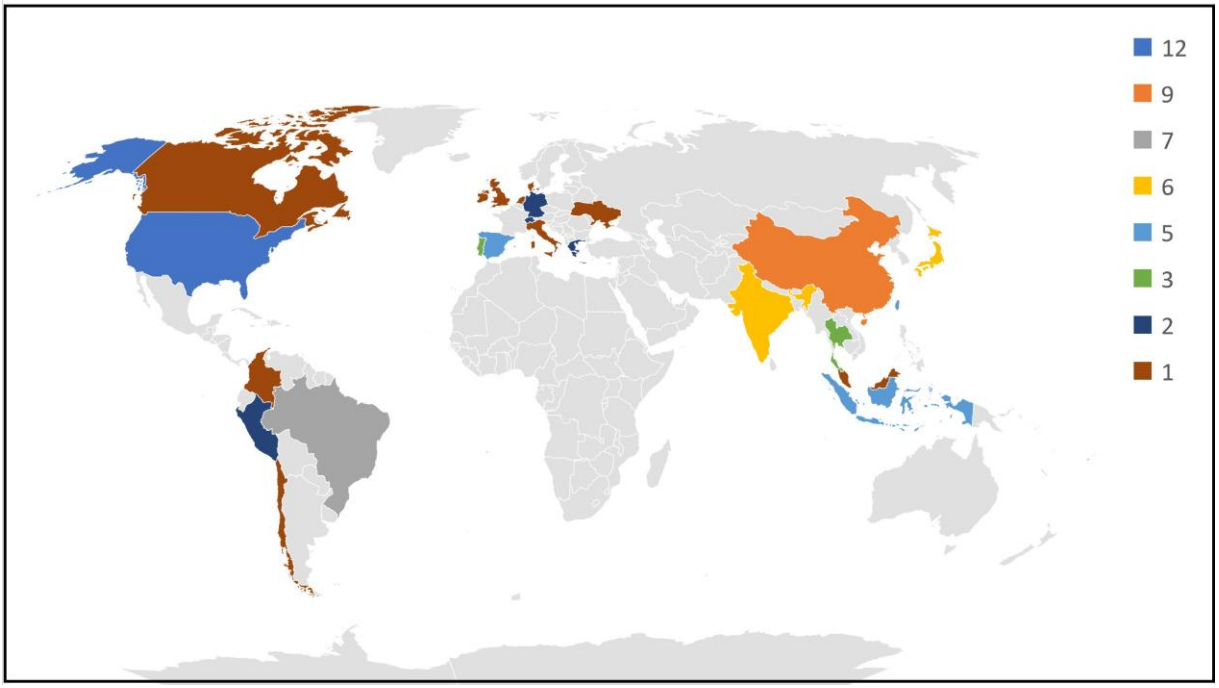
5.2.1 Distribuição Geográfica

Apensar de não ser uma das questões de pesquisa, mapear a origem dos trabalhos permite identificar polos de pesquisa mais ativos, revelar possíveis assimetrias entre continentes e apontar oportunidades de cooperação internacional. Além disso, compreender a diversidade de contextos socioeconômicos e educacionais nos quais os estudos foram conduzidos é fundamental para interpretar resultados, adaptar metodologias e fomentar iniciativas que ampliem o alcance e a representatividade das investigações em PC ao redor do mundo.

Dos **26 países** que contribuíram com publicações sobre avaliação de *PC* no *MSL*, observa-se a predominância dos Estados Unidos, responsáveis por 12 documentos (14,81 %), seguidos pela China, com 9 (11,11 %), e pelo Brasil, com 7 trabalhos (8,64 %). Índia e Japão produziram 6 artigos cada (7,41 %), enquanto Espanha, Indonésia e Taiwan registraram 5 estudos cada (6,17 %). Portugal e Tailândia apresentaram 3 pesquisas (3,70 %), ao passo que Alemanha, Grécia, Suíça e Peru contribuíram com 2 artigos cada (2,47 %). Outros 12 países somaram um único artigo (1,23 %). Em termos continentais, a produção distribui-se entre Ásia (36 artigos), Américas (24) e Europa (20), com a África representada apenas por um estudo realizado em

Cabo Verde. **Esses resultados evidenciam uma forte concentração de pesquisas em nações com maior investimento em inovação educacional, apontando uma lacuna significativa no cenário africano e indicando oportunidades para ampliar a diversidade geográfica dos estudos sobre avaliação de PC**. A Figura 11 resume a distribuição dos países mapeados.

Figura 11 – Países onde os estudos se concentram



Fonte: elaborada pelo autor (2025).

Nas 5.2.2, 5.2.3 e 5.2.4, apresentamos as respostas às três questões de pesquisa deste trabalho, fundamentadas nos dados extraídos do MSL e nas respostas coletadas no *survey*.

5.2.2 Dificuldades, Características e Funcionalidades da Avaliação de PC

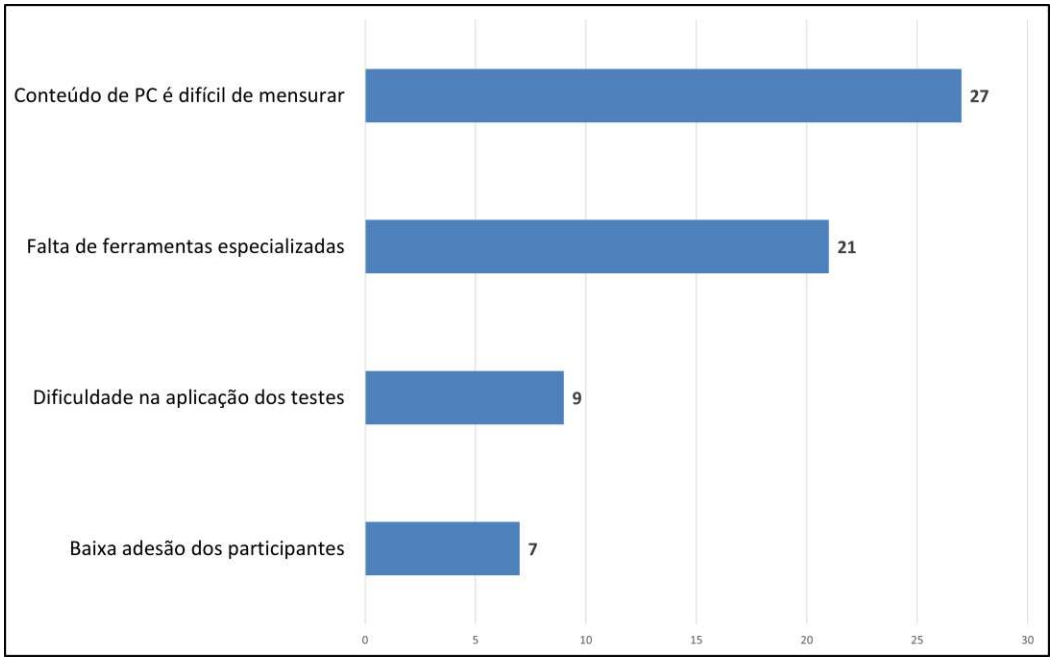
Esta seção apresenta os resultados que visam responder à primeira questão de pesquisa: "Quais são as principais dificuldades, características e funcionalidades indicadas pelos autores na avaliação do Pensamento Computacional?".

No tocante às dificuldades, a questão do *survey* "Quais as principais dificuldades na avaliação de PC?"[1] recebeu 39 respostas. A análise de conteúdo agrupou-as em quatro categorias, totalizando 64 citações. O obstáculo mais recorrente foi **Conteúdo de PC difícil de mensurar** (27 citações), seguido por *falta de ferramentas especializadas* (21), *dificuldades na aplicação dos testes* (9) e *baixa adesão dos participantes* (7). Conforme sintetiza a Figura 12, os maiores

[1] Pergunta aberta; cada respondente pôde indicar mais de uma dificuldade.

entraves concentram-se em desafios metodológicos e instrumentais, indicando que o avanço na definição de métricas e no desenvolvimento de ferramentas deve preceder esforços para ampliar a escala e a participação em avaliações de PC.

Figura 12 – Principais dificuldades na avaliação de pc



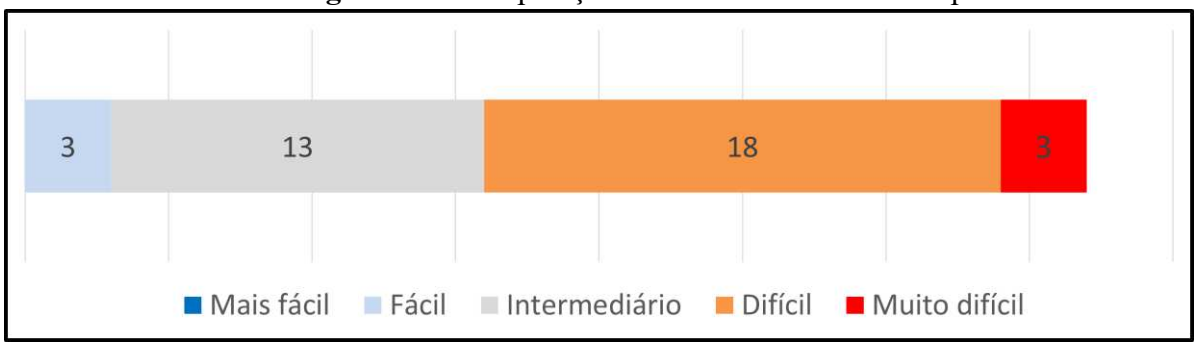
Fonte: elaborada pelo autor (2025).

Ao serem questionados — "Você utiliza alguma ferramenta de avaliação de *PC*? Quais dificuldades encontrou?", os especialistas apontaram cinco problemas recorrentes: (i) tarefas excessivamente genéricas; (ii) dificuldade em calibrar o nível de complexidade; (iii) inadequação para estudantes sem experiência prévia em programação; (iv) aplicação pouco viável em turmas numerosas; e (v) limitada adequação ao estágio de desenvolvimento infantil.

A Figura 13 compila as respostas à pergunta *"Comparada à avaliação de outros conceitos de computação, qual é a dificuldade em avaliar PC?"*. Das 37 respostas recebidas, nenhum pesquisador classificou a avaliação de PC como *"Mais fácil"*. Apenas três indicaram ser *"Fácil"* (8,11 %), 13 apontaram dificuldade *"Intermediária"* (35,14 %) e 21 a julgaram *"Difícil"* ou *"Muito difícil"* (56,76 %). Esses resultados evidenciam que a avaliação de PC é percebida como mais complexa que outros conceitos de computação e reforçam que a avaliação de PC permanece um desafio substancial para a comunidade de pesquisa.

A Figura 14 evidencia que as metodologias didáticas mais investigadas concentram-se em *gamificação* (19 artigos mapeados), *programação em blocos* (17 artigos), linguagens de programação tradicionais (14 artgos) e *robótica educacional* (13 artigos). Abordagens emergentes

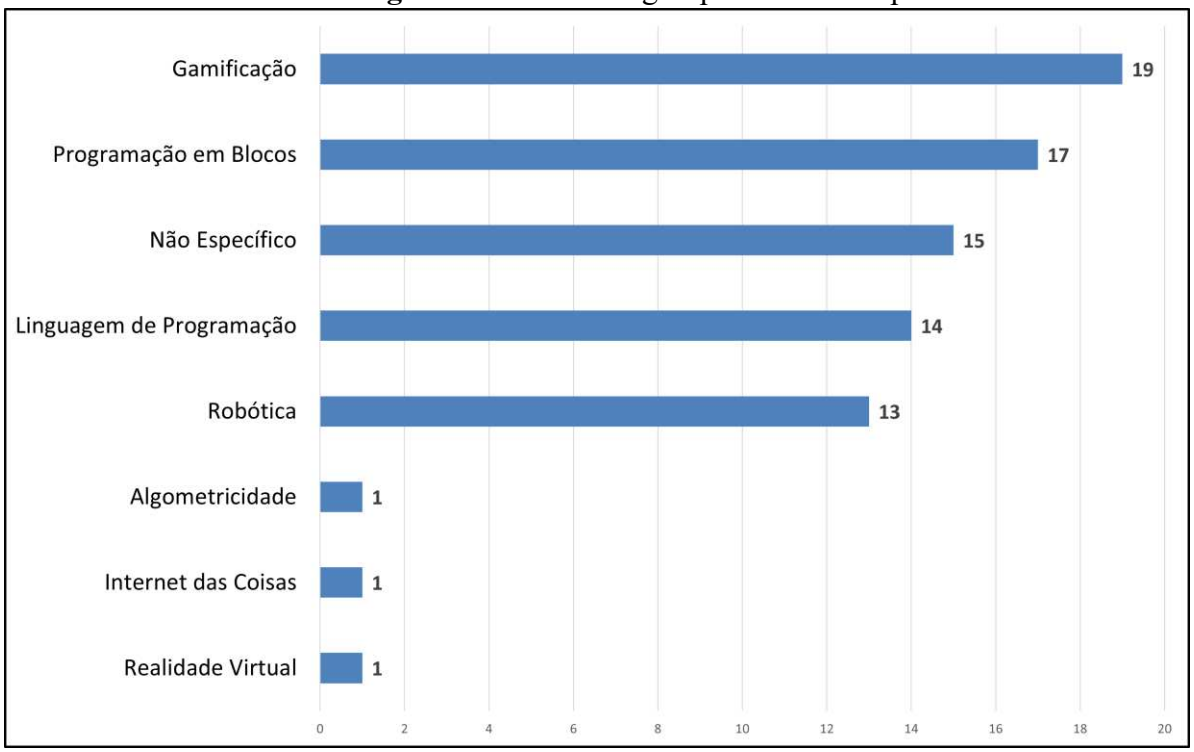
Figura 13 – Comparação da dificuldade em avaliar pc



Fonte: elaborada pelo autor (2025).

— como Letramento Algorítmico, Realidade Virtual e Internet das Coisas — surgem de modo pontual, cada qual em um único estudo, indicando estágio inicial de exploração. Além disso, quinze trabalhos não especificam a metodologia utilizada, o que ressalta uma lacuna de transparência que dificulta a replicação e a comparação sistemática entre pesquisas. Coletivamente, esses dados sugerem a preferência por estratégias práticas e visuais, ao mesmo tempo em que apontam oportunidades para diversificar e detalhar as práticas avaliadas na literatura sobre PC.

Figura 14 – Metodologias para ensino de pc



Fonte: elaborada pelo autor (2025).

Na questão do *survey* "Quais características seriam importantes para uma ferramenta

de avaliação de PC?", obtiveram-se 30 respostas. A análise de conteúdo revelou 15 categorias distribuídas em 40 citações, visto que cada resposta podia contemplar mais de uma categoria. A característica mais frequentemente mencionada foi **avaliar múltiplas dimensões do PC** (9 citações). Em seguida, sobressaíram-se *validade e confiabilidade* e *adequação ao público-alvo*, ambas com 6 citações. A Figura 15 sintetiza esses achados, indicando que os respondentes valorizam, sobretudo, abordagens abrangentes e metodologicamente sólidas, capazes de atender a diferentes perfis de aprendizes.

Figura 15 – Características desejáveis em uma ferramenta de avaliação de pc

Fonte: elaborada pelo autor (2025).

A complexidade intrínseca do conteúdo, a ausência de padronização e a carência de instrumentos apropriados, se apresentam como fatores que dificultam o processo de avaliação do PC. Tais limitações evidenciam a natureza multifacetada do PC e o déficit de ferramentas específicas, apontando para a necessidade de estudos adicionais que tragam confirmações estatísticas. Como resposta a esse cenário, propõe-se o desenvolvimento de uma ferramenta composta por desafios específicos, de aplicação simples, dotada de mecanismos de calibração automática de dificuldade, capaz de abarcar as múltiplas dimensões do PC, proporcionar avaliações formativas e de operar eficientemente em turmas numerosas. A criação de instrumentos válidos, confiáveis e adaptados a diferentes perfis de aprendizes emerge, portanto, como prioridade estratégica para

impulsionar de forma sinérgica tanto a pesquisa quanto a prática educacional.

5.2.3 Ferramentas de Avaliação PC

Esta seção apresenta os resultados que buscam esclarecer a segunda questão de pesquisa: "Quais ferramentas são atualmente empregadas para avaliar o Pensamento Computacional?".

A Tabela 2 sintetiza as ferramentas identificadas nos estudos analisados. *Questionários* foram empregados em 30 artigos, sinalizando preferência por instrumentos genéricos. *Pré- e pós-testes* apareceram em 13 estudos; *Scratch*, em 9; *Python*, em 8; *Dr. Scratch*, em 6; e *Bebras Tasks*, em 4. Já *Code.org*, *BCTt* e *Java* surgiram em 2 trabalhos cada. Treze artigos não mencionaram o uso de ferramenta específica, ao passo que outras 37 ferramentas ocorreram de forma isolada, demonstrando elevada diversidade de abordagens metodológicas.

Tabela 2 – Ferramentas de avaliação de pc

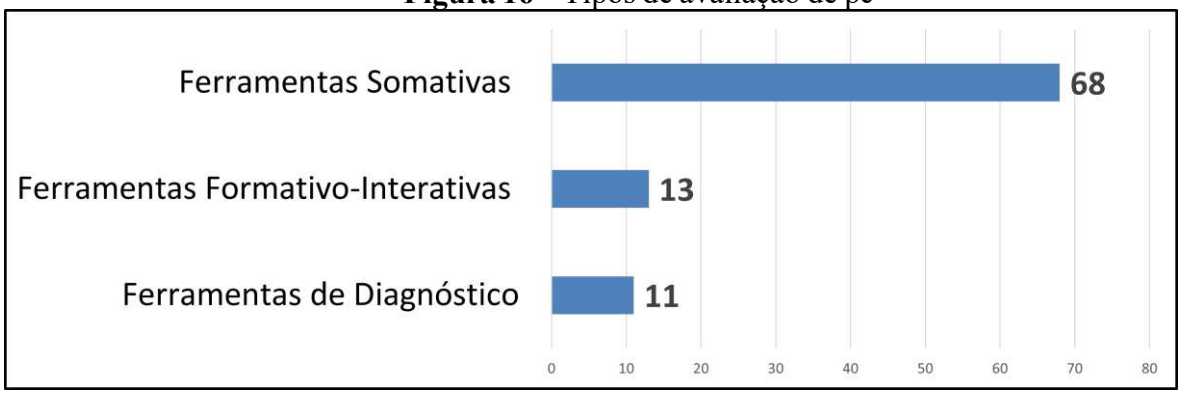
Ferramenta	**Quantidade**
Questionários	30
Nenhuma ferramenta, Pré- e Pós-teste, Scratch	13
Python	8
Dr. Scratch	6
Bebras Tasks	4
Code.org, Beginners Computational Thinking Test (BCTt), Java	2
Rubric TEC, LEGO Music Notation, GALT test, Shapiro–Wilk, VEnvI, LEGO Robotics EV-3, Método de Henrich, UNiREQ (jogo), Edu Coder, Blue Ant Code (BAC), Pepper (robô), TechCheck-k, FORESITY, Maker Uno, Robot Builder, JUSThink, Automatic Formative Assessment (AFA), Block-Py, Snap4Arduino, App Inventor, Pic2Program, Maze, Canvas, mBlock 5, Scratch 3.0-Based, DEEPSTEALTH (game), HERA, Design Scenario, Robot ROOT, Bee-Bot (robô), Blockly, Latent Dirichlet Allocation, Arducation Bot, Alg-Design, Abstractly, Robô BRICKO, Alert e RoboPad	1

Fonte: elaborada pelo autor (2025).

A categorização das ferramentas de avaliação do PC apresentadas nas Figuras 16 e 17, foram feitas com base nos estudos de (ROMÁN-GONZÁLEZ *et al.*, 2019).

Ao examinar os tipos de ferramentas de avaliação empregadas no MSL, observa-se um claro predomínio de procedimentos focados nos resultados finais de aprendizagem. A Figura 16 revela uma clara predominância da avaliação *somativa*, presente em 68 estudos (83,95 %), enquanto as modalidades *formativa* e *diagnóstica* aparecem em 13 (16,04 %) e 11 (13,58 %)

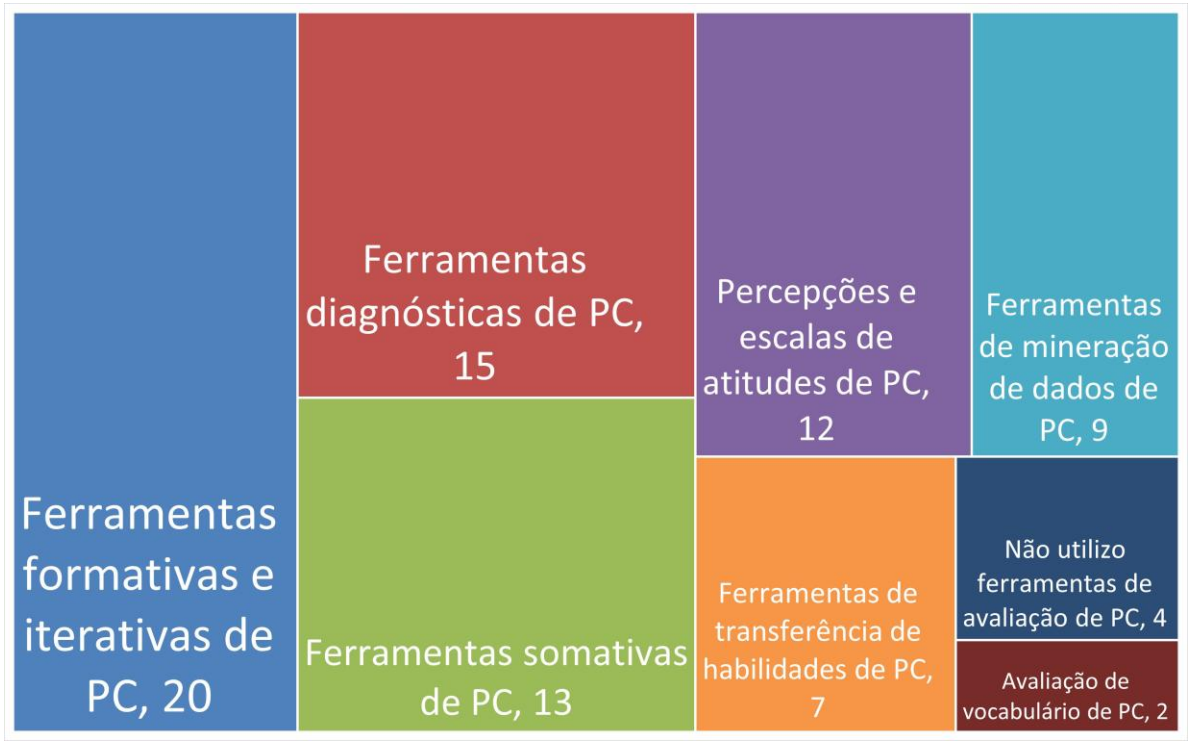
Figura 16 – Tipos de avaliação de pc



Fonte: elaborada pelo autor (2025).

investigações, respectivamente[2]. Esse padrão reforça a ênfase do campo em verificações finais de aprendizagem, indicando espaço para ampliar abordagens que acompanhem o processo e identifiquem lacunas antes da conclusão das atividades.

Figura 17 – Categorias das ferramentas empregadas



Fonte: elaborada pelo autor (2025).

Para validar esses achados, o *survey* incluiu a questão "A(s) ferramenta(s) que você usa pode(m) ser caracterizada(s) como:". Entre 37 respondentes, registraram-se 82 citações, já que cada participante pôde selecionar mais de uma categoria. A Figura 17 apresenta a

[2] Os percentuais ultrapassam 100 % porque alguns estudos empregaram mais de uma categoria de ferramenta.

categorização das ferramentas empregadas. Predominam as abordagens *formativo-interativas* (20 respostas - 54,05 %), seguidas pelas modalidades *diagnóstica* (15 - 40,54 %) e *somativa* (13 - 35,14 %), sugerindo preferência por instrumentos que acompanham o processo de aprendizagem sem descartar a relevância das avaliações finais. A disparidade entre os achados da literatura — que privilegia *ferramentas somativas* (Figura 16) — e a prática declarada pelos especialistas — que recorre majoritariamente a abordagens *formativo-interativas* — sinaliza uma evolução nas estratégias de avaliação de *PC*.

Figura 18 – Ferramentas de avaliação citadas pelos especialistas

Fonte: elaborada pelo autor (2025).

No que tange às ferramentas, vinte e oito especialistas citaram **49 ferramentas distintas** (Figura 18). Destacaram-se o *CT Test* (nove citações – 32,14 %), o *Bebras* (seis citações – 21,43 %) e as *Atividades Desplugadas* (três citações – 10,71 %). Um resultado bastante interessante que merece destaque é que 10 dos 28 pesquisadores (35,71 %) indicaram desenvolver internamente suas próprias ferramentas de avaliação de PC; a Tabela 3 apresenta o que cada um desses dez pesquisadores descreveu em suas respostas. Outro dado relevante é que seis pesquisadores (21,43 %) declararam não utilizar qualquer ferramenta de avaliação de PC.

Tabela 3 – Ferramentas de avaliação desenvolvidas internamente

Ferramentas internas descritas pelos pesquisadores
Engineering Computational Thinking Diagnostic (ECTD)
Ferramenta baseada em Modelo de Markov
Scripts próprios para análise de jogos
Instrumentos criados pelo grupo de pesquisa
Testes pré e pós desenvolvidos internamente
Questionário de percepções de PC (vide https://ieeexplore.ieee.org/abstract/document/9453940)
Criamos nossa própria ferramenta de avaliação tomando como referência instrumentos já consolidados, como o *CTT* de Gonzolas, o *CTMA* de Wiebe e o *ACE* de Parker.

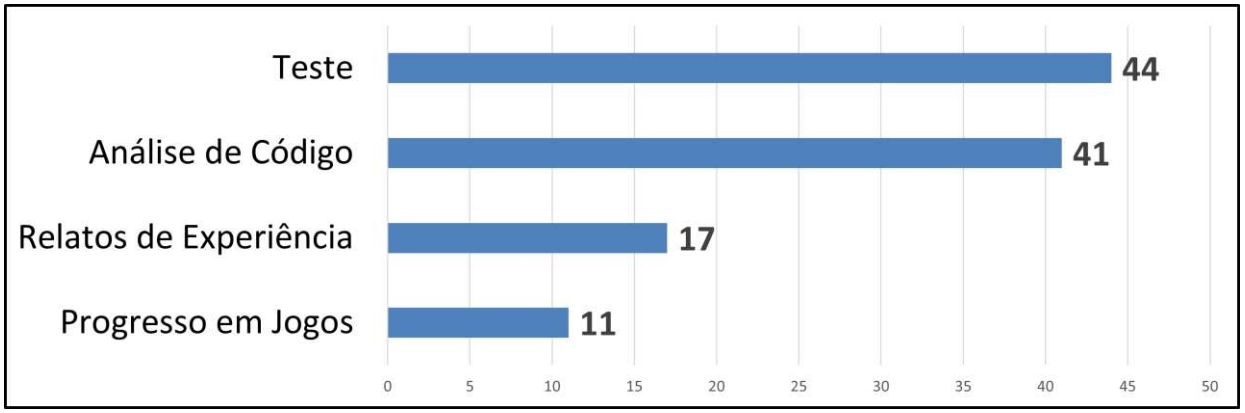
Fonte: elaborada pelo autor (2025).

A elevada incidência de soluções “caseiras” evidencia a necessidade de instrumentos flexíveis, modulares e ajustáveis às múltiplas dimensões do constructo, a fim de atender simultaneamente às demandas de pesquisa e às exigências da prática pedagógica.

5.2.4 Estratégias e Instrumentos de Avaliação PC

Esta seção expõe os resultados destinados a elucidar a terceira e última questão de pesquisa: “Quais estratégias e instrumentos são empregados na avaliação do Pensamento Computacional?”.

Figura 19 – Instrumentos de avaliação de pc



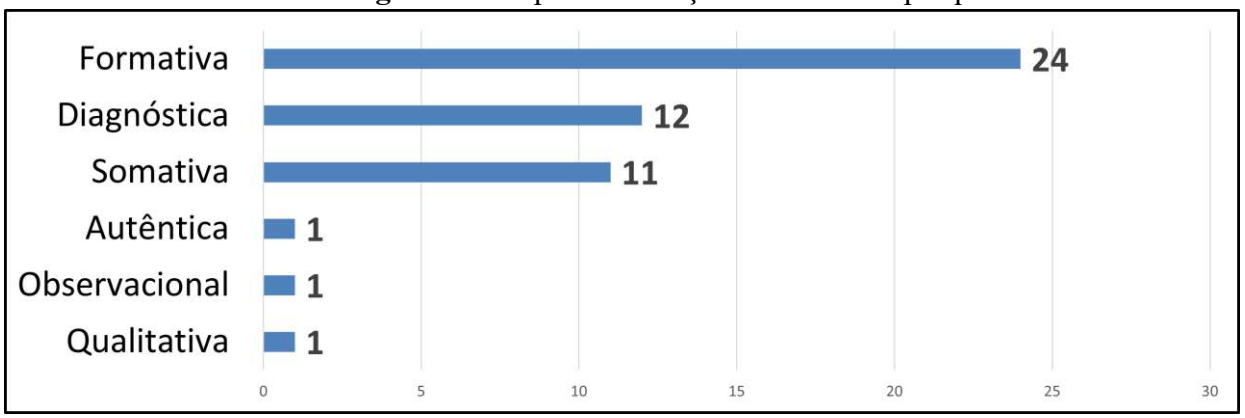
Fonte: elaborada pelo autor (2025).

No MSL, prevaleceram os instrumentos de *testes*, empregados em 44 artigos (54,32 %), e de *análise de código*, em 41 (50,61 %). *Relatos de experiência* e métricas de *progresso em jogos* surgiram em 17 (20,98 %) e 11 estudos (13,58 %), respectivamente (Figura 19), evidenciando a hegemonia de abordagens quantitativas, ainda que se observe interesse crescente por evidências qualitativas na avaliação de *PC*.

Conforme ilustrado na Figura 20, as avaliações *formativas* foram citadas por 24 pesquisadores, enquanto as modalidades *diagnóstica* e *somativa* receberam, respectivamente, 12 e 11 menções.

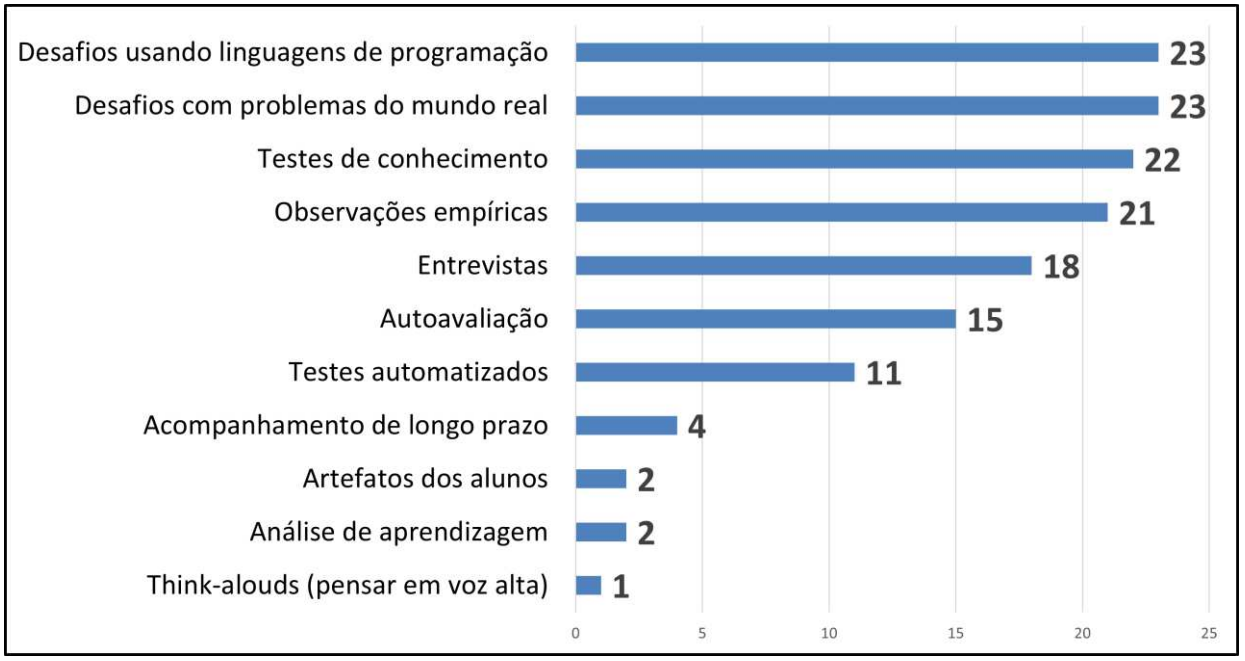
Entre os 40 especialistas consultados foram descritas **142 estratégias avaliativas** distintas (Figura 21). Os métodos mais recorrentes foram *desafios contextualizados com problemas do mundo real* e *desafios baseados em linguagens de programação*, ambos com 23 citações; em seguida vieram *testes tradicionais* (22), *observações empíricas* (21) e *entrevistas semiestruturadas* (18). Juntos, os dois tipos de desafios respondem por 57,5 % das menções, evidenciando a preferência por tarefas práticas e autênticas que exigem aplicação explícita dos conceitos de *PC*. Complementam a lista *autoavaliações* e *testes automatizados*, indicando uma diversidade metodológica que permite triangulação de evidências e captura simultânea de aspectos conceituais, procedimentais e atitudinais do pensamento computacional.

Figura 20 – Tipo de avaliação utilizada nas pesquisas



Fonte: elaborada pelo autor (2025).

Figura 21 – Estratégias de avaliação de pc mais utilizadas



Fonte: elaborada pelo autor (2025).

A predominância de **desafios contextualizados** evidencia a valorização de avaliações autênticas e aplicadas, pois tais tarefas exigem que os aprendizes mobilizem o pensamento computacional em situações que extrapolam exercícios artificiais e se aproximam de problemas do mundo real. Esse alinhamento entre avaliação e prática profissional potencializa a transferência da aprendizagem e favorece a mensuração de habilidades de ordem superior, como abstração, decomposição e depuração colaborativa. Paralelamente, a diversidade de instrumentos — testes tradicionais, análise de código, métricas de desempenho em jogos, entrevistas e autoavaliações — reforça a necessidade de abordagens multifacetadas capazes de captar o amplo espectro de competências que compõem o *PC*. Essa pluralidade não apenas viabiliza a triangulação de

dados, aumentando a validade dos resultados, como também oferece flexibilidade para adaptar os métodos às especificidades de diferentes públicos, faixas etárias e contextos educacionais.

5.3 Considerações Finais

Os resultados desta pesquisa evidenciam que a avaliação do PC enfrenta obstáculos substanciais, destacando-se: (i) a complexidade intrínseca do conteúdo; (ii) a ausência de padronização e de uma definição operacional consensual do constructo; e (iii) a carência de instrumentos adequados.

A recorrência de soluções avaliativas proprietárias revela a demanda por instrumentos flexíveis, modulares e ajustáveis às diversas dimensões do PC, capazes de contemplar simultaneamente as exigências da pesquisa acadêmica e da prática pedagógica. Além disso, a predominância de **desafios contextualizados** indica a valorização de avaliações autênticas e aplicadas, dado que tais tarefas exigem que os aprendizes mobilizem o pensamento computacional em situações que transcendem exercícios artificiais e se aproximam de problemas reais. Esse alinhamento entre avaliação e prática profissional potencializa a transferência da aprendizagem e favorece a mensuração de habilidades de ordem superior, como abstração, decomposição, reconhecimento de padrões e pensamento algorítmico.

Paralelamente, a diversidade de instrumentos—testes tradicionais, análise de código, métricas de desempenho em jogos, entrevistas e autoavaliações—reforça a necessidade de abordagens multifacetadas capazes de abarcar o amplo espectro de competências que compõem o PC. Em síntese, à medida que o campo evolui, torna-se imperativo desenvolver instrumentos avaliativos que combinem validade estatística, escalabilidade e sensibilidade pedagógica, de modo a subsidiar tanto a investigação acadêmica quanto a prática em sala de aula. Tal estratégia permitirá aprofundar o entendimento científico sobre o constructo e promover intervenções educacionais mais eficazes e equitativas.

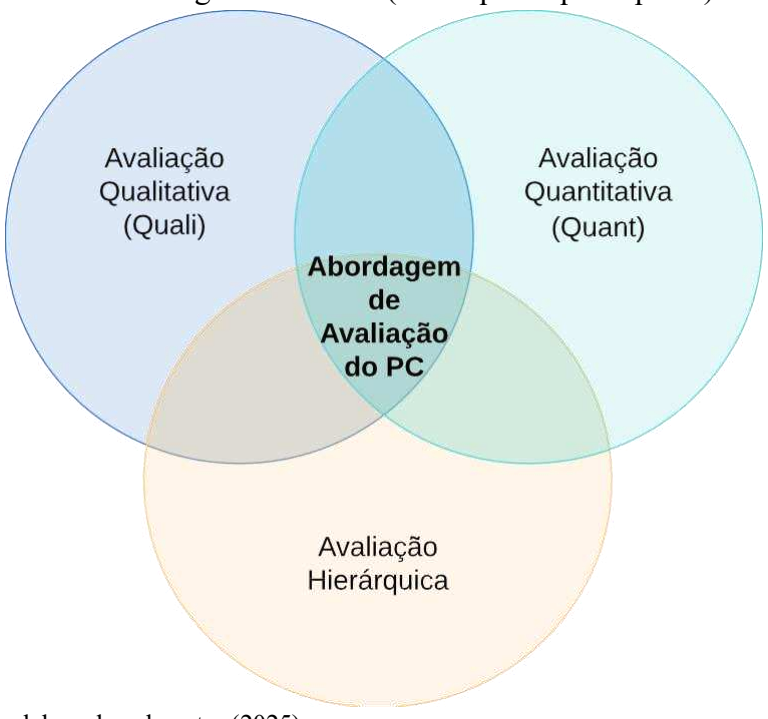
Nesse contexto, propõe-se o desenvolvimento e a avaliação de um instrumento de avaliação do PC composto por desafios específicos, de aplicação simples, dotado de calibração automática de dificuldade, capaz de abranger múltiplas dimensões do PC, oferecer avaliações formativas e operar eficientemente em turmas numerosas. A criação de instrumentos válidos, confiáveis e ajustados a diferentes perfis de aprendizes emerge, portanto, como prioridade estratégica para impulsionar, de forma sinérgica, a pesquisa e a prática educacional.

6 AVALIAÇÃO QUALI-QUANTI HIERÁRQUICA DO PC

Nesta seção, são apresentados em detalhes a estrutura e o funcionamento da abordagem avaliativa adotada nesta tese de doutorado, que integra uma abordagem hierárquica combinando análises qualitativas (quali) e quantitativas (quanti).

A avaliação qualitativa permite observar aspectos subjetivos e contextuais da aprendizagem, tais como as estratégias de raciocínio empregadas pelos estudantes, o desenvolvimento de habilidades metacognitivas e a forma como interagem entre si e com o ambiente de ensino. Já a avaliação quantitativa complementa esse quadro, oferecendo métricas objetivas que permitem comparar resultados, verificar progressos ao longo do tempo e identificar tendências mais amplas dentro de grupos ou populações de estudantes. Além disso, nossa abordagem possui um arranjo avaliativo hierárquico, que permite que as avaliação qualitativa e quantitativa, possam ser realizadas por meio da análise granular de cada componente do PC, assegurando rigor metodológico e rastreabilidade no processo avaliativo.

Figura 22 – Abordagem avaliativa (hierárquica-quali-quanti)



Fonte: elaborada pelo autor (2025).

A Figura 22 apresenta um diagrama de Venn que representa a abordagem avaliativa adotada nesta tese, estruturada a partir da interseção de três aspectos complementares. Essa interseção reflete a integração dos diferentes componentes metodológicos que sustentam a abordagem avaliativa desenvolvida. Ao articular os resultados dessas três vertentes, busca-se

constituir um retrato mais fiel do desenvolvimento de habilidades no âmbito do PC, levando em consideração tanto indicadores empíricos quanto o contexto e as dinâmicas de sala de aula. Dessa forma, a abordagem avaliativa não se restringe à mensuração de acertos e erros, mas procura também compreender as motivações, dificuldades e práticas que sustentam a aprendizagem, evidenciando elementos que podem orientar futuras intervenções pedagógicas e o aperfeiçoamento das estratégias de ensino.

É válido salientar que a abordagem de avaliação do PC utilizada nesta tese foi fundamentada na metodologia apresentada em (OLIVEIRA; PEREIRA, 2023). A abordagem apresentada subdivide o PC em um conjunto de habilidades, detalhando-as em evidências específicas.

Em sua pesquisa, os autores apresentam uma abordagem estruturada que inclui:

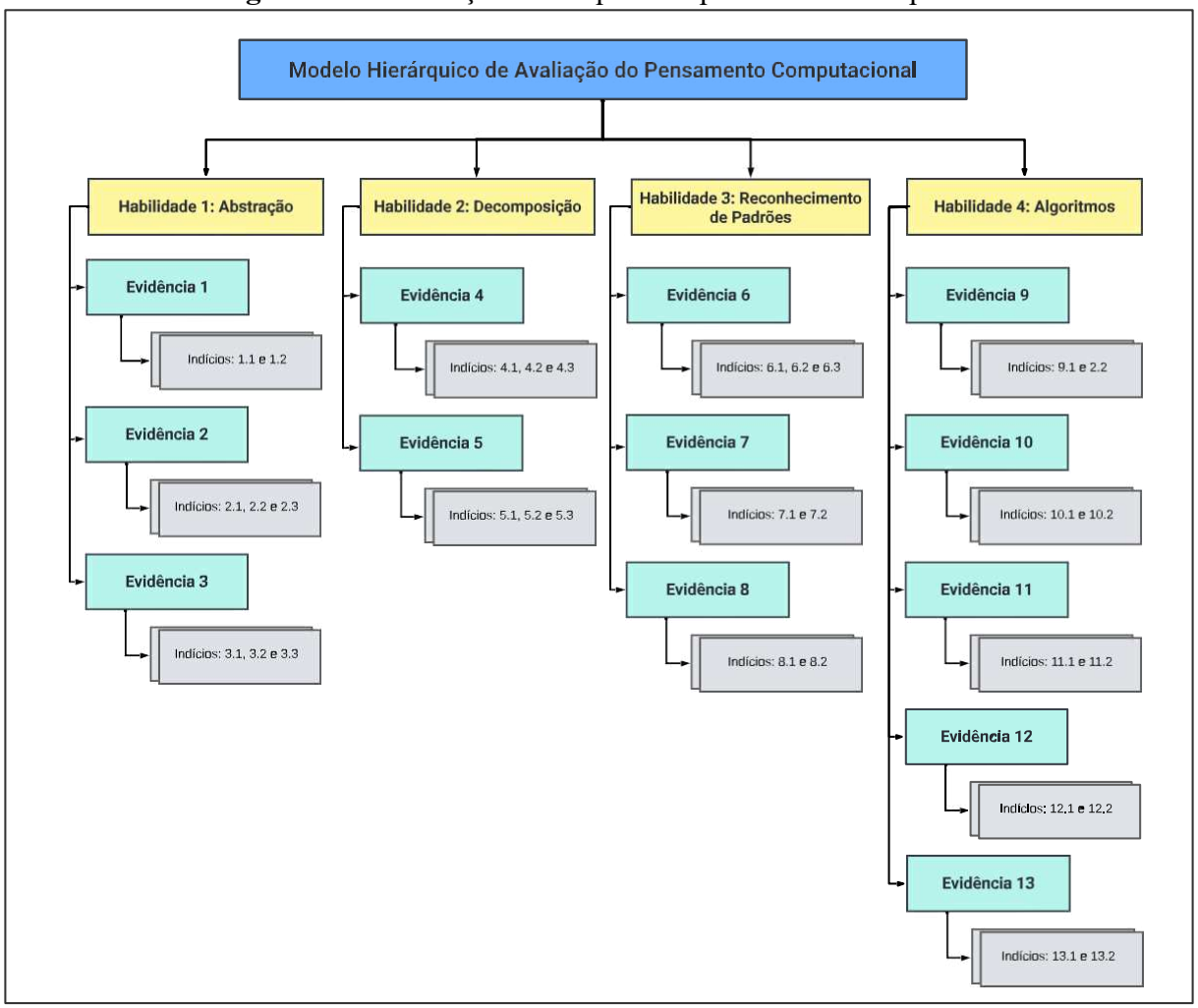
1. um conjunto de diretrizes para preparar e conduzir exercícios relacionados ao PC;
2. um conjunto de sete práticas (desafios) que podem ser aplicadas diretamente ou servir de inspiração para a criação de novos desafios;
3. um método para orientar a condução das práticas e instruir os estudantes na proposição de soluções; e
4. um artefato destinado a apoiar a análise das soluções, facilitando a coleta de evidências relacionadas ao exercício do Pensamento Computacional chamado de Computacionalidade.

6.1 Avaliação Hierárquica do Pensamento Computacional

Com o objetivo de organizar de forma sistemática e criteriosa o processo de avaliação do PC, esta tese propõe uma avaliação hierárquica, no qual o PC é estruturado em habilidades, que se desdobram em evidências observáveis, as quais, por sua vez, são decompostas em indícios específicos e mensuráveis. Essa abordagem hierárquica proporciona uma análise detalhada e orientada, assegurando que cada componente do desenvolvimento do PC seja devidamente identificado, categorizado e avaliado de maneira consistente, precisa e objetiva.

A Figura 23 apresenta uma representação geral da Abordagem Hierárquica de Avaliação do PC, desenvolvido no contexto desta pesquisa. Esta abordagem sintetiza a organização conceitual adotada, evidenciando a relação hierárquica entre habilidades, evidências e indícios que fundamentam o processo avaliativo proposto. Apresentando de maneira geral como as habilidades do PC se desdobram em evidências observáveis, sustentadas por indícios específicos e mensuráveis.

Figura 23 – Avaliação hierárquica do pensamento computacional



Fonte: elaborada pelo autor (2025).

Essa abordagem está estruturada em quatro habilidades principais: Abstração, Decomposição, Reconhecimento de Padrões e Algoritmos, as quais representam os pilares cognitivos do PC. Cada habilidade desdobra-se em um conjunto de evidências observáveis, que refletem comportamentos, práticas e decisões adotadas pelo sujeito durante a resolução de problemas computacionais. Por sua vez, cada evidência é sustentada por um conjunto de indícios, que consistem em elementos específicos, objetivos e mensuráveis no artefato produzido, como trechos de código, comentários, estruturação lógica ou padrões de implementação. Essa estrutura hierárquica visa garantir robustez e consistência no processo de avaliação, permitindo uma análise granular e precisa das habilidades associadas ao desenvolvimento do PC.

Essa organização permite uma análise do processo de aprendizagem, pois não se limita à verificação do resultado final de uma tarefa ou atividade. Os critérios empregados para cada indício encontram-se disponíveis no Apêndice F.

Conforme apresentado anteriormente, esta tese adota como uma de suas bases teóricas o estudo de (OLIVEIRA; PEREIRA, 2023), no qual os autores propõem uma estrutura de subdivisão do PC em 13 habilidades, organizadas em categorias de evidências específicas. Entretanto, para garantir maior rigor na avaliação, optou-se por ampliar o escopo das evidências originais, decompondo-as em indícios específicos capazes de revelar de forma mais minuciosa como cada componente do PC se manifesta no código produzido pelos estudantes. Essa estratégia de refinamento permite um monitoramento mais preciso do progresso de cada participante, bem como a identificação de possíveis lacunas ou fortalezas relacionadas a conceitos fundamentais de programação, tais como estruturas de controle, manipulação de dados e pensamento lógico. Ademais, a adoção de indicadores mais detalhados subsidia a elaboração de relatórios de desempenho e a proposição de intervenções pedagógicas direcionadas, além de gerar evidências empíricas robustas para futuros estudos.

6.1.1 Habilidades do Pensamento Computacional

Na primeira versão, optou-se por dividir o PC em quatro habilidades, as quais são amplamente reconhecidas como pilares fundamentais para o desenvolvimento de pensamento computacional: **(i) Abstração**, **(ii) Decomposição**, **(iii) Reconhecimento de Padrões** e **(iv) Algoritmos**.

A escolha por essas quatro habilidades foi motivada por diversos fatores. Primeiramente, eles constituem uma definição amplamente aceita e difundida em diferentes contextos educacionais ao redor do mundo, conforme apontado em estudos de referência (WING; STANZIONE, 2016; PALTS; PEDASTE, 2020; ZORZO *et al.*, 2017). Em segundo lugar, essa abordagem tem como vantagem a simplicidade de aplicação na avaliação, tornando mais clara a identificação dos progressos e dificuldades dos estudantes em cada habilidade específica. Por fim, a *Base Nacional Comum Curricular* (BNCC) adota a mesma subdivisão para o ensino de Computação, o que reforça sua adequação ao contexto educacional brasileiro e assegura alinhamento com as diretrizes pedagógicas vigentes.

Ao adotar esses quatro pilares como base para o artefato, garante-se um alinhamento consistente com padrões internacionais de ensino de Computação, além de cumprir as orientações da BNCC. Esse enquadramento facilita a adoção do artefato por professores, que podem facilmente relacionar cada habilidade às competências esperadas em sala de aula, bem como orientar as práticas pedagógicas de forma intencional e estruturada. Além disso, a clareza pro-

porcionada por essas categorias contribui para o processo avaliativo, permitindo que educadores identifiquem com mais precisão qual o estudante apresenta maiores dificuldades ou demonstrou maior evolução.

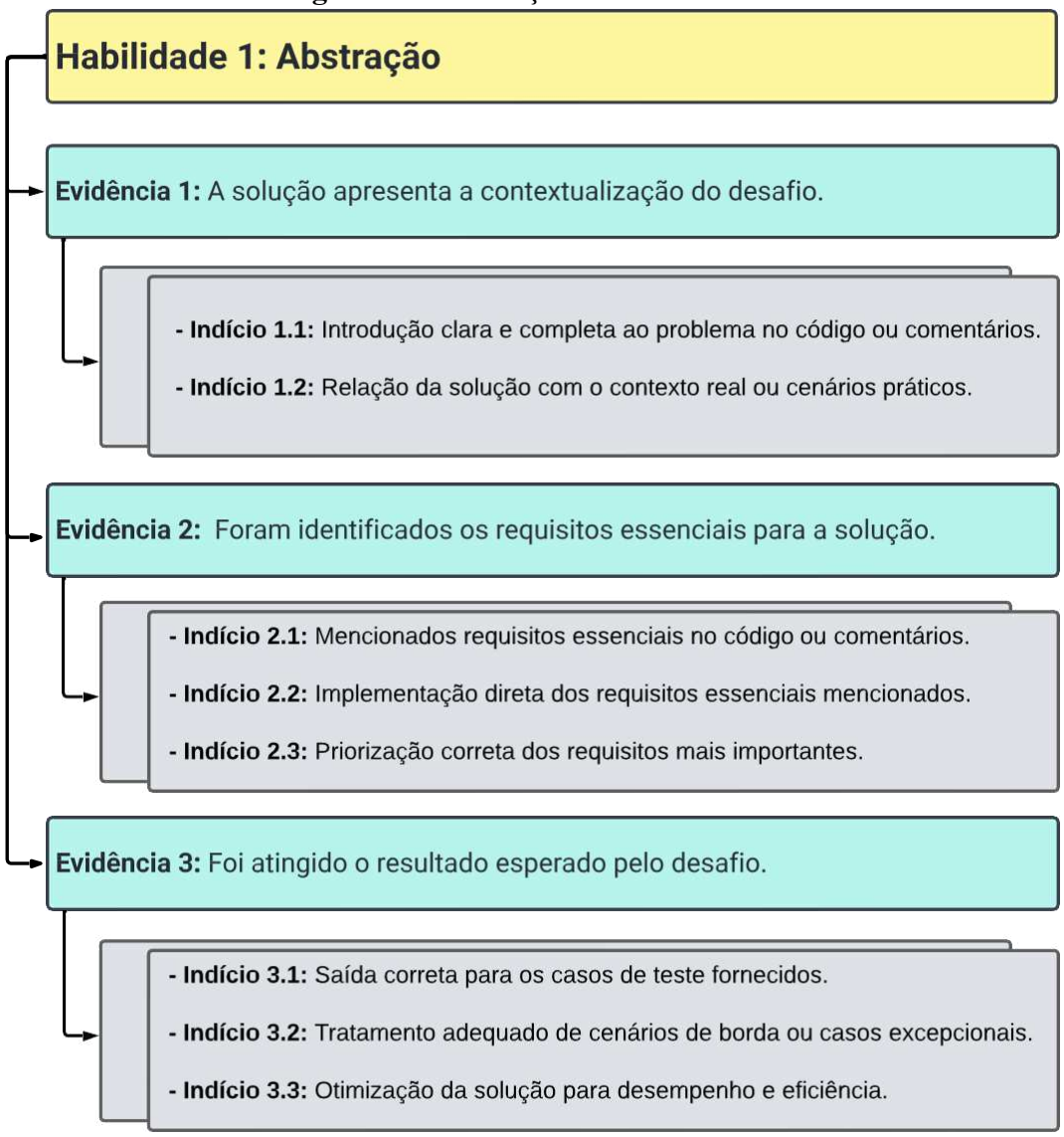
Dessa forma, a subdivisão em Abstração, Decomposição, Reconhecimento de Padrões e Algoritmos constitui não apenas um recorte teórico amplamente validado, mas também uma ferramenta pragmática para a construção, aplicação e avaliação do PC no contexto educacional.

6.1.1.1 Abstração

Considerada por muitos pesquisadores como o cerne do PC, a abstração envolve a habilidade de destacar os elementos essenciais de um problema e, simultaneamente, ignorar ou simplificar os aspectos que não são relevantes para sua resolução (WING; STANZIONE, 2016; PALTS; PEDASTE, 2020). Esse processo viabiliza uma compreensão mais clara do cerne da questão, permitindo que os estudantes tracem estratégias de solução mais focadas. Em sala de aula, a prática da abstração pode ser identificada, por exemplo, na forma como o aluno filtra informações redundantes ou desnecessárias ao descrever o problema e ao criar modelos simplificados que representem apenas os dados fundamentais para a solução.

A Figura 24 representa a modelagem hierárquica da **Habilidade Abstração**, composta por três **evidências avaliativas**, cada uma subdividida em **indícios operacionais**, que funcionam como critérios objetivos para a análise do desempenho dos participantes no desenvolvimento de soluções computacionais. A **Evidência 1** avalia a capacidade de contextualização do desafio, operacionalizada pelos indícios **1.1**, que verifica a presença de uma introdução clara e completa ao problema no código ou nos comentários, e **1.2**, que observa se há uma relação explícita entre a solução desenvolvida e o contexto real ou cenários práticos. A **Evidência 2** refere-se à identificação dos requisitos essenciais, sendo mensurada pelos indícios **2.1**, que verifica a menção explícita dos requisitos no código ou comentários; **2.2**, que analisa a implementação efetiva desses requisitos; e **2.3**, que avalia a priorização correta dos requisitos mais relevantes para a solução proposta. Por sua vez, a **Evidência 3** está associada à avaliação do resultado esperado, sustentada pelos indícios **3.1**, que examina a geração de saídas corretas frente aos casos de teste fornecidos; **3.2**, que verifica o tratamento adequado de cenários de borda ou casos excepcionais; e **3.3**, que analisa se foram aplicadas práticas de otimização visando desempenho e eficiência do código.

Figura 24 – Abstração - evidências e indícios



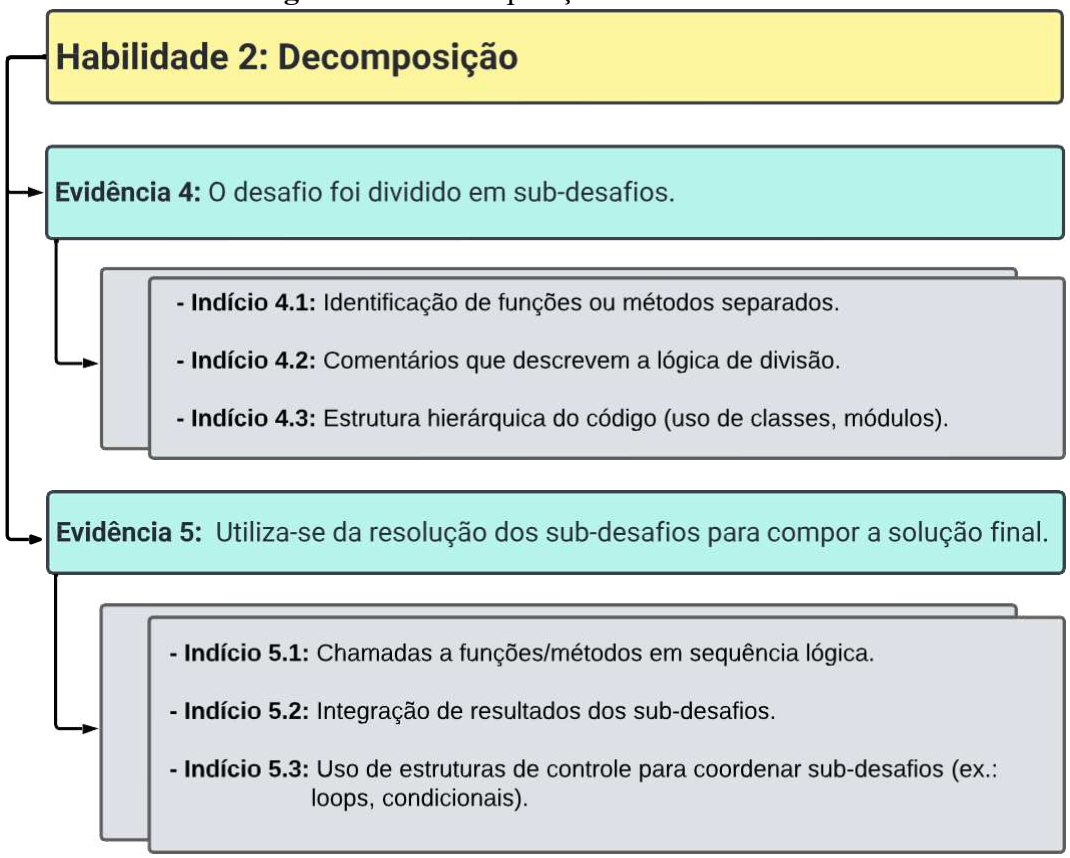
Fonte: elaborada pelo autor (2025).

6.1.1.2 Decomposição

A decomposição complementa o processo de abstração ao permitir que o problema seja fragmentado em partes menores e mais manejáveis (WING; STANZIONE, 2016; PALTS; PEDASTE, 2020). Essa estratégia favorece um entendimento passo a passo dos desafios, facilitando a elaboração de pequenas metas que podem ser solucionadas de maneira incremental. Em termos pedagógicos, a decomposição pode ser observada quando os alunos repartem um problema complexo (por exemplo, um jogo ou aplicativo) em funcionalidades ou etapas lógicas, trabalhando gradualmente em cada componente.

A Figura 25 apresenta a modelagem hierárquica da **Habilidade Decomposição**, composta por duas **evidências avaliativas**, cada uma subdividida em **indícios operacionais**, que

Figura 25 – Decomposição - evidências e indícios



Fonte: elaborada pelo autor (2025).

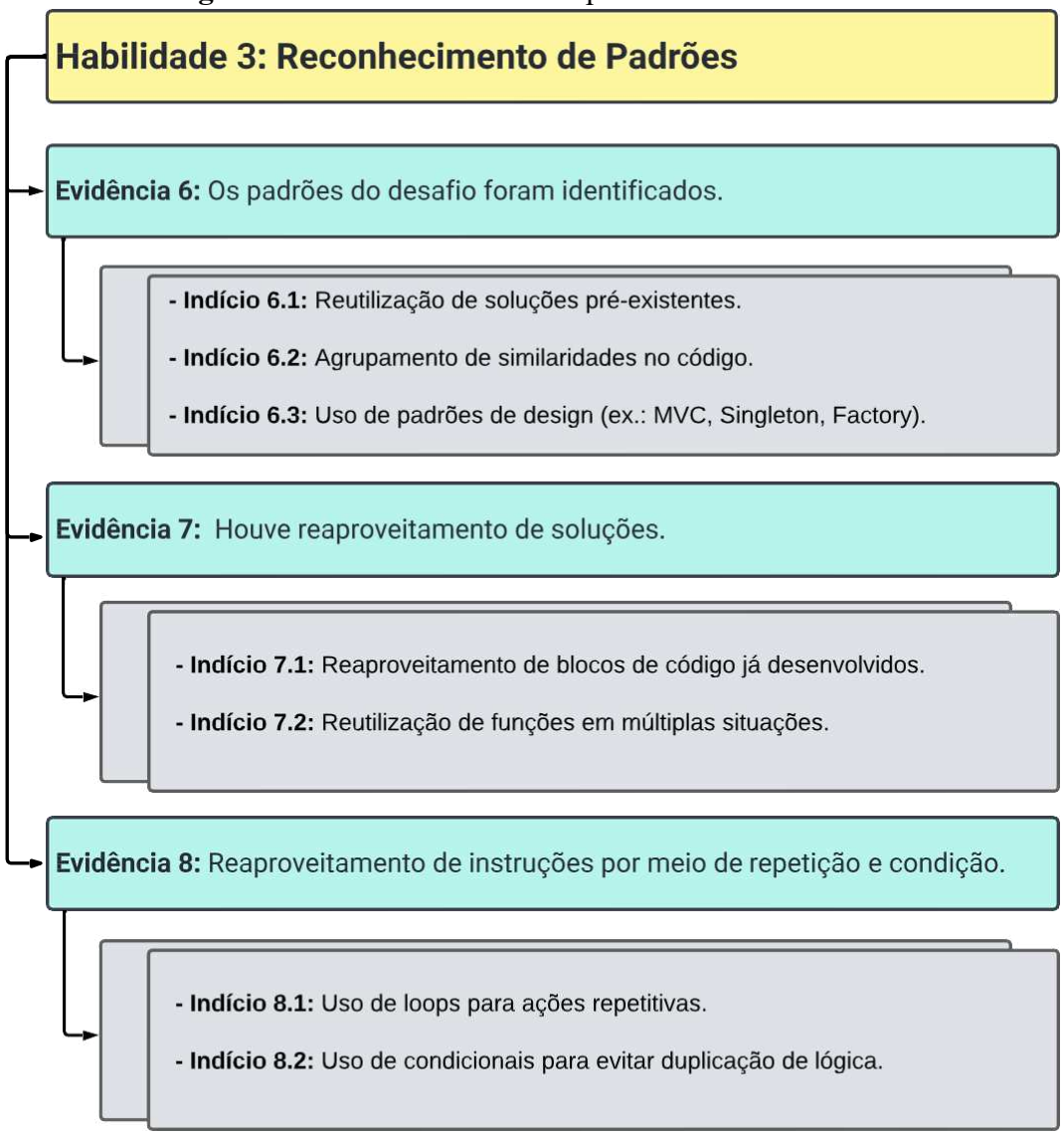
servem como critérios objetivos para a análise da capacidade dos participantes em decompor problemas computacionais em partes menores e gerenciáveis. A **Evidência 4** verifica se o desafio foi devidamente dividido em sub-desafios, sendo operacionalizada pelos indícios **4.1**, que observa a presença de funções ou métodos separados no código; **4.2**, que avalia a existência de comentários que descrevem claramente a lógica da divisão realizada; e **4.3**, que examina o uso de estruturas hierárquicas no código, como classes, módulos ou outras formas de encapsulamento funcional. A **Evidência 5** analisa se a solução final resulta da composição correta dos sub-desafios identificados, sustentada pelos indícios **5.1**, que verifica se há chamadas a funções ou métodos organizadas em sequência lógica; **5.2**, que avalia a integração adequada dos resultados obtidos em cada sub-desafio; e **5.3**, que examina o uso de estruturas de controle (tais como loops e condicionais) para coordenar a execução dos subcomponentes da solução.

6.1.1.3 Reconhecimento de Padrões

Ao identificar similaridades e regularidades entre diferentes problemas ou soluções, o reconhecimento de padrões permite que os estudantes transfiram conhecimentos adquiridos

de um contexto para outro (ZORZO *et al.*, 2017). Tal habilidade é fundamental para promover eficiência e criatividade, uma vez que possibilita a reutilização de abordagens testadas e a adaptação de estratégias bem-sucedidas a novos cenários. Em aulas de programação, por exemplo, essa competência pode ser avaliada quando os alunos percebem semelhanças entre estruturas de código, solucionam exercícios análogos de maneira mais ágil ou formulam hipóteses que generalizam situações aparentemente distintas.

Figura 26 – Reconhecimento de padrões - evidências e indícios



Fonte: elaborada pelo autor (2025).

A Figura 26 apresenta a modelagem hierárquica da **Habilidade Reconhecimento de Padrões**, estruturada em três **evidências avaliativas**, cada uma suportada por um conjunto de **indícios operacionais** que permitem a análise objetiva da capacidade dos participantes em identificar, abstrair e reaproveitar padrões durante a resolução de problemas computacionais. A

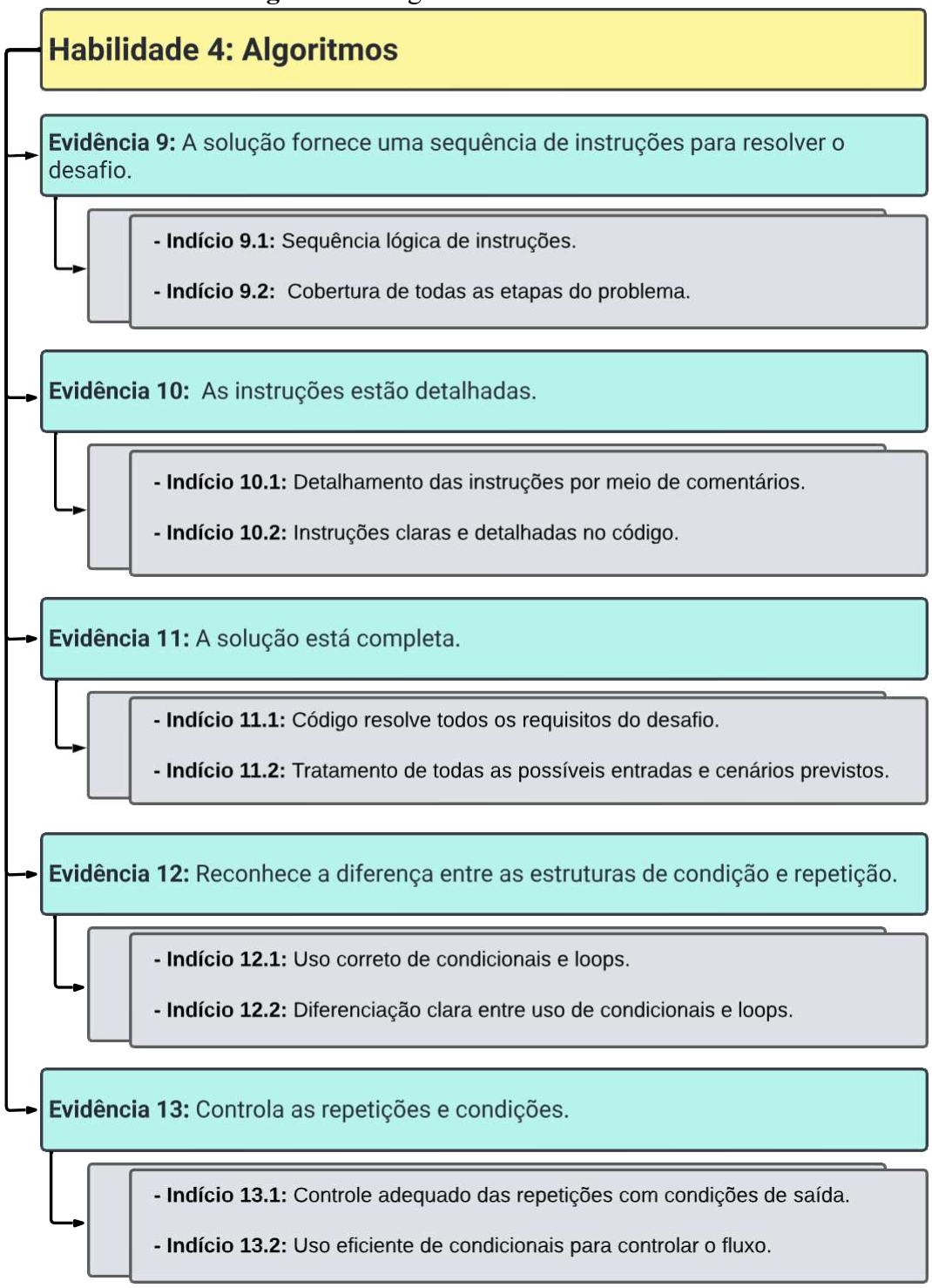
Evidência 6 refere-se à identificação dos padrões presentes no desafio, sendo mensurada pelos indícios **6.1**, que verifica a reutilização de soluções pré-existentes; **6.2**, que avalia o agrupamento de similaridades no código, como funções, estruturas de dados ou lógicas recorrentes; e **6.3**, que observa a adoção de padrões de design, tais como *Model-View-Controller* (MVC), *Singleton* ou *Factory*. A **Evidência 7** avalia o reaproveitamento de soluções anteriormente desenvolvidas, sustentada pelos indícios **7.1**, que examina o reaproveitamento direto de blocos de código, e **7.2**, que verifica a reutilização de funções ou métodos em diferentes contextos da solução. Por fim, a **Evidência 8** analisa o reaproveitamento de instruções por meio do uso de estruturas de repetição e condição, operacionalizada pelos indícios **8.1**, que observa a utilização de loops para a realização de ações repetitivas, e **8.2**, que verifica o uso adequado de condicionais para evitar duplicação de lógica.

6.1.1.4 Algoritmos

Por fim, o pilar de algoritmos compreende a habilidade de construir sequências lógicas de instruções que, em conjunto, levam à solução do problema (PALTS; PEDASTE, 2020; ZORZO *et al.*, 2017). Esse pilar abrange tanto a clareza de raciocínio na descrição das etapas de solução quanto o domínio de conceitos fundamentais de lógica, como variáveis, estruturas de controle de fluxo (condicionais e laços) e organização de dados. A avaliação desse aspecto pode envolver, por exemplo, a análise de pseudocódigos, fluxogramas, ou mesmo a implementação de um programa funcional.

A Figura 27 apresenta a modelagem hierárquica da **Habilidade Algoritmos**, composta por cinco **evidências avaliativas**, cada uma sustentada por **indícios operacionais** que permitem avaliar de forma objetiva a competência dos participantes na construção de algoritmos para a resolução de problemas computacionais. A **Evidência 9** verifica se a solução apresenta uma sequência de instruções coerente e adequada, mensurada pelos indícios **9.1**, que avalia a existência de uma sequência lógica de instruções, e **9.2**, que examina se todas as etapas necessárias para a resolução do problema estão devidamente contempladas. A **Evidência 10** avalia o nível de detalhamento das instruções fornecidas, operacionalizada pelos indícios **10.1**, que observa o uso de comentários para descrever as instruções, e **10.2**, que verifica se as instruções estão claras e suficientemente detalhadas diretamente no código. A **Evidência 11** está associada à completude da solução, sustentada pelos indícios **11.1**, que avalia se o código atende integralmente aos requisitos do desafio, e **11.2**, que verifica se há tratamento adequado para todas as

Figura 27 – Algoritmos - evidências e indícios



Fonte: elaborada pelo autor (2025).

possíveis entradas e cenários previstos. A **Evidência 12** analisa a capacidade do participante em reconhecer a diferença entre estruturas de condição e repetição, operacionalizada pelos indícios **12.1**, que verifica o uso correto dessas estruturas, e **12.2**, que observa a diferenciação clara no uso de condicionais e loops. Por fim, a **Evidência 13** avalia a competência no controle de repetições e condições, sustentada pelos indícios **13.1**, que verifica se há controle adequado das

repetições por meio de condições de saída bem definidas, e **13.2**, que analisa se os condicionais são utilizados de maneira eficiente para gerenciar o fluxo de execução do código.

6.2 Avaliação Qualitativa do PC

Nesta subseção, apresenta-se a análise qualitativa, voltada à compreensão de cada indício do PC evidenciado pelos estudantes ao longo das atividades de programação e resolução de problemas. Diferentemente de uma perspectiva puramente quantitativa, que foca em métricas de desempenho ou índices de acerto, a abordagem qualitativa se concentra nos processos, estratégias e reflexões que os estudantes empregam.

Para cada indício de PC identificado, adota-se um procedimento que oferece tanto um direcionamento pedagógico quanto uma verificação precisa de como essas habilidades são empregadas na prática:

1. **Feedback personalizado:** O artefato fornece uma descrição textual indicando se o estudante atendeu ou não ao indício em questão. Essa descrição abrange não apenas a constatação da presença (ou ausência) da habilidade, mas também a apresentação de exemplos concretos e sugestões de melhoria. Dessa forma, o estudante obtém uma compreensão clara sobre quais aspectos de seu raciocínio computacional estão bem desenvolvidos e quais necessitam de aperfeiçoamento.
2. **Evidências no código:** Paralelamente, o artefato destaca partes específicas do código que demonstram (ou deveriam demonstrar) a manifestação do indício. Ao apontar diretamente em que trecho do programa a habilidade foi empregada — ou deveria ter sido —, obtém-se uma visão objetiva sobre como o raciocínio do estudante se traduz em elementos concretos de programação. Além de facilitar o acompanhamento do progresso, esse procedimento fornece subsídios para correções pontuais e orienta o professor a propor intervenções pedagógicas mais direcionadas.

Essa abordagem qualitativa assegura um retorno individualizado e esclarecedor, permitindo ao estudante identificar, em quais momentos e de que maneira suas habilidades de PC foram evidenciadas ou deixaram de se manifestar. O resultado é um processo avaliativo que estimula o aprimoramento contínuo do pensamento computacional.

6.3 Avaliação Quantitativa do PC

Para fornecer um grau adicional de precisão na determinação de níveis de desenvolvimento ds habilidades do PC, a abordagem avaliativa desta tese inclui uma análise quantitativa que atribui um conceito específico a cada indício. Nessa abordagem, inspirada no modelo proposto por (OLIVEIRA; PEREIRA, 2023) — que, por sua vez, fundamenta-se em (FRANÇA; SILVA, 2020) —, cada indício é enquadrado em um dos seguintes níveis:

- **Não se aplica**: Quando a habilidade ou o recurso do PC não é pertinente para a atividade realizada.
- **Nenhum**: Indica que não foram encontradas evidências de aplicação da habilidade em questão.
- **Baixo**: Aponta para uma manifestação incipiente do indício, com limitações claras na forma de aplicação ou entendimento.
- **Regular**: Sugere que o estudante demonstra o uso do indício, porém com inconsistências ou necessidade de maior aprofundamento.
- **Bom**: Evidencia o emprego adequado da habilidade, embora ainda existam oportunidades de refinamento e consolidação.
- **Ótimo**: Denota uma aplicação consistente e bem estruturada, indicando domínio expressivo do indício avaliado.

Essa classificação, ao quantificar os diferentes graus de proficiência, possibilita estabelecer métricas objetivas para cada indício identificado. Com isso, torna-se viável comparar resultados entre diferentes estudantes, acompanhar a evolução individual ao longo do tempo e diagnosticar áreas que requerem intervenções pedagógicas mais direcionadas. Em conjunto com a análise qualitativa, a adoção desse modelo híbrido (qualitativo-quantitativo) contribui para uma compreensão mais completa do desenvolvimento do PC.

Para cada indício, o artefato sugere um conceito (por exemplo, *Nenhum*, *Baixo*, *Regular*, *Bom* ou *Ótimo*) com base nas evidências identificadas no código e nas informações fornecidas pela análise qualitativa. Assim, o professor não só reconhece quais elementos de PC estão efetivamente presentes na solução do estudante, mas também obtém parâmetros objetivos para mensurar o grau de compreensão e aplicação de cada indício. Além disso, a análise quantitativa abre possibilidades para:

- **Acompanhamento Formativo:** Monitoramento contínuo do progresso, permitindo ao professor ajustar estratégias pedagógicas ao longo do curso e oferecer intervenções imediatas

quando necessário.

- **Criação de Estatísticas de Acompanhamento:** Elaboração de relatórios e análises consolidadas do desempenho dos estudantes em diferentes níveis (indivíduos, turmas ou até mesmo escolas), subsidiando a tomada de decisões mais assertivas e baseadas em dados.

O processo de avaliação, tanto quantitativo quanto qualitativo, oferece amplo potencial de personalização, permitindo que o professor selecione os indícios, evidências ou habilidades do PC a serem priorizados. Dessa maneira, a avaliação se adapta a distintos cenários e objetivos pedagógicos, desde turmas com perfis heterogêneos até disciplinas específicas. Essa flexibilidade possibilita:

- **Foco Temático:** Caso o professor deseje enfatizar apenas determinadas habilidades (por exemplo, Algoritmos ou Reconhecimento de Padrões), é possível estruturar a avaliação em torno desses aspectos.
- **Relevância Contextual:** Em cursos de nível avançado ou com uma proposta específica, pode-se ajustar o conjunto de indícios para abranger competências mais complexas, enquanto turmas iniciais podem trabalhar com um subconjunto mais simples.
- **Dinâmica Pedagógica:** A liberdade de escolha sobre o que se avalia e como se avalia permite um refinamento contínuo das estratégias de ensino e facilita o alinhamento entre expectativas curriculares, necessidades dos estudantes e diretrizes institucionais.

A abordagem avaliativa procura estar alinhada aos princípios de autonomia docente e de ensino centrado no aprendiz, proporcionando um processo avaliativo consistente e criterioso e versátil.

6.4 Considerações Finais

A abordagem avaliativa quali-quantitativa hierárquica do PC, apresentada neste capítulo, configura-se como uma abordagem robusta e sistemática. Ao combinar uma estrutura hierárquica — composta por habilidades, evidências e indícios — com abordagens qualitativa e quantitativa, torna-se possível realizar uma análise abrangente. Essa organização oferece uma visão detalhada do desenvolvimento das habilidades do PC de cada estudante.

Nos capítulos seguintes, serão detalhados o desenvolvimento do artefato computacional que operacionaliza essa abordagem avaliativa e os procedimentos adotados para sua avaliação empírica.

7 PROJETO E DESENVOLVIMENTO DO ARTEFATO

Neste capítulo, são expostos os aspectos que compõem o projeto e a implementação do artefato desenvolvido para a avaliação automática da aprendizagem em PC. Busca-se demonstrar tanto o arcabouço conceitual quanto as decisões práticas adotadas no processo de desenvolvimento.

É perceptível que esta tese, bem como o artefato descrito neste capítulo, sofreram influência direta do artefato "Computacionalidade" proposto em (OLIVEIRA; PEREIRA, 2023). Contudo, a análise dos resultados provenientes da aplicação prática do artefato evidenciou algumas oportunidades para aprimoramento. Dentre os principais desafios identificados, destaca-se a ausência de sugestões automatizadas (ou semi-automatizadas) para apoiar o processo avaliativo, o que mantém a análise fortemente vinculada ao conhecimento prévio do professor em Ciência da Computação e PC.

Além disso, o fato de a avaliação ser feita manualmente dificulta a padronização dos critérios de julgamento, pois cada professor pode empregar referências distintas ao determinar a qualidade ou o nível de proficiência dos estudantes. Essa carência de homogeneidade, por sua vez, limita a aplicabilidade do artefato em contextos com grande número de turmas ou em instituições que busquem adotar um processo avaliativo unificado. Dessa maneira, escalar a solução para diferentes realidades torna-se mais complexo, pois qualquer expansão requer a manutenção de um corpo docente especializado, familiarizado com os princípios do PC e alinhado em termos de critérios avaliativos.

Essas limitações reforçaram a necessidade de desenvolver mecanismos mais robustos que auxiliem o docente na análise do código e forneçam subsídios automatizados para a geração de feedback, garantindo, ao mesmo tempo, coerência, padronização e maior agilidade no processo de avaliação. Nas próximas seções, discutiremos os aprimoramentos propostos e como eles visam a mitigar esses entraves, otimizando a eficiência e a confiabilidade do artefato.

7.1 Projeto do Artefato

O artefato foi idealizado para atuar como um aliado do professor no processo de avaliação do desenvolvimento de PC pelos estudantes. Para isso, faz uso de uma análise automatizada dos códigos de programação criados a partir de desafios propostos em sala de aula. A intenção é fornecer um mecanismo que detecte como os conceitos e boas práticas de programação

refletem as habilidades do PC, tais como Abstração, Decomposição, Reconhecimento de Padrões e Algoritmos. Por meio desse processo automatizado, busca-se garantir maior consistência na avaliação, ao mesmo tempo em que se facilita o trabalho docente, permitindo ao professor concentrar seus esforços em intervenções pedagógicas direcionadas, em vez de despender tempo e energia em correções manuais extensivas.

7.1.1 Escolha do Python

Dentre as múltiplas linguagens de programação empregadas para estimular o desenvolvimento de PC, o uso de Python destaca-se por sua combinação de simplicidade sintática e grande versatilidade (GOU *et al.*, 2021). Esse equilíbrio torna Python especialmente atrativa para iniciantes, pois possibilita a criação de programas funcionais com código conciso e de fácil leitura, reduzindo barreiras iniciais de complexidade e motivando a exploração de conceitos de programação. Ademais, estudos conduzidos no âmbito desta pesquisa evidenciaram que Python se destaca como uma das ferramentas mais empregadas para ensinar PC, reforçando a escolha por essa linguagem na condução do experimento proposto.

Além disso, Python é amplamente empregado no contexto educacional, tanto para o desenvolvimento do Pensamento Computacional quanto para o ensino de programação em escolas e universidades. Sua sintaxe simples e a vasta gama de bibliotecas disponíveis tornam a linguagem particularmente adequada para iniciantes, ao mesmo tempo em que oferece recursos avançados para projetos mais complexos, possibilitando uma transição fluida do básico ao aprofundado no ambiente acadêmico (MARQUES *et al.*, 2011; BOGDANCHIKOV *et al.*, 2013; SILVA *et al.*, 2019b).

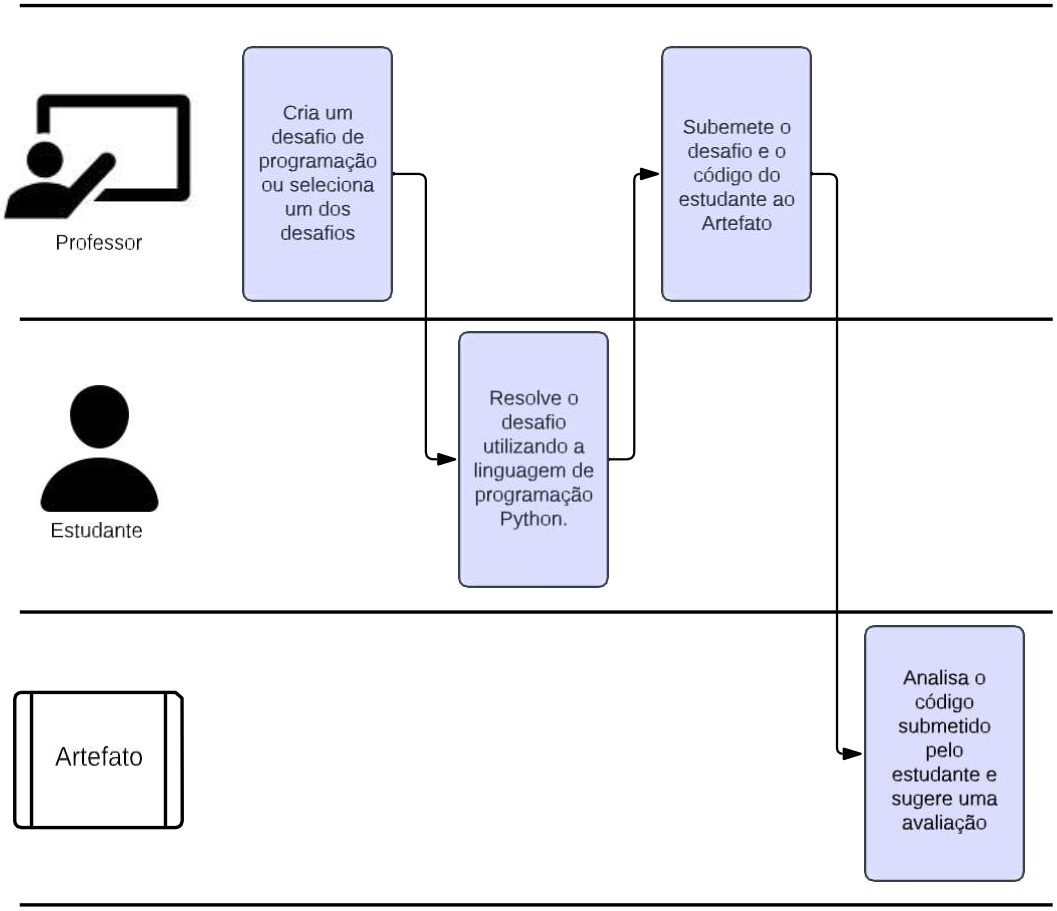
Embora a avaliação restrita exclusivamente a códigos em Python constitua uma limitação, conforme apresentado no Capítulo 10.3, já se delineia uma estratégia para expandir o escopo da avaliação, de modo a contemplar o suporte a outras linguagens de programação. Tal evolução visa promover uma abordagem mais abrangente e representativa das práticas de desenvolvimento, contribuindo para o avanço dos métodos avaliativos na área.

7.1.2 Fluxo da Avaliação

O fluxo de avaliação do desenvolvimento de PC, conforme proposto pelo artefato, está ilustrado na Figura 28. O processo, embora simples, abarca quatro etapas essenciais que orientam a condução da avaliação:

1. **Definição do Desafio**: O professor cria um desafio de programação personalizado ou opta por selecionar um dos desafios previamente cadastrados no artefato, garantindo que a dificuldade e o tema estejam alinhados aos objetivos de aprendizagem do grupo.
2. **Resolução pelo Estudante**: Cada estudante elabora sua solução usando a linguagem Python, empregando conhecimentos de PC no processo de desenvolvimento do código.
3. **Submissão à Plataforma**: Em seguida, o professor encaminha tanto o enunciado do desafio quanto o código produzido pelo estudante à plataforma. Esse passo centraliza as informações, de forma que o artefato possa realizar análises específicas sobre o desenvolvimento de PC.
4. **Análise Semi-automatizada e Avaliação**: O artefato analisa o código desenvolvido pelo estudante e sugere uma avaliação baseada no desenvolvimento do PC evidenciado no código.

Figura 28 – Fluxo de avaliação do pc



Fonte: elaborada pelo autor (2025).

Esse fluxo visa dar suporte ao trabalho docente ao automatizar parte da correção e da análise de competências relacionadas a PC, liberando o professor para se concentrar em

intervenções pedagógicas mais pontuais e na proposição de desafios ainda mais significativos para o desenvolvimento dos estudantes.

7.2 Aspectos Técnicos do Artefato

A construção de um artefato que integre análise e avaliação semi-automatizada demanda uma abordagem criteriosa na seleção das tecnologias, com o objetivo de assegurar escalabilidade, flexibilidade e desempenho. Para atingir esses objetivos, foi fundamental adotar ferramentas e *frameworks* capazes não apenas de processar grandes volumes de dados com eficiência, mas também de se adaptarem de forma dinâmica às mudanças nos requisitos pedagógicos e técnicos. Essa estratégia possibilitou o desenvolvimento de uma solução modular e atualizável, que pode evoluir em consonância com as inovações tecnológicas e as necessidades emergentes do ambiente educacional.

Nesta seção, detalhamos as tecnologias e os *frameworks* escolhidos para o desenvolvimento do artefato, justificando cada seleção com base em parâmetros como robustez, eficiência operacional e compatibilidade com os objetivos propostos para a avaliação do pensamento computacional. A integração de tecnologias modernas possibilitou a implementação de um artefato dinâmico e responsivo, capaz de realizar análises complexas que atendem às demandas específicas de identificação e mensuração de evidências de PC.

Entre os critérios de escolha, destacam-se:

- **Robustez:** A capacidade de manter a integridade e a consistência dos dados mesmo em cenários de alta carga ou múltiplas requisições simultâneas.
- **Eficiência:** A agilidade na execução dos algoritmos de análise, garantindo respostas em tempo hábil e otimizando o processo de avaliação.
- **Compatibilidade:** A facilidade de integração entre os diferentes componentes do artefato, assegurando que as ferramentas escolhidas se alinhem com os objetivos acadêmicos e metodológicos delineados para a avaliação do PC.
- **Manutenibilidade:** A facilidade de manutenção e atualização do artefato, permitindo a correção de erros e a implementação de aprimoramentos sem comprometer a estabilidade geral do artefato. Esse critério é fundamental para possibilitar a evolução contínua da ferramenta, adaptando-a a novas demandas pedagógicas e a avanços tecnológicos.

A abordagem adotada no desenvolvimento do artefato permite que o este seja evolutivo, possibilitando a incorporação de novas funcionalidades e a adaptação a diferentes contextos

de ensino, sem comprometer a consistência e a qualidade das análises.

Em resumo, o uso combinado de tecnologias modernas não só habilitou a implementação de um artefato automatizado de avaliação, mas também assegurou que o artefato fosse capaz de fornecer *feedbacks* detalhados e personalizados, facilitando a identificação e o acompanhamento do desenvolvimento das habilidades de PC.

7.2.1 Tecnologias Selecionadas

Para a implementação, optou-se por uma arquitetura modular que integra soluções de *backend* e *frontend*. A seguir, são apresentadas as tecnologias selecionadas, juntamente com as justificativas para suas escolhas.

7.2.1.1 Backend

O desenvolvimento do backend foi realizado em Python, uma linguagem reconhecida por sua versatilidade e forte integração com ferramentas de Inteligência Artificial (CHOLLET, 2017). Python possui uma rica variedade de bibliotecas e frameworks que facilitam o processamento de dados e a implementação de algoritmos complexos. Essa escolha se justifica por diversos motivos:

- *Integração com LLM:* A facilidade de integração com *Large Language Models* possibilita a implementação de funcionalidades avançadas, como a geração de documentos orientadores com base na descrição dos desafios.
- *Processamento Eficiente de Dados:* Bibliotecas como NumPy, Pandas e scikit-learn oferecem suporte robusto para o processamento e análise de grandes volumes de dados, o que é essencial para a avaliação do desempenho dos estudantes.
- *Comunidade Ativa e Recursos Didáticos:* Python é suportado por uma comunidade ativa, oferecendo ampla documentação e recursos que aceleram o desenvolvimento e a resolução de problemas técnicos, facilitando a manutenção e a evolução contínua do artefato.

7.2.1.2 Frontend

Para a camada de frontend, a escolha recaiu sobre o *framework React* devido à sua popularidade e à consolidação de seu ecossistema no desenvolvimento de interfaces de usuário interativas e responsivas (WIERUCH, 2020). Complementarmente, a biblioteca Vite foi integrada

para otimizar o ambiente de desenvolvimento. Essa combinação traz diversos benefícios:

- *Desenvolvimento Ágil e Modular:* React permite a criação de componentes reutilizáveis que facilitam a gestão e a escalabilidade da interface, contribuindo para um código mais organizado e de fácil manutenção.
- *Otimização do Ambiente de Desenvolvimento:* Vite proporciona um tempo de compilação reduzido e um ambiente de desenvolvimento mais dinâmico, o que melhora a experiência do desenvolvedor e permite iterações mais rápidas durante a construção e testes da interface.
- *Responsividade e Interatividade:* A combinação de React com Vite permite o desenvolvimento de interfaces ricas, dinâmicas e responsivas, garantindo que as informações sobre a avaliação automatizada sejam apresentadas de forma clara e acessível aos usuários finais.

A integração entre *backend* e *frontend*, alicerçada nas tecnologias selecionadas, resulta num artefato robusto e adaptável, capaz de atender às demandas técnicas específicas.

7.2.1.3 Integração com LLM

A análise do código-fonte e a geração de feedback textual foram viabilizadas por meio da integração com modelos de linguagem de grande porte LLM, como GPT e LLaMA3. Para garantir uma implementação flexível e modular, esses modelos foram incorporados utilizando o padrão de projeto *Adapter* (GAMMA, 1995). Essa abordagem permite a substituição ou atualização dos modelos de IA sem impactar significativamente a estrutura geral da aplicação, assegurando independência tecnológica e facilitando a manutenção e evolução do artefato.

A utilização do padrão *Adapter* oferece diversas vantagens:

- **Flexibilidade e Modularidade:** Permite a integração de diferentes modelos de linguagem de maneira plug-and-play, possibilitando a adoção de inovações tecnológicas e a migração para novos LLM conforme forem surgindo avanços na área.
- **Independência Tecnológica:** Ao desacoplar a lógica da aplicação da implementação específica do modelo de IA, o artefato se torna robusto contra mudanças de plataforma e atualizações de *Application Programming Interface* (API), mantendo sua funcionalidade de forma consistente.
- **Aprimoramento Contínuo:** Essa estratégia de design possibilita o ajuste fino dos feedbacks gerados pelo artefato, permitindo que a avaliação do código-fonte seja cada vez mais precisa e alinhada aos critérios estabelecidos para a avaliação do Pensamento

Computacional.

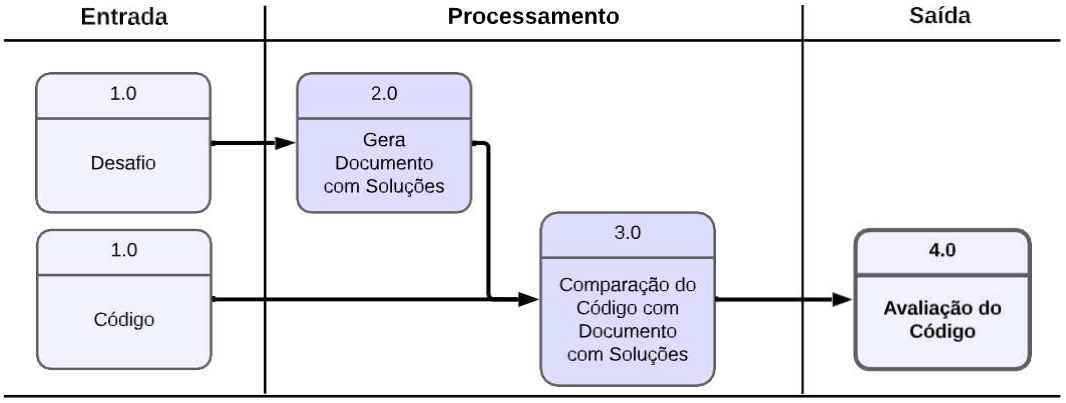
Dessa forma, a integração com LLM não apenas automatiza a análise e a geração de *feedbacks*, mas também confere ao artefato uma elevada capacidade de adaptação às mudanças tecnológicas, consolidando sua eficácia na avaliação semi-automatizada do desenvolvimento do PC.

O código completo está disponível no repositório GitHub, acessível em https://github.com/ederporfirio/tese_doutorado. Alguns trechos didaticamente úteis estão destacados no Apêndice G.

7.3 Fluxo: Entrada, Processamento e Saída

Nesta seção, detalhamos o fluxo operacional que rege o funcionamento do artefato, abrangendo as etapas de entrada, processamento e saída. A Figura 29 oferece uma visão global desse processo, ilustrando como cada componente interage para possibilitar a avaliação semi-automatizada do desenvolvimento do PC.

Figura 29 – Fluxo de entrada, processamento e saída



Fonte: elaborada pelo autor (2025).

7.3.1 Entradas

Para garantir a utilidade prática do artefato, é essencial que os resultados gerados sejam consistentes, claros e fáceis de interpretar. Esta seção descreve as saídas esperadas do artefato, destacando como as informações fornecidas pelo usuário são transformadas em relatórios estruturados que auxiliam professores na avaliação de evidências e indícios de pensamento computacional em soluções de programação.

O artefato desenvolvido opera a partir de duas entradas fundamentais: que pode ser

tanto formulado e proposto pelo professor quanto selecionado dentre os desafios previamente cadastrados no próprio artefato, e o código desenvolvido pelo estudante como solução para esse desafio.

De forma resumida, as entradas processadas pelo artefato são:

1. **Descrição Geral do Desafio:** Fornecida pelo professor, essa entrada é utilizada para a criação de uma documentação completa, contendo passos esperados e exemplos que ilustram o problema proposto.
2. **Arquivo** .py **com a Solução do Aluno:** O código enviado é analisado por uma LLM, que identifica a presença ou ausência dos indícios previamente definidos para cada evidência avaliada.

7.3.2 Processamento

Após o recebimento das informações, o artefato utiliza um *Large Language Model* (LLM) para gerar automaticamente um documento de referência, que descreve de maneira detalhada a solução ideal para o problema apresentado. Esse documento serve como parâmetro para a avaliação subsequente, permitindo a comparação sistemática entre a abordagem ideal e as evidências de PC presentes no código submetido pelo estudante.

Essa geração baseia-se na descrição detalhada do desafio, permitindo que o LLM compreenda os critérios e as expectativas associadas à resolução. O documento resultante contém uma exposição minuciosa da abordagem ideal, articulando os passos, as estratégias e as estruturas de código que melhor atendem aos requisitos do desafio. Essa solução de referência serve, então, como um guia orientador para a avaliação das evidências de PC manifestadas no código submetido pelo estudante.

A utilização do LLM nesse contexto agiliza o processo de comparação entre a solução desenvolvida e a solução ideal. Em última instância, essa abordagem visa facilitar o trabalho do professor, promovendo uma análise semi-automatizada que complementa a avaliação qualitativa e quantitativa dos elementos do PC, contribuindo para um acompanhamento do desempenho dos estudantes.

O documento gerado pelo LLM contém os seguintes elementos:

- **Casos de Uso:** Exemplos de situações práticas onde o desafio pode ser aplicado, com a descrição das funcionalidades esperadas e cenários de uso detalhados.
- **Lista de Requisitos do Desafio:** Requisitos essenciais que os estudantes devem considerar

para resolver o problema.

- **Exemplos de Entrada e Saída Esperados:** Dados de entrada e os resultados esperados associados a esses dados.
- **Sequência de Passos Esperada para a Solução:** Uma sequência lógica de passos que a solução deve seguir para resolver o problema.
- **Nível de Detalhamento Esperado para as Instruções:** Orientações sobre o nível de detalhamento esperado no código.
- **Critérios de Completude:** Diretrizes que definem o que constitui uma solução completa, incluindo todas as partes necessárias do desafio.
- **Exemplos de Uso Correto de Estruturas de Condição e Repetição:** Exemplos práticos do uso correto de estruturas como loops (for, while) e condicionais (if-else).
- **Exemplos de Controle de Repetições e Condições:** Estratégias para controlar eficientemente a execução de loops e condições.

Após a geração do documento pelo LLM, o artefato procede com uma comparação entre o código fornecido pelo estudante e o documento de referência gerado. Essa análise é realizada internamente, por meio de algoritmos que mapeiam as evidências presentes no código contra os critérios estabelecidos no documento, permitindo assim identificar a correspondência e as discrepâncias entre ambos.

7.3.3 Saídas

Com base nessa análise comparativa, o artefato fornece ao professor uma sugestão de avaliação para todos os indícios previamente selecionados para a mensuração do desenvolvimento do PC. Essa sugestão de avaliação abrange tanto aspectos quantitativos, quanto feedbacks qualitativos, que descrevem as evidências encontradas e destacam pontos de melhoria. Dessa forma, o artefato contribui para padronizar a avaliação, tornando o processo não apenas mais objetivo, mas também capaz de oferecer um retorno detalhado e fundamentado, o que facilita a intervenção pedagógica de maneira eficaz e individualizada.

O artefato gera três tipos principais de saídas para cada indício identificado, visando oferecer uma análise aprofundada, prática e de fácil interpretação para o professor. Tais saídas permitem correlacionar diretamente as evidências de PC com o código produzido pelos estudantes, bem como sugerir classificações objetivas que apoiem o processo avaliativo:

1. **Descrição Textual:** O artefato apresenta uma análise clara e fundamentada, apontando se o

estudante atendeu ou não ao indício em questão. Essa descrição explica de forma detalhada as razões pelas quais o indício foi considerado presente (ou não), oferecendo exemplos concretos ou contraexemplos que ilustram o nível de domínio evidenciado. Tal recurso fornece ao professor subsídios para entender as escolhas do aluno e planejar intervenções pedagógicas direcionadas.

Os relatórios gerados apresentam uma estrutura padronizada, com campos bem definidos, que incluem:

- Definição do Problema: Descrição detalhada do desafio proposto.
- Casos de Uso e Lista de Requisitos: Exposição dos principais requisitos e possíveis casos de uso.
- Exemplos de Entrada e Saída: Exemplos que esclarecem os resultados esperados.
- Sequência de Passos: Instruções detalhadas sobre a resolução do desafio.
- Critérios de Completude: Indicadores que determinam o nível de finalização do código.
- Análise do Código do Aluno: Observações claras, incluindo o quantitativo de funções implementadas, trechos destacados de código e sugestões de melhorias.

Essa padronização visa eliminar ambiguidades e fornecer um retorno estruturado ao professor, facilitando o acompanhamento do desempenho do aluno.

2. **Apontamento e Anotação no Código:** Os trechos do código desenvolvidos pelo estudante e associados ao indício avaliado são destacados e anotados. Esse procedimento permite ao professor visualizar, de maneira imediata, onde exatamente o raciocínio computacional se manifesta no programa — ou onde ele deveria ter se manifestado. Por meio dessa correlação direta entre a teoria (indício de PC) e a prática (trechos de código), o acompanhamento do progresso e a identificação de pontos de melhoria tornam-se mais objetivos e ágeis.
3. **Sugestão de Conceito:** Para cada indício analisado, o artefato propõe um conceito classificatório, atribuindo um grau de proficiência de acordo com as evidências encontradas no código e na análise qualitativa. A escala utilizada segue a seguinte classificação:
   - Não se aplica
   - Nenhum
   - Baixo
   - Regular
   - Bom

– Ótimo

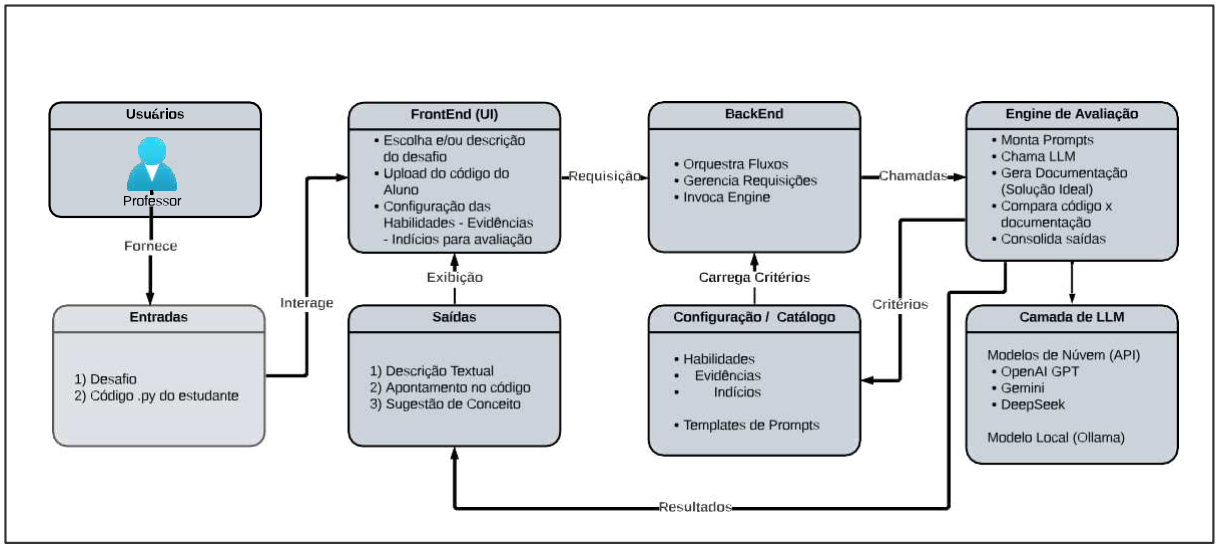
Essa categorização fornece um referencial objetivo para comparar e acompanhar a evolução dos estudantes ao longo do tempo. Além disso, facilita a elaboração de estatísticas e relatórios que podem subsidiar tanto o planejamento pedagógico quanto a avaliação em larga escala.

Em conjunto, esses três tipos de saída permitem uma avaliação consistente e fundamentada do desenvolvimento do PC, ao mesmo tempo que promovem *feedback* individualizado e pontual para cada estudante. Vale ressaltar que o professor possui autonomia para selecionar quais indícios, evidências e habilidades deseja incluir na avaliação, adequando o processo às particularidades de cada turma, disciplina ou instituição de ensino. Tal flexibilidade torna o artefato uma ferramenta versátil, capaz de se adaptar a diferentes cenários.

7.4 Arquitetura Técnica do Artefato

Esta seção detalha a arquitetura técnica do artefato desenvolvido. O sistema foi concebido para integrar dados de configuração (habilidades, evidências e indícios) e técnicas de Processamento de Linguagem Natural (PLN), por meio de modelos de linguagem de grande porte (LLM), de forma a automatizar análises.

Figura 30 – Arquitetura do artefato



Fonte: elaborada pelo autor (2025).

A Figura 30 apresenta uma visão geral da arquitetura do artefato, destacando os principais componentes e o fluxo de interação entre eles. A estrutura contempla desde as entradas fornecidas por professores e estudantes até o processamento realizado pelo *FrontEnd*, *BackEnd*

e *Engine de Avaliação*, que interagem com diferentes modelos de linguagem. Os resultados consolidados são exibidos em forma de descrições textuais, apontamentos no código e sugestões de conceito. Nas seções seguintes, os principais módulos serão detalhados de forma a esclarecer os aspectos técnicos e operacionais que sustentam o funcionamento do artefato.

7.4.1 Classe Engine

A classe Engine constitui o núcleo operacional do artefato, sendo responsável por orquestrar toda a lógica de interação entre os dados de entrada do processo avaliativo e os LLM integrados ao sistema. Seu propósito principal é montar os prompts, administrar o contexto da análise e processar as respostas devolvidas pelos modelos, viabilizando a avaliação semi-automatizada do PC.

7.4.1.1 Atributos Internos

Os principais atributos mantidos pela classe Engine são os seguintes:

- ai: instância do modelo LLM utilizado, implementando a interface AIInterface.
- dimension: objeto contendo dados da habilidade do PC atualmente em avaliação.
- evidence: objeto contendo dados da evidência associada à habilidade corrente.
- clue: objeto contendo dados do indício (clue) específico a ser avaliado.
- code: código-fonte submetido pelo estudante.
- documentation: documentação técnica gerada automaticamente pela LLM, a partir do contexto geral do desafio.
- clue_key: identificador da chave do indício atualmente em análise.

7.4.1.2 Geração de Documentação

O método make_documentation tem por finalidade gerar, via LLM, a documentação técnica que orientará a avaliação do desafio proposto. O processo ocorre da seguinte forma:

1. O histórico de mensagens do modelo LLM é reinicializado.
2. O contexto geral do desafio, fornecido pelo usuário, é adicionado ao prompt como mensagem do tipo user.
3. Um prompt adicional é enviado como system, solicitando a geração de um documento

estruturado.

4. Um arquivo Markdown contendo um template padrão é carregado e anexado ao prompt enviado à LLM. Este arquivo encontra-se no caminho ./../../md/prompt/teacher_documentation
5. O método chat() da LLM é então chamado para gerar o conteúdo da documentação.

O resultado gerado é armazenado e posteriormente utilizado no restante do processo de avaliação.

7.4.1.3 Atualização do Contexto de Avaliação

O método __update_ai_context é utilizado para preparar todo o contexto necessário que será enviado à LLM antes de cada avaliação de indício. Esse contexto inclui, se disponíveis:

- O código-fonte submetido pelo aluno, precedido de uma mensagem explicativa.
- O documento técnico gerado anteriormente, enviado como mensagem do tipo user.
- Dados sobre a dimensão e a evidência em análise, cada um incluído como mensagem do tipo system.
- Informações detalhadas sobre o indício específico, incluindo:
  - O título do indício.
  - O texto de detecção, explicando como identificar a presença do indício no código.
  - O conteúdo do campo output, que descreve as saídas esperadas da análise, incluindo texto, trechos de código e a nota atribuída ao aluno.

O prompt gerado para um indício segue o padrão ilustrado na Listagem 1.

Código-fonte 1 – Estrutura do prompt gerado para análise de um indicio.

```
Indicio <clue_key>: <titulo do indicio>

Deteccao: <texto explicativo de como detectar o indicio>

Saidas: (O que deve retornar como saida os pontos de 1 a 3...)

Atencao: Para a saida "1. Texto: ..." pode ser a negativa
do texto exemplo, caso o aluno nao tenha satisfeito o indicio proposto.

<conteudo do campo output>
```

7.4.1.4 Configuração do Estado da Avaliação

A classe Engine oferece métodos para configurar progressivamente o estado da avaliação:

- set_dimension(dimension): define a habilidade atualmente em análise e atualiza o contexto da LLM.
- set_evidence(evidence_key): define a evidência a ser avaliada dentro da habilidade selecionada.
- set_clue(clue_key): define o indício específico a ser analisado.

Esses métodos garantem que o prompt construído para a LLM contenha todas as informações necessárias para uma avaliação precisa.

7.4.1.5 Execução da Análise

Por fim, o método output realiza a requisição final ao modelo LLM. Esse método chama chat() para processar o contexto montado, retornando o texto gerado, que inclui:

- A análise textual sobre o indício avaliado.
- A marcação de trechos relevantes no código do aluno.
- A sugestão de nota para o indício, conforme critérios definidos nos arquivos de configuração.

Opcionalmente, o resultado pode ser:

- Exibido no console, através do método show() do modelo LLM.
- Persistido em arquivo Markdown, em diretório dedicado aos resultados (./../../md/output/).

Dessa forma, a classe Engine representa a peça central do artefato, encapsulando toda a lógica de transformação dos dados, promovendo a integração entre a base de conhecimento definida nos arquivos de configuração e a capacidade de interpretação textual provida pelas LLM.

7.4.2 Modelos LLM Suportados

O artefato foi desenvolvido para operar de forma modular com diferentes modelos de linguagem, implementados por meio de classes específicas que seguem a interface comum AIInterface. Atualmente, estão disponíveis as seguintes integrações:

- OpenAI GPT-4o
- DeepSeek

- Gemini
- Ollama, para execução local via API.

Como exemplo, a Listagem 2 apresenta o código utilizado para integração do artefato com o modelo ChatGPT (GPT-4o) da OpenAI, conforme implementado no artefato.

Código-fonte 2 – Exemplo de integração do artefato com a API ChatGPT (OpenAI GPT-4o).

```
// Cliente configurado em src/core/clients/open_ai.py
import os
from openai import OpenAI

API_KEY = os.getenv('OPENAI_API_KEY')

OpenaiClient = OpenAI(
  api_key=API_KEY
)
__all__ = ['OpenaiClient']

// Modelo implementado em src/core/ai/models/openai_model.py

from .base_model import BaseModel
from ..clients.open_ai import OpenaiClient

class OpenaiModel(BaseModel):
  def __init__(self):
    super().____init____()
    self.client = OpenaiClient

  def chat(self):
    self.completion = self.client.chat.completions.create(
      model="gpt-4o",
      messages=self.messages,
    )
    output = self.completion.choices[0].message.content
    return output

  def show(self):
    for choice in self.completion.choices:
      print(choice.message.content)
```

No artefato, a classe OpenaiModel é responsável por encapsular a chamada ao endpoint de chat da OpenAI, utilizando o modelo gpt-4o. O método chat() envia uma lista de mensagens armazenadas em self.messages e retorna a resposta gerada pela LLM.

7.4.3 Fluxo Técnico de Processamento

O fluxo técnico executado pelo artefato compreende as seguintes etapas:

1. Definição, pelo usuário, de parâmetros de entrada:
   - Contexto geral do desafio.
   - Código-fonte do aluno.
   - Habilidades, evidências e indícios a serem avaliados.
   - Modelo LLM a ser utilizado.
2. Geração da documentação técnica pelo método make_documentation().
3. Para cada habilidade, evidência e indício definidos:
   a) Atualização do contexto da LLM via __update_ai_context.
   b) Execução da análise pelo método output().
   c) Armazenamento e eventual persistência dos resultados, incluindo:
      - Texto descritivo gerado pela LLM.
      - Trechos de código identificados.
      - Nota atribuída ao indício.

7.5 Integração com o Backend

A integração do artefato com a API backend é implementada por meio da função process_challenge_submission(). Esta função centraliza a lógica de execução do fluxo completo, conforme ilustrado na Código Fonte 3.

Código-fonte 3 – Trecho do backend que integra o artefato ao endpoint da API.

```
engine = Engine(ai_model)
documentation = engine.make_documentation(general_context)
engine.inputs(code=code_text, documentation=documentation)

for dimension in dimensions:
    ---
    for evidence in dimension.evidences:
        ---
        for clue in evidence.clues:
            engine.set_clue(clue.key)
            output = engine.output()
```

7.6 Infraestrutura do Artefato

A infraestrutura necessária para a operação do artefato varia conforme o LLM escolhido, sendo possível tanto o uso de serviços baseados em nuvem quanto a execução local, dependendo das características de cada provedor.

7.6.1 Modelos Baseados em Nuvem

Para os modelos **GPT (OpenAI)**, **Gemini (Google)** e **DeepSeek**, a infraestrutura utilizada corresponde à própria infraestrutura das empresas provedoras desses serviços. Nessas integrações, o artefato realiza chamadas diretas às APIs públicas disponibilizadas por essas plataformas. Assim, toda a capacidade computacional necessária para o processamento das requisições — incluindo o treinamento, a execução dos modelos e a manutenção de alta disponibilidade — está a cargo das respectivas empresas fornecedoras.

Essa arquitetura baseada em nuvem apresenta diversas vantagens, entre as quais se destacam:

- **Escalabilidade automática**: capacidade de atender grandes volumes de requisições sem necessidade de dimensionamento manual de servidores.
- **Manutenção reduzida**: não há necessidade de gerenciamento direto de infraestrutura local para processamento dos modelos.
- **Atualizações contínuas**: os provedores mantêm os modelos atualizados com as últimas evoluções tecnológicas, sem intervenção do usuário final.

Apesar dessas facilidades, é importante observar que o uso das APIs públicas implica custos variáveis baseados no volume de requisições, podendo se tornar elevado em aplicações de larga escala. Entretanto, em cenários de **baixa utilização**, os custos tendem a ser previsíveis e muitas vezes menores do que manter uma infraestrutura própria em nuvem dedicada à execução local de modelos.

7.6.2 Modelo Executado Localmente (Ollama)

O modelo **Ollama** apresenta um cenário distinto. Para utilizá-lo, é necessário instalar o serviço em um servidor local ou em um ambiente de nuvem configurado pelo próprio usuário ou pela instituição interessada. O artefato conecta-se ao servidor Ollama através de requisições HTTP, utilizando *endpoints* específicos disponibilizados pela API local do Ollama.

A operação do Ollama envolve algumas particularidades importantes:

- Assim como a maioria das LLM **o Ollama exige hardware robusto, incluindo *Graphics Processing Unit* (GPU)**, para garantir desempenho adequado e tempos de resposta aceitáveis.
- Quando o Ollama é executado **localmente**, pode representar uma solução de menor custo para instituições que já possuem a infraestrutura necessária, sobretudo em ambientes onde o volume de requisições é elevado.
- Entretanto, se a opção for executar o Ollama em ambiente de nuvem — por exemplo, na AWS — **em cenários de baixa utilização**, o custo operacional geralmente se torna **superior ao custo de simplesmente consumir as API públicas dos provedores GPT, Gemini ou DeepSeek**. Isso ocorre porque, mesmo sem uso intenso, é preciso manter servidores provisionados (incluindo máquinas virtuais com GPU), o que gera custos fixos relativamente altos, independentemente do volume de chamadas efetuadas.

Portanto, do ponto de vista da infraestrutura, a decisão de utilizar Ollama em nuvem só se justifica financeiramente em contextos de uso constante e volumoso, onde os custos variáveis das APIs públicas superariam os custos fixos de manter a infraestrutura própria. Para cenários de baixo uso, **as API públicas frequentemente representam a solução mais econômica e prática**.

A definição do modelo LLM e, consequentemente, da infraestrutura utilizada, deve considerar aspectos como:

- Custos de operação, tanto em nuvem quanto em servidores locais.
- Políticas de privacidade e segurança de dados.
- Exigências de desempenho e escalabilidade.
- Capacidade técnica da equipe responsável pela manutenção do ambiente.

Dessa forma, o artefato foi projetado para se manter flexível, podendo ser integrado tanto a modelos em nuvem quanto a soluções locais, atendendo a diferentes cenários institucionais e requisitos específicos do projeto.

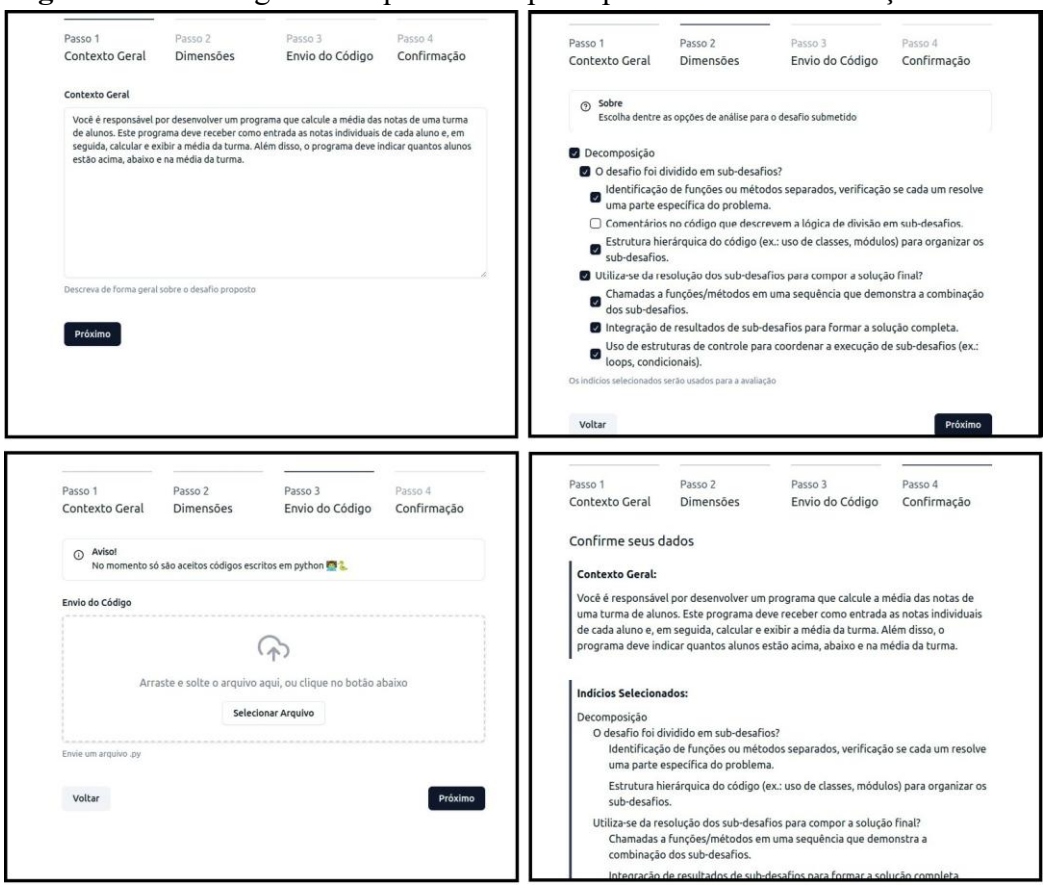
7.7 Fluxo de Interação

A experiência do usuário é um componente crítico no sucesso de qualquer ferramenta computacional. Nesta seção, detalha-se o fluxo de interação proposto para a utilização do artefato, dividido em etapas lógicas que visam simplificar o uso e garantir a eficiência do processo.

Cada etapa é descrita de forma a mostrar como os usuários podem aproveitar plenamente as funcionalidades oferecidas pelo artefato.

O fluxo de interação com o artefato é organizado em quatro etapas principais, conforme ilustrado na Figura 31:

Figura 31 – Visão geral das quatro telas principais do fluxo de interação do artefato

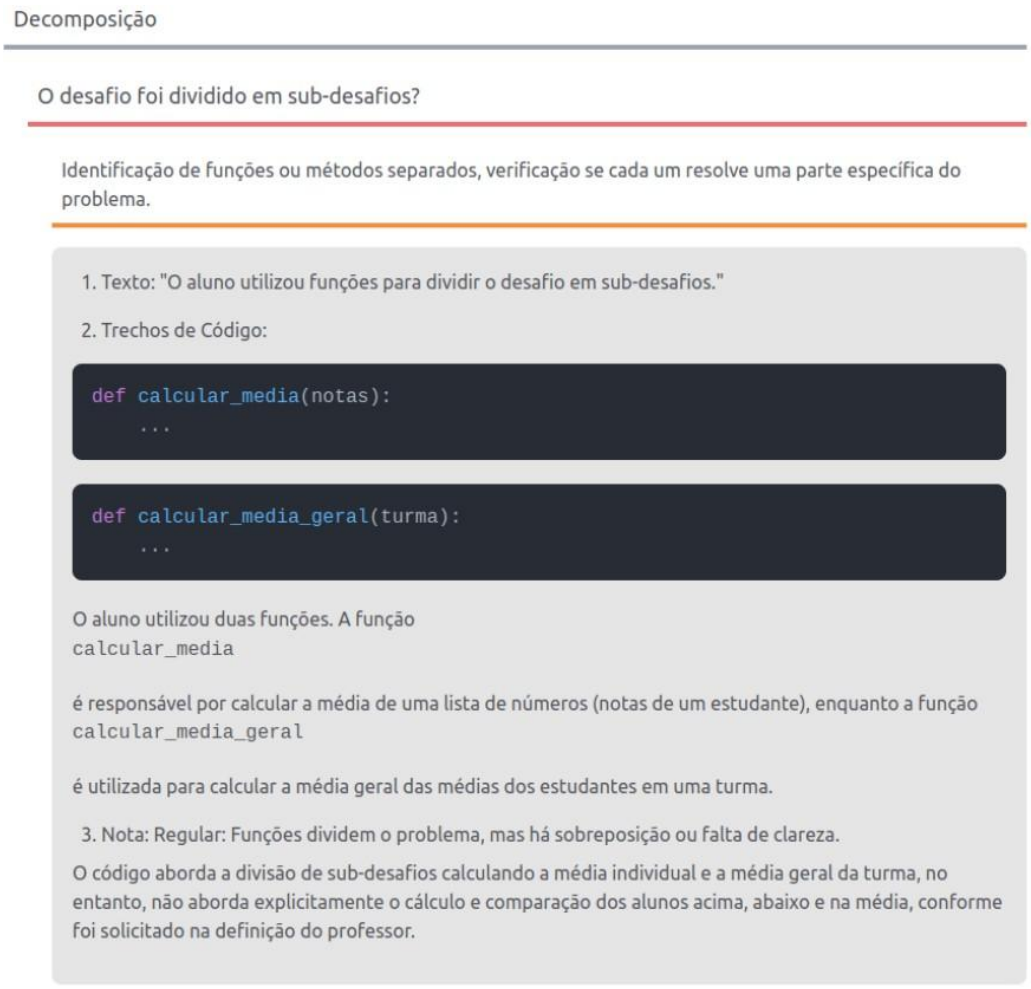


Fonte: *print* da tela do artefato (2025).

1. **Contexto Geral:** Nesta etapa, o docente insere uma descrição abrangente do desafio a ser avaliado ou seleciona um dos desafios disponíveis no artefato, fornecendo os elementos necessários para a contextualização da problemática abordada.
2. **Seleção de Habilidades, Evidências e Indícios:** Através de uma interface baseada em um sistema de *checklist* aninhado, o usuário seleciona as habilidades, evidências e indícios que serão considerados durante a avaliação, assegurando uma análise detalhada e criteriosa.
3. **Envio do Código:** O docente ou discente anexa um arquivo com extensão .py que contém a implementação da solução para o desafio proposto, possibilitando a análise semi-automatizada do código-fonte.
4. **Confirmação dos Dados:** O sistema exibe um resumo das informações fornecidas,

permitindo que o usuário revise e ajuste os dados antes da geração do relatório final, garantindo a integridade e a fidelidade dos registros.

Figura 32 – Saída do artefato



Fonte: *print* da tela do artefacto (2025).

Após a confirmação, a aplicação processa os dados, consulta a LLM configurada através do padrão *Adapter* (GAMMA, 1995) e gera um relatório detalhado em formato *Markdown*. Como ilustrado na Figura 32, esse relatório inclui:

- Descrição detalhada e estruturada do desafio, apresentando a problemática de forma clara e organizada.
- Análise das habilidades, evidências e indícios, com detalhamento dos elementos identificados — ou a ausência destes — na solução submetida pelo estudante.
- Marcação do trecho do código em que se evidencia aquele indício específico na solução apresentada.
- Atribuição de níveis de proficiência, que categorizam o desempenho do aluno com base na

análise dos indícios identificados.

O fluxo proposto assegura uma experiência de avaliação eficiente e sistemática, promovendo uma análise rigorosa e padronizada do pensamento computacional dos estudantes.

7.8 Exemplo Condutor

Com o objetivo de ilustrar de forma prática o funcionamento do artefato desenvolvido, esta seção apresenta um exemplo condutor que percorre as etapas do fluxo de avaliação do PC.

De forma intencional, foi selecionado um desafio de baixa complexidade, acompanhado da escolha de avaliar apenas um indício específico. Essa estratégia visa facilitar a compreensão de quem lê este trabalho, permitindo que o leitor acompanhe, de forma clara e objetiva, como o artefato gera a saída para o processo de avaliação. A simplicidade do código, aliada ao foco em um único indício do modelo hierárquico de PC, evita sobreposição de informações e possibilita uma visualização precisa e direta da relação entre o código analisado, os critérios avaliativos e o resultado gerado pelo artefato.

7.8.1 Desafio Proposto

Para exemplificar o funcionamento do artefato de avaliação semi-automatizada, foi proposto um desafio simples, porém representativo, que permite demonstrar com clareza o fluxo de análise, a identificação dos indícios e a geração das respectivas saídas. A escolha de um problema de baixa complexidade visa não apenas simplificar a leitura e a compreensão do leitor, mas também permitir que a dinâmica do artefato seja visualizada de forma objetiva, sem ruídos decorrentes de códigos extensos ou múltiplas camadas de avaliação.

O desafio selecionado consiste na implementação de um programa que simule um cofre digital, conforme especificado a seguir:

Enunciado do Desafio:

Implemente um programa que simule um cofre digital com as seguintes funcionalidades:

- O usuário pode realizar depósitos quantas vezes desejar.
- Valores iguais a zero ou negativos devem ser ignorados pelo sistema.
- Quando o usuário digitar o comando “sair”, o programa deve exibir:
  - O número total de depósitos realizados.

– O valor total acumulado no cofre.

7.8.2 Código de Entrada

A Figura 33 apresenta o código submetido pelo estudante para solução do desafio proposto.

Figura 33 – Entrada do artefato — código desenvolvido pelo estudante

```
1   cofre = 0
2   deposito = 0
3
4   while (True):
5       valor = input("\nQuanto depositar ou sair: ")
6
7       if (valor.isnumeric()):
8           valor = int(valor)
9
10          if (valor > 0):
11              cofre += float(valor)
12              deposito += 1
13
14          else:
15              print("Nada depositado!\n")
16      else:
17          if (valor == "sair"):
18              print(f"valor depósitado: R${cofre:.2f}")
19              print(f"Depósitos: {deposito}")
20              break
21          else:
22              print("Valor inválido!\n")
```

Fonte: *print* da interface do artefato (2025).

7.8.2.1 Modelo LLM Utilizado

Para este exemplo, a avaliação foi realizada utilizando o modelo **ChatGPT**, configurado como *backend* do artefato para análise do código-fonte.

7.8.3 Configurações de Avaliação no Artefato

O processo de avaliação foi conduzido considerando os seguintes parâmetros, alinhados ao modelo hierárquico das habilidades do Pensamento Computacional:

– **Habilidade:** Abstração

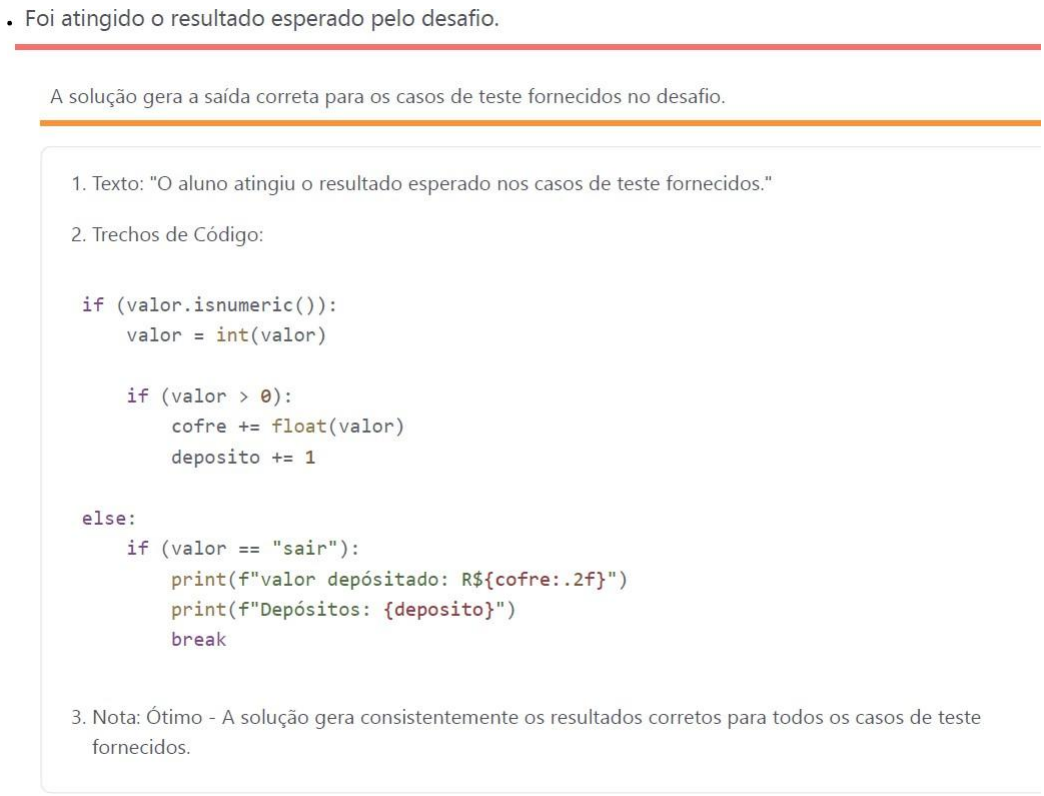
– **Evidência 3:** Foi atingido o resultado esperado pelo desafio.

– **Indício 3.1:** A solução gera a saída correta para os casos de teste fornecidos.

7.8.4 Saída Gerada pelo Artefato

A Figura 34 exibe a tela com a saída gerada pelo artefato, correspondente à análise do indício selecionado. Nesta etapa, o sistema avalia se a solução implementada produz os resultados esperados para os casos de teste previamente definidos no enunciado do desafio.

Figura 34 – Exemplo de saída do artefato



Fonte: *print* da interface do artefato (2025).

O artefato apresentou os seguintes resultados para este cenário:

1. Confirmou que o estudante atingiu o **resultado esperado**, produzindo as saídas corretas para os casos de teste fornecidos.
2. Destacou o **trecho específico do código** que estão diretamente associados à geração da saída correta, contribuindo para a rastreabilidade da avaliação.
3. Atribuiu uma **nota correspondente** ao desempenho do estudante no âmbito do indício avaliado, conforme os critérios previamente definidos na modelagem da habilidade de Abstração.

7.9 Considerações Finais

Este capítulo apresentou a concepção, os fundamentos e o desenvolvimento do artefato computacional proposto para a avaliação semi-automatizada do desenvolvimento do PC. A abordagem avaliativa e o artefato foram concebida com o objetivo de superar as limitações observadas em soluções anteriores, especialmente no que diz respeito à ausência de padronização, à dependência excessiva do conhecimento técnico do avaliador e à dificuldade de escalabilidade dos processos avaliativos.

A arquitetura técnica do artefato foi projetada de forma modular e extensível, permitindo:

- Adaptação a diferentes modelos LLM.
- Facilidade de manutenção e evolução do sistema.
- Padronização e confiabilidade nas avaliações do Pensamento Computacional.

A adoção de uma arquitetura modular, combinando tecnologias robustas e atualizadas tanto no backend quanto no frontend, bem como a integração com modelos de linguagem de grande porte (LLM), conferiu ao artefato a capacidade de realizar análises sofisticadas, mantendo elevada flexibilidade e escalabilidade. O uso do padrão de projeto *Adapter* garante a independência tecnológica e a evolução contínua do sistema, permitindo a incorporação de novos modelos de IA e de funcionalidades futuras.

Além dos aspectos técnicos, a abordagem avaliativa destaca-se por sua aderência ao modelo hierárquico de avaliação do PC, estruturado em habilidades, evidências e indícios, o que permite uma análise granular e criteriosa das competências desenvolvidas pelos estudantes. A geração de documentos de referência, a análise estruturada dos códigos-fonte e a produção de relatórios detalhados — contendo *feedbacks* qualitativos, apontamentos diretos no código e classificações quantitativas — configuram uma solução inovadora, alinhada às necessidades pedagógicas contemporâneas.

No capítulo seguinte, são apresentados os procedimentos metodológicos adotados para a avaliação empírica do artefato.

8 REVISÃO POR ESPECIALISTAS

8.1 Análises

A avaliação por especialistas constitui uma etapa essencial dentro da metodologia DSR (DRESCH *et al.*, 2020), permitindo avaliar a eficácia e a aplicabilidade do artefato desenvolvido. O processo seguiu quatro etapas principais: testes com o artefato, gravação das entrevistas, análise e tratamento dos dados e análise de conteúdo, conforme descrito a seguir.

8.1.1 Perfil dos Especialistas

8.1.1.0.1 Especialista 1

Docente de uma universidade pública estadual desde 2008, com mais de 17 anos de experiência no magistério superior. É graduado em Ciência da Computação e possui título de mestre em Administração, com ênfase em Competitividade e Relações Interorganizacionais. Atualmente ministra disciplinas nas áreas de Lógica Matemática, Matemática Discreta e Introdução à Ciência de Dados, além de orientar Trabalhos de Conclusão de Curso.

8.1.1.0.2 Especialista 2

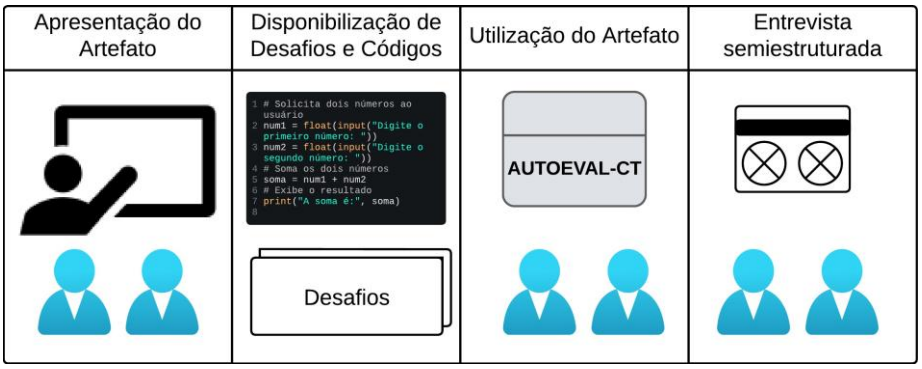
Docente de uma universidade pública estadual desde 2006, com ampla experiência no ensino superior. É graduado em Ciência da Computação e mestre em Ciência da Computação, título obtido em 2009. Atua em disciplinas da área de programação e computação, contribuindo para a formação de estudantes e para a avaliação de competências relacionadas ao pensamento computacional.

8.1.2 Processo 1: Testes com o Artefato

Nesta etapa, dois especialistas em pensamento computacional e ensino de programação foram introduzidos ao artefato desenvolvido. Inicialmente, foi apresentada uma visão geral sobre o funcionamento do artefato, destacando suas funcionalidades e aplicações pedagógicas. Para a análise prática, foram disponibilizados desafios de programação e códigos reais do estudantes, permitindo que os especialistas testassem o artefato em um ambiente autêntico de uso. Além disso, os professores tiveram a liberdade de aplicar o artefato utilizando seus próprios

desafios e códigos, possibilitando uma avaliação mais personalizada e abrangente. A Figura 35 apresenta o processo realizado para a revisão.

Figura 35 – Processo de revisão por especialistas



Fonte: elaborada pelo autor (2025).

Cabe ressaltar que, para a revisão, foi implementada exclusivamente a dimensão de Decomposição do PC.

8.1.3 Processo 2: Gravação das Entrevistas

Com o objetivo de documentar as percepções dos especialistas, foram conduzidas entrevistas individuais gravadas em ambiente acadêmico. As gravações ocorreram na Universidade Estadual Vale do Acaraú (UVA) – Campus CIDAO, no contexto do Curso de Ciência da Computação.

O primeiro especialista foi entrevistado em 04 de dezembro de 2024, enquanto a segunda entrevista ocorreu em 23 de janeiro de 2025. Os áudios completos das entrevistas estão disponíveis no seguinte link: Acesso às entrevistas.

8.1.4 Processo 3: Análise e Tratamento dos Dados

Para assegurar a integridade e qualidade da análise dos dados coletados, foi seguido um processo de pré-processamento composto pelas seguintes etapas:

1. **Transcrição das Entrevistas**: As gravações foram transcritas utilizando a ferramenta de transcrição embutida no Microsoft Word (Office 365), garantindo precisão na conversão do áudio em texto. A transcrição completa pode ser acessada no seguinte link: Acesso à transcrição.
2. **Higienização dos Dados**: A transcrição passou por um processo de revisão e edição

para remover ruídos linguísticos, expressões redundantes e ajustar a estrutura textual sem comprometer o conteúdo original das entrevistas. A versão higienizada da transcrição pode ser acessada no link: Acesso à transcrição revisada.

8.1.5 Processo 4: Análise de Conteúdo

Para a interpretação dos dados qualitativos, foi empregada a metodologia de Análise de Conteúdo proposta por Laurence Bardin (BARR; STEPHENSON, 2011a), seguindo as seguintes fases:

1. **Pré-Análise:** Nesta etapa inicial, foi realizada uma leitura flutuante do material transcrito, permitindo a identificação de temas recorrentes e a definição de categorias de análise.
2. **Exploração do Material:** O corpus de análise foi segmentado em unidades temáticas, classificadas de acordo com os eixos de investigação, como usabilidade, impacto no ensino, precisão da análise e sugestões de melhorias.
3. **Tratamento dos Resultados:** Os achados foram organizados e interpretados, permitindo a elaboração de inferências sobre o impacto do artefato na avaliação do pensamento computacional, bem como suas limitações e potencialidades.

A abordagem sistemática adotada garantiu que os resultados fossem analisados de maneira rigorosa e fundamentada, contribuindo para a compreensão do papel do artefato no apoio às práticas docentes no ensino de programação e PC.

8.2 Resultados

Conforme exposto na subseção 8.1.5, a análise de conteúdo foi conduzida em três etapas: Pré-Análise, Exploração do Material e Tratamento dos Resultados. Nesta seção, serão apresentados os resultados obtidos em cada uma dessas etapas.

8.3 Pré-Análise

Nesta etapa, realizamos uma leitura flutuante do material para identificar os principais temas e definir os critérios de categorização. As entrevistas fornecem informações sobre a percepção dos professores em relação ao artefato de avaliação do PC, abordando aspectos como usabilidade, impacto pedagógico e possíveis melhorias.

Os principais eixos de análise identificados foram:

- Relevância do artefato para a avaliação do PC;
- Facilidade de uso e experiência do usuário;
- Precisão da análise e adequação dos critérios de avaliação;
- Impacto no ensino e na formação dos estudantes;
- Sugestões para melhorias e funcionalidades adicionais.

Com base nesses eixos, definimos um esquema de codificação para a próxima etapa.

Exploração do Material

Nesta fase, extraímos unidades de significado e agrupamos as informações dentro dos eixos definidos:

(A) Relevância do Artefato

- Os entrevistados consideram o artefato essencial para auxiliar os professores na avaliação do pensamento computacional.
- Ela melhora a produtividade do professor, especialmente em turmas numerosas.
- Permite uma avaliação mais objetiva, ajudando a identificar padrões e aspectos que poderiam passar despercebidos na análise manual.

(B) Facilidade de Uso e Experiência do Usuário

- O artefato foi avaliada como intuitiva, com um fluxo bem estruturado.
- Há sugestões para melhorias na interface, como aprimoramento na experiência do usuário e um tutorial introdutório sobre pensamento computacional.
- A necessidade de conexão com a internet foi apontada como uma possível limitação.

(C) Precisão da Análise e Adequação dos Critérios

- A análise dos códigos foi considerada precisa e coerente.
- Alguns entrevistados mencionaram que a quantidade de critérios pode ser confusa inicialmente, sugerindo uma estrutura mais clara na apresentação dos resultados.
- A escolha da linguagem Python foi questionada, pois sua simplicidade pode mascarar a compreensão real de conceitos computacionais.

(D) Impacto no Ensino e Formação dos Estudantes

- O artefato pode melhorar a qualidade do *feedback* fornecido aos estudantes.
- Permite que os professores focalizem mais na orientação pedagógica do que na correção manual dos códigos.
- Se bem aplicada, pode promover um ensino mais sistemático e estruturado do pensamento computacional.

(E) Sugestões para Melhorias

- Inclusão de um tutorial introdutório sobre pensamento computacional para professores.
- Implementação de um relatório mais detalhado da análise do código.
- Acompanhamento da evolução do estudante ao longo do tempo.
- Adaptação do artefato para outras linguagens de programação que exijam maior estrutura lógica.

8.4 Tratamento dos Resultados

Com base na categorização dos dados, algumas interpretações gerais podem ser destacadas:

1. O artefato foi bem recebido pelos especialistas, contudo, necessita de refinamentos na apresentação dos critérios de avaliação e na experiência do usuário.
2. A principal contribuição do artefato reside na otimização do trabalho docente e na melhoria do *feedback* fornecido aos estudantes.
3. A implementação em larga escala pode demandar formação em serviço dos professores, a fim de maximizar seu uso e potencializar os benefícios pedagógicos.
4. Há um potencial significativo para a expansão do artefato para outras linguagens de programação e contextos educacionais.

As sugestões levantadas servem como diretrizes para aprimoramentos imediatos e futuros do artefato, garantindo sua adequação às necessidades pedagógicas dos professores e estudantes.

Melhorias Implementadas para as Próximas Etapas

Para aprimorar a eficiência e aplicabilidade do artefato, foram realizadas as seguintes melhorias:

- **Ajuste na engenharia de prompt**: aprimoramento no reconhecimento do uso de estruturas *built-in* durante as análises de decomposição.
- **Inclusão das dimensões restantes do PC**: implementação das categorias de Abstração, Pensamento Algorítmico e Reconhecimento de Padrões.
- **Aprimoramento da interface e da experiência do usuário (UI/UX)**: melhorias na usabilidade do artefato, tornando-a mais intuitiva e eficiente.

Para ampliar o impacto e a escalabilidade do artefato, algumas implementações futuras estão descritas na Seção 10.3.

9 AVALIAÇÕES EMPÍRICAS

Este capítulo apresenta os resultados obtidos na Etapa 4 - Experimentação, cuja condução seguiu a metodologia descrita na Seção 4.2.4. Os dados aqui expostos incluem as avaliações realizados com diferentes *Large Language Models* (LLM), a comparação entre a avaliação do artefato AutoEval-CT e a de um especialista humano, e, por fim, a avaliação em ambiente escolar conduzida em uma turma do Ensino Médio. Os resultados foram organizados em três seções.

9.1 Comparação de Desempenho entre Modelos de LLMs

Esta seção apresenta os resultados obtidos na comparação entre os modelos ChatGPT, Google Gemini e deepSeek, considerando o tempo de processamento necessário para realizar a **avaliação completa de cada código**, ou seja, a análise simultânea das quatro habilidades previstas no artefato AutoEval-CT. O objetivo foi mensurar o desempenho geral de cada LLM frente a códigos com diferentes níveis de complexidade (simples, intermediário e complexo), em condições controladas e reprodutíveis.

9.1.1 Considerações sobre a Amostra

Para assegurar a validade estatística dos resultados obtidos neste estudo, é essencial definir uma quantidade mínima adequada de códigos a serem avaliados. Optou-se por utilizar uma amostra de **10 códigos por nível de complexidade** — simples, intermediário e complexo — totalizando **30 códigos por modelo de LLM**. A decisão de utilizar uma amostra composta por 10 códigos para cada nível de complexidade foi orientada por critérios metodológicos e logísticos.

Essa classificação permitiu a construção de uma amostra heterogênea e representativa dos diferentes estágios de desenvolvimento do PC. A distinção por níveis também foi fundamental para analisar o desempenho dos modelos de LLM de forma mais granular, avaliando sua capacidade de identificar evidências específicas em diferentes contextos de dificuldade e estrutura algorítmica.

Para aumentar a robustez estatística das análises, adotou-se o **método de Monte Carlo com repetição** (ROBERT *et al.*, 1999; KALOS; WHITLOCK, 2009), no qual cada código foi avaliado **três vezes**, em momentos distintos. Conforme apontam (KALOS; WHITLOCK,

2009) e (ROBERT *et al.*, 1999) , a eficácia do método de Monte Carlo não está necessariamente atrelada ao tamanho absoluto da amostra, mas à capacidade de replicação controlada do experimento e à análise das distribuições geradas. Ao executar múltiplas repetições por código (neste caso, três execuções em diferentes horários), foi possível obter uma amostra estendida com **90 avaliações por modelo de LLM**, o que contribui para a mitigação de ruídos provenientes de variáveis externas, como instabilidade na API, variações no contexto de avaliação ou carga computacional nos servidores.

Os códigos utilizados foram selecionados com base em diferentes fontes e níveis de complexidade:

- **Códigos Simples:** Foram extraídos de produções reais de estudantes do Ensino Médio, participantes da avaliação em sala de aula. Esses códigos envolviam soluções diretas e com estrutura linear para problemas introdutórios, com uso restrito de estruturas condicionais simples e poucas repetições.
- **Códigos Intermediários:** Obtidos a partir de atividades desenvolvidas por estudantes do curso de Ciência da Computação durante a etapa de revisão do artefato. Esses códigos apresentavam maior organização estrutural, uso de funções, laços aninhados e manipulação de dados.
- **Códigos Complexos:** Foram selecionados manualmente em repositórios públicos. Os critérios de inclusão consideraram a presença de múltiplas estruturas computacionais (como condicionais encadeadas, funções recursivas, dicionários aninhados, manipulação de arquivos ou listas multidimensionais) e a capacidade dos códigos em responder a desafios com maior grau de complexidade, como jogos, simulações e resolução de problemas matemáticos.

O número total de códigos utilizados por LLM foi ($n = 30$), e o desvio padrão média das execuções foi de aproximadamente $\sigma = 3{,}55$ segundos. Considerando um nível de confiança de 95% ($Z = 1{,}96$), a margem de erro associada a essa amostra é dada por:

$$E = \frac{Z \times \sigma}{\sqrt{n}} = \frac{1{,}96 \times 3{,}55}{\sqrt{30}} \approx 1{,}27 \text{ segundos}$$

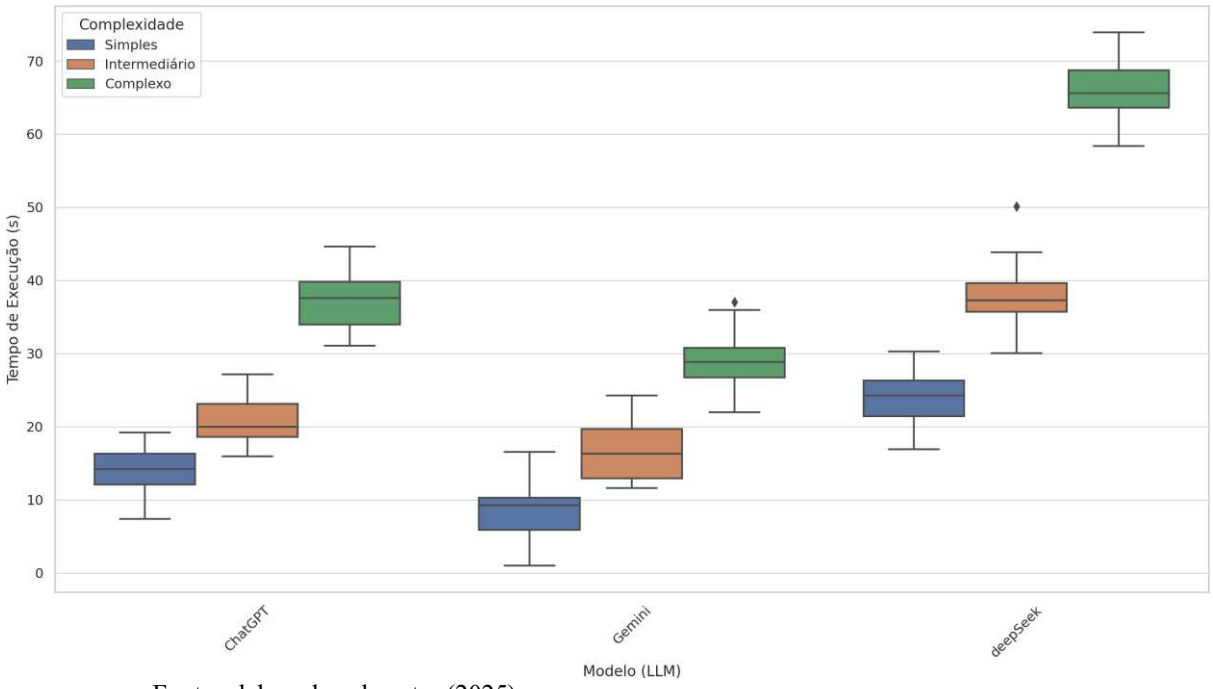
Consideramos esse valor satisfatório para os objetivos do estudo, representando um equilíbrio entre precisão estatística e viabilidade prática na aplicação e avaliação dos códigos. Portanto, embora se reconheça a limitação quanto à representatividade estatística, o delineamento adotado se mostra coerente com os objetivos do estudo, alinhado às práticas metodológicas

da área e suficientemente robusto para oferecer evidências relevantes sobre o desempenho do artefato proposto.

9.1.2 Resultados por LLM e Complexidade

A Figura 36 apresenta a distribuição dos tempos de execução dos LLM para diferentes níveis de complexidade dos códigos analisados. Como esperado, em geral, os códigos classificados como *complexos* tendem a demandar maior tempo de processamento, com medianas superiores quando comparadas aos códigos *simples*. Além disso, a dispersão dos tempos é mais acentuada nos códigos complexos, indicando maior variabilidade no desempenho dos modelos frente a desafios computacionais mais exigentes.

Figura 36 – Distribuição dos tempos de execução por LLM e complexidade

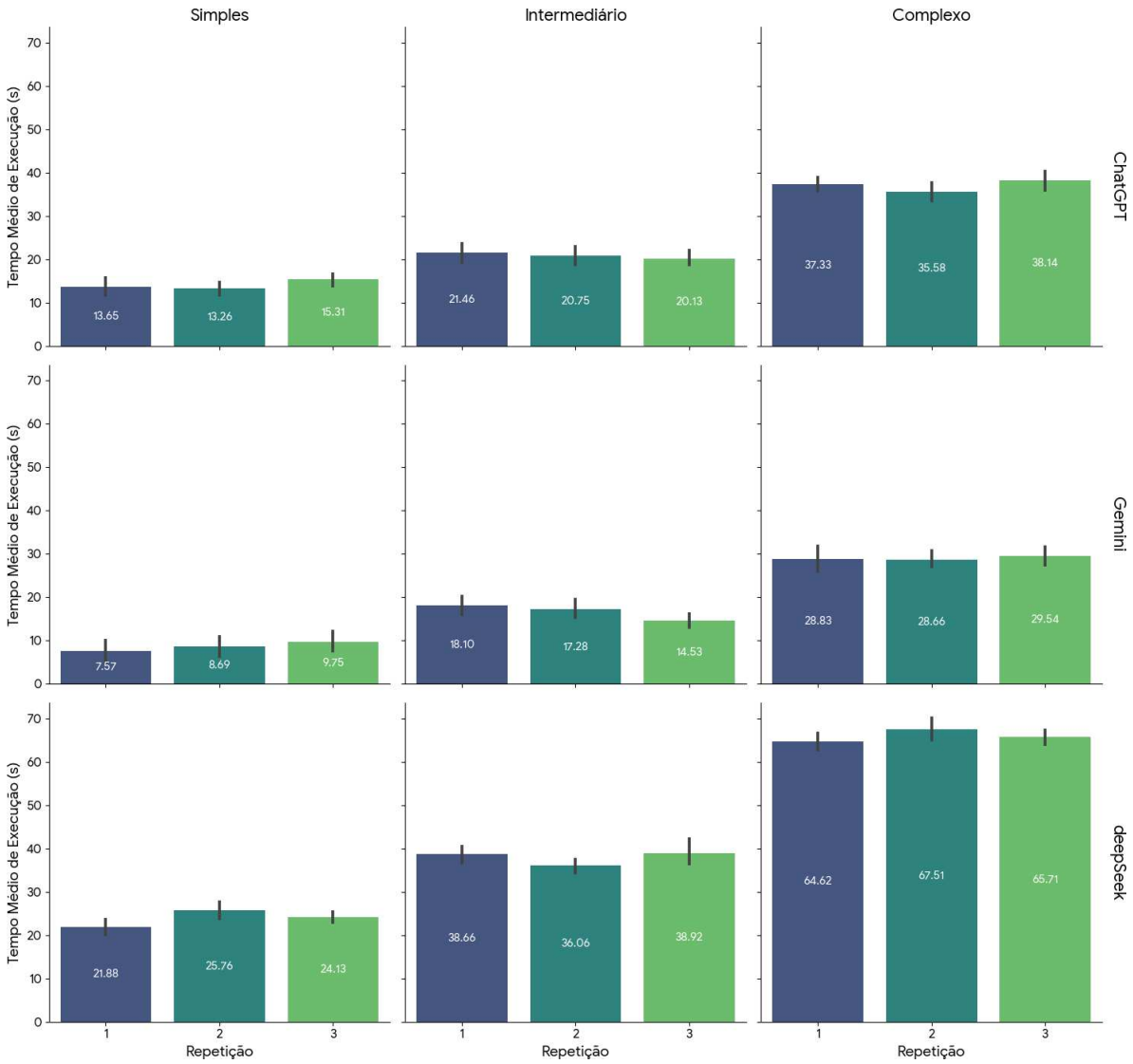


Fonte: elaborada pelo autor (2025).

A Figura 37 apresenta os tempos médios registrados para cada repetição, agrupados por modelo de LLM (linhas) e nível de complexidade dos códigos (colunas). Observa-se uma variação natural entre as repetições, refletindo flutuações reais de desempenho dos modelos em diferentes momentos de execução. Observa-se também que não houve discrepâncias significativas entre as médias obtidas nas três repetições, quando consideradas a mesma LLM e o mesmo nível de complexidade dos códigos. Esse resultado sugere estabilidade no comportamento dos modelos em condições controladas de execução. Apresentaremos uma análise mais aprofunda

na seção 9.1.3

Figura 37 – Tempo médio de execução por LLM, complexidade e repetição



Fonte: elaborada pelo autor (2025).

Os resultados demonstram que o **Google Gemini** apresentou os menores tempos de execução em todos os níveis de complexidade, sendo o modelo mais eficiente de forma consistente. O **ChatGPT** exibiu desempenho intermediário, com tempos superiores aos do Gemini, mas ainda aceitáveis para uso educacional. Por outro lado, o **deepSeek** apresentou os maiores tempos médios, com valores próximos ou superiores a 60 segundos na análise de códigos complexos.

Embora tenham sido observadas variações entre as repetições realizadas para um

mesmo código, decorrentes de fatores como flutuações na carga dos servidores e latência das APIs, a execução de 10 códigos distintos por categoria de complexidade permitiu que essas oscilações fossem suavizadas. Assim, as médias finais de tempo de execução por modelo e por nível de complexidade apresentaram uma boa representatividade do comportamento geral das LLMs, aproximando os valores e reduzindo o impacto de outliers pontuais.

Além disso, observou-se uma variabilidade mais acentuada entre as repetições do deepSeek, indicando possível instabilidade de desempenho. Em contrapartida, o Gemini demonstrou tempos mais consistentes entre execuções, o que o torna mais confiável para aplicações em ambientes que exigem maior responsividade.

9.1.3 Análise de Estabilidade Temporal nas Execuções

Com o objetivo de verificar a estabilidade dos tempos de execução dos LLM ao longo de múltiplas execuções, foi aplicado o teste não paramétrico de Friedman (ZIMMERMAN; ZUMBO, 1993). Esse teste permite avaliar se há diferenças estatisticamente significativas entre os tempos médios das três repetições realizadas para cada código, considerando o mesmo modelo de LLM e o mesmo nível de complexidade.

Os resultados, apresentados na Tabela 4, indicam que, para quase todas as combinações de LLM e complexidade analisadas, os valores-p obtidos foram superiores ao limiar de significância de 0,05. Isso sugere que não há diferenças estatisticamente significativas entre os tempos das repetições, reforçando a consistência dos resultados obtidos sob condições controladas.

Dentre os resultados obtidos com o teste de Friedman, observou-se um caso isolado de diferença estatisticamente significativa: o modelo *deepSeek*, quando avaliado com códigos de complexidade simples, apresentou um valor-p de 0,007 (Tabela 4), indicando variação relevante entre os tempos médios das três repetições. Essa discrepância pode refletir instabilidades no processamento do modelo em tarefas menos exigentes, possivelmente associadas à alocação de recursos computacionais ou à variabilidade interna da arquitetura de execução. No entanto, é necessária uma análise mais aprofundada para validar esse resultado.

A estabilidade conseguida para a grande maioria das análises é especialmente importante em contextos de avaliação semi-automatizada, pois indica que o desempenho dos modelos é previsível mesmo em execuções distintas, o que contribui para a confiabilidade do artefato proposto. A Figura 38 apresenta a distribuição dos tempos por repetição e modelo, permitindo

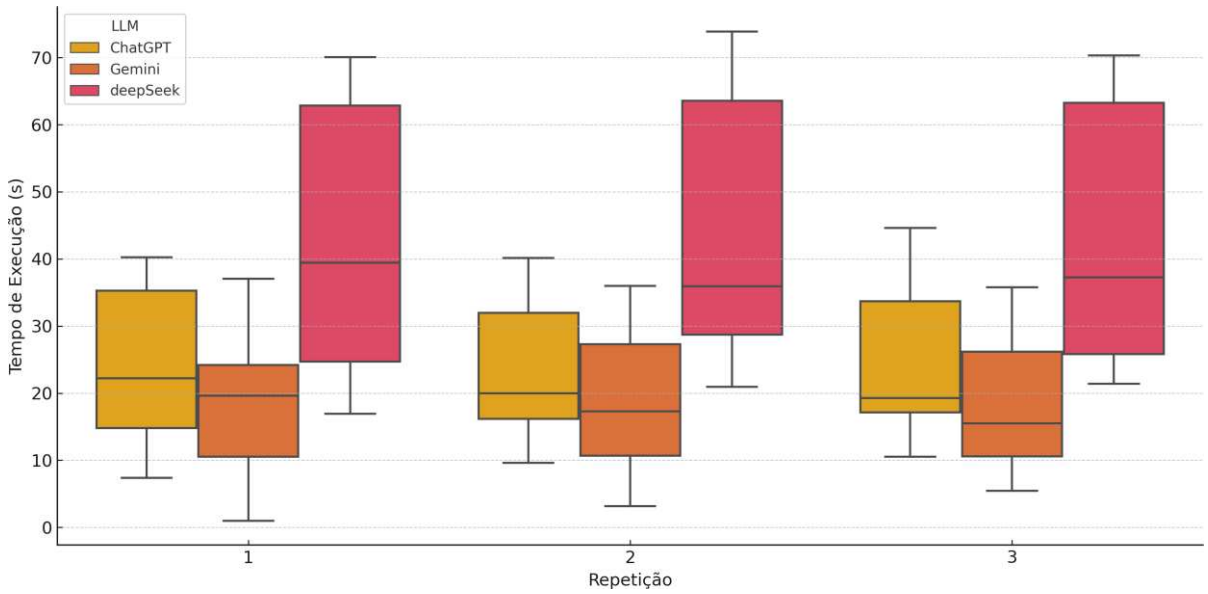
Tabela 4 – Resultados do teste de Friedman para comparação entre repetições (por LLM e complexidade)

LLM	Complexidade	Estatística	Valor-p
ChatGPT	Complexo	2.6	0.272532
ChatGPT	Intermediário	0.6	0.740818
ChatGPT	Simples	1.4	0.496585
Gemini	Complexo	0.0	1.000000
Gemini	Intermediário	3.2	0.201897
Gemini	Simples	0.8	0.670320
deepSeek	Complexo	1.4	0.496585
deepSeek	Intermediário	2.6	0.272532
deepSeek	Simples	9.8	0.007447

Fonte: elaborada pelo autor (2025).

uma análise visual complementar dos resultados.

Figura 38 – Distribuição dos tempos de execução por repetição e LLM

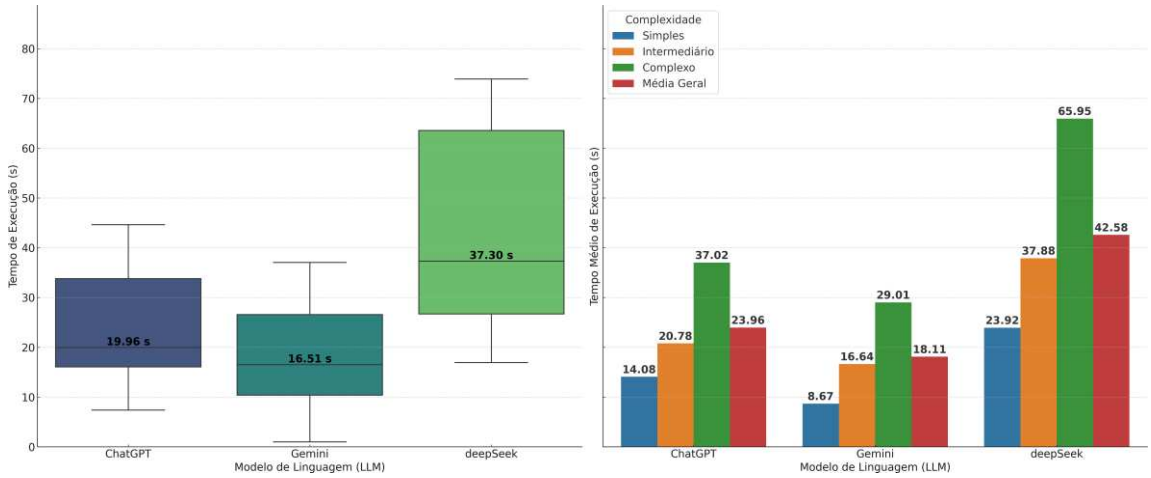


Fonte: elaborada pelo autor (2025).

9.1.4 Análise Consolidada

A Figura 39 apresenta uma análise comparativa dos tempos de execução dos LLM. A Subfigura (A) exibe um boxplot consolidado, evidenciando a distribuição dos tempos e as medianas de cada modelo, enquanto a Subfigura (B) mostra um gráfico de barras agrupadas com os tempos médios por nível de complexidade (*Simples*, *Intermediário* e *Complexo*) e inclui, ainda, uma barra adicional correspondente à média geral de cada LLM.

Figura 39 – (A) Boxplot consolidado por LLM (B) tempo médio por LLM/complexidade



Fonte: elaborada pelo autor (2025).

Esses resultados ratificam o desempenho superior do modelo *Gemini*, que apresentou tempo médio de execução de 23,96 segundos e mediana de 19,96 segundos. Em contraste, o modelo *DeepSeek* revelou-se o menos eficiente nesse aspecto, com tempo médio de 42,58 segundos e mediana de 37,30 segundos. Essa discrepância pode representar um fator crítico em cenários que envolvem grandes volumes de dados ou demandam respostas em tempo real, comprometendo a escalabilidade ou a aplicabilidade do sistema em ambientes educacionais com alta demanda.

9.1.5 Discussão dos Resultados de Desempenho

A análise dos tempos de execução dos modelos ChatGPT, Google Gemini e deepSeek revela padrões coerentes com o comportamento esperado.

O **Google Gemini** foi o modelo com melhor desempenho em termos de tempo de execução, provavelmente beneficiando-se de uma infraestrutura altamente otimizada para respostas rápidas. O **ChatGPT**, utilizando o modelo **GPT-4o**, apresentou tempos um pouco superiores, mas ainda bastante eficientes. Já o **deepSeek**, registrou os maiores tempos médios, o que pode estar relacionado à infraestrutura menos madura da plataforma ou à distribuição limitada de servidores.

A utilização do *método de Monte Carlo*, com três repetições para cada um dos 10 códigos por nível de complexidade, permitiu capturar oscilações naturais nos tempos de execução decorrentes de fatores como variações momentâneas na carga das APIs ou instabilidades de rede.

Embora tenham sido observadas variações entre as repetições, a média das execuções se mostrou estável e suficientemente representativa, oferecendo uma visão confiável do comportamento de cada modelo frente a diferentes níveis de complexidade.

Dessa forma, os resultados demonstram que, embora todas os LLM avaliadas sejam funcionalmente capazes de executar a tarefa proposta, existem diferenças quanto à eficiência temporal.

9.2 Comparação com Avaliação de Especialista

Esta seção apresenta os resultados da etapa de comparação entre a avaliação sugerida pelo artefato *AutoEval-CT* e a avaliação manual conduzida por um *professor especialista*. O principal objetivo desta análise foi verificar a capacidade do artefato em reproduzir, com precisão aceitável, os critérios e padrões adotados na avaliação humana especializada. Para a geração semi-automatizada das avaliações, foi utilizado especificamente o modelo de LLM do *ChatGPT*.

9.2.1 Metodologia da Comparação

Na metodologia adotada, a avaliação dos códigos foi realizada por um professor especialista, com base em uma versão adaptada do artefato *Computacionalidade* (OLIVEIRA; PEREIRA, 2023). Essa avaliação foi conduzida no nível das **evidências** associadas às quatro **habilidades** do Pensamento Computacional: *Abstração*, *Decomposição*, *Reconhecimento de Padrões* e *Algoritmos*. Para cada habilidade, o avaliador atribuiu conceitos segundo uma escala ordinal de 0 a 3 — sendo 0 correspondente a "Nenhuma Evidência" e 3 a "Muita Evidência" —, além da opção "Não se Aplica", destinada a situações em que a evidência em questão não poderia ser inferida a partir do código analisado.

Para a realização da comparação, o professor especialista avaliou **34 códigos Python**, distribuídos em **sete desafios distintos**, conforme descrito a seguir:

- **Desafio 1:** 6 códigos avaliados
- **Desafio 2:** 6 códigos avaliados
- **Desafio 3:** 5 códigos avaliados
- **Desafio 4:** 5 códigos avaliados
- **Desafio 5:** 4 códigos avaliados
- **Desafio 6:** 5 códigos avaliados

– **Desafio 7:** 3 códigos avaliados

Os desafios foram aplicados em turmas dos **períodos iniciais do curso de Ciência da Computação**, possibilitando a análise do Pensamento Computacional em fase de formação introdutória. Essa divisão permitiu a investigação em diferentes contextos, assegurando robustez metodológica à comparação entre as avaliações humana e as sugeridas pelo artefato.

Por outro lado, o artefato AutoEval-CT realiza sua análise num nível mais granular, atribuindo notas individuais aos **indícios** que compõem cada evidência. Para possibilitar a comparação quantitativa entre as avaliações, foi necessário consolidar as notas dos indícios em uma única nota para a evidência correspondente.

9.2.2 Equivalência das Escalas

Para viabilizar a comparação direta entre os julgamentos emitidos pelo professor especialista e aqueles gerados automaticamente pelo artefato AutoEval-CT, foi necessário estabelecer um procedimento de **normalização das escalas de avaliação** utilizadas por ambas as abordagens.

O artefato adota uma escala ordinal estruturada com base na **escala de Likert** (LUNA, 2007), amplamente empregada em pesquisas educacionais, composta pelos seguintes níveis conceituais e valores correspondentes:

– **Nenhum**: Nota 1
– **Baixo**: Nota 2
– **Regular**: Nota 3
– **Bom**: Nota 4
– **Ótimo**: Nota 5

Por sua vez, o professor especialista utilizou uma escala de 0 a 3 pontos, com a possibilidade de registrar o valor “Não se Aplica” nos casos em que determinada evidência não fosse pertinente ao código analisado.

Com o objetivo de permitir a comparação quantitativa e homogênea entre os dois sistemas de avaliação, adotou-se a **normalização das notas para uma escala comum de 1 a 10**, o que possibilitou a aplicação de análises estatísticas compatíveis entre os diferentes instrumentos.

O processo de normalização ocorreu da seguinte maneira:

– A nota atribuída pelo professor foi obtida por meio da média das evidências associadas à

habilidade em questão e posteriormente normalizada para a escala de 1 a 10, conforme a fórmula:

$$\text{Nota Normalizada do Professor} = \frac{\text{Média das Evidências}}{3} \times 9 + 1$$

– A nota gerada pelo artefato foi obtida a partir da média das avaliações das evidências na escala de Likert (1 a 5) e convertida para a escala de 1 a 10 por meio da multiplicação por 2, conforme indicado abaixo:

$$\text{Nota Normalizada do Artefato} = \text{Média das Evidências} \times 2$$

A adoção dessa estratégia de normalização assegurou a compatibilidade entre as diferentes escalas utilizadas, permitindo a comparação direta das avaliações e a mensuração precisa dos índices de concordância entre o avaliador humano e o sistema automatizado.

9.2.3 Análise das Evidências Classificadas como "Não se Aplica"

Além da análise de concordância baseada nas notas atribuídas, foi realizada uma avaliação específica da capacidade do artefato *AutoEval-CT* em reconhecer corretamente os casos em que as evidências de determinadas habilidades do PC não eram aplicáveis ao contexto do desafio proposto. Para isso, foram consideradas as classificações "Não se Aplica" atribuídas pelo professor especialista, bem como a respectiva correspondência gerada automaticamente pelo artefato.

Com o intuito de assegurar maior coerência metodológica e compatibilidade analítica entre os diferentes níveis de granularidade das avaliações, estabeleceu-se um procedimento específico para o tratamento das evidências classificadas como "Não se Aplica" pelo professor especialista. Nesses casos, foram definidos três níveis de correspondência entre as avaliações:

– **Concordância Total**: ocorre quando uma determinada evidência é avaliada como "Não se Aplica" pelo professor, e todos os indícios que a compõem aquela evidência no artefato também recebem essa mesma classificação. Essa situação representa um alinhamento completo quanto à irrelevância daquela evidência no contexto do desafio proposto.
– **Concordância Parcial**: verifica-se quando a evidência é considerada "Não se Aplica" pelo professor, mas apenas um dos indícios associados a ela é classificado de forma

divergente, enquanto os demais permanecem como “Não se Aplica”. Essa condição sugere um elevado grau de convergência, com uma única discrepância pontual na interpretação da aplicabilidade.

- **Discordância Total**: caracteriza-se pela ocorrência de duas ou mais divergências entre os indícios vinculados à mesma evidência, indicando uma discordância substancial quanto à pertinência da evidência analisada no contexto específico.

Esse critério de análise contribui para um maior alinhamento entre as abordagens humana e semi-automatizada, ao mesmo tempo em que proporciona uma leitura pedagógica mais robusta sobre a aplicabilidade — ou não — das habilidades do Pensamento Computacional nos desafios propostos.

9.2.3.1 Cálculo do Percentual de Concordância

O grau de concordância entre as avaliações foi calculado com base na **Concordância Total**, definida pela expressão:

$$\text{Concordância Total (\%)} = \frac{\text{Nº de coincidências exatas após equivalência}}{\text{Nº total de comparações realizadas}} \times 100$$

9.2.3.1.1 Exemplo de cálculo

Para determinada evidência, tenham sido realizadas 18 comparações. Se houve coincidência exata em 15 casos, concordância parcial em 2 casos e discordância total em 1 caso, a Concordância Absoluta seria:

$$\text{Concordância Total} = \frac{15}{18} \times 100 \approx 83,33\%$$

9.2.4 Identificação de Evidências “Não se Aplica”

A Tabela 5 e a Figura 40 apresentam a distribuição de concordância por habilidade, considerando três categorias distintas: **CT** (Concordância Total), **CP** (Concordância Parcial) e **DT** (Discordância Total). Os percentuais foram calculados com base no total de ocorrências analisadas para cada habilidade.

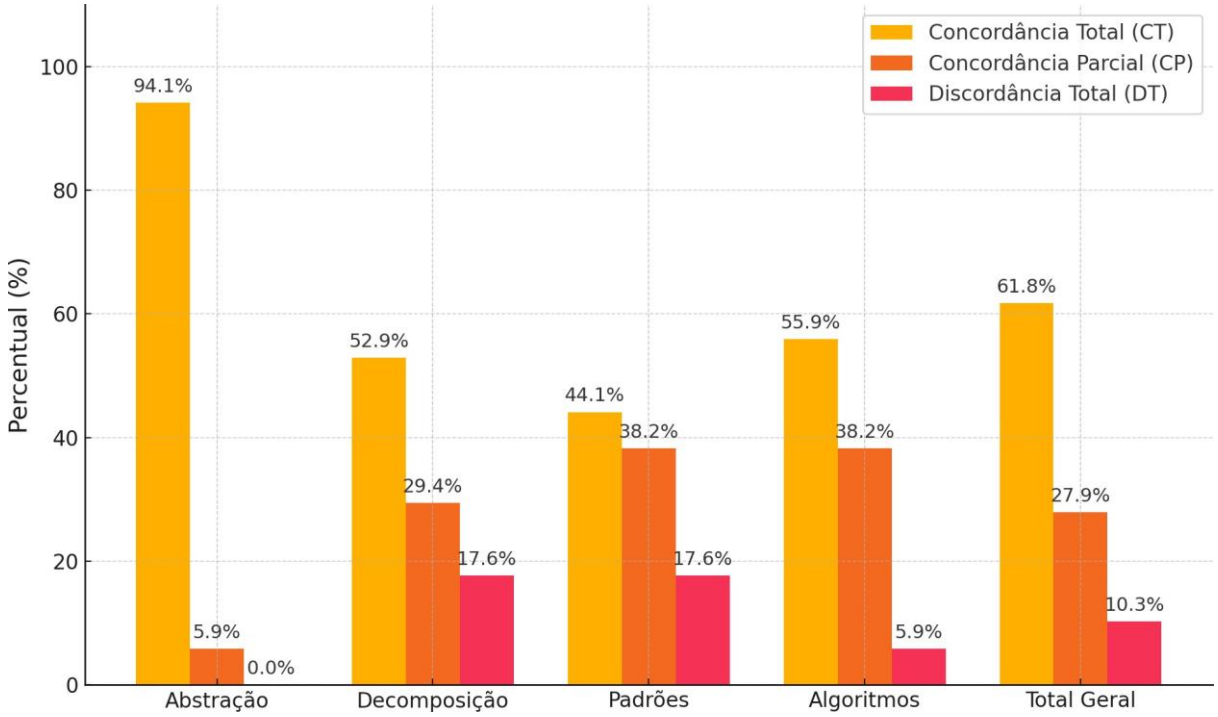
Os dados evidenciam que a maior taxa de concordância total foi observada na habilidade de **Abstração** (94,12%), o que indica elevada precisão do artefato na identificação de

Tabela 5 – Distribuição da concordância nos casos “não se aplica” por habilidade do pc

Habilidade	CT	CP	DT	% CT	% CP	% DT
Abstração	32	2	0	94,12%	5,88%	0,00%
Decomposição	18	10	6	52,94%	29,41%	17,65%
Reconhecimento de Padrões	15	13	6	44,12%	38,24%	17,65%
Algoritmos	19	13	2	55,88%	38,24%	5,88%
Total Geral	84	38	14	61,76%	27,94%	10,29%

Fonte: elaborada pelo autor (2025).

Figura 40 – Distribuição da concordância nos casos “não se aplica” por habilidade do pc



Fonte: elaborada pelo autor (2025).

contextos nos quais essa habilidade não era exigida. Por outro lado, habilidades como **Reconhecimento de Padrões** e **Decomposição** apresentaram níveis mais elevados de discordância parcial e total, revelando maior complexidade interpretativa na determinação da aplicabilidade dessas habilidades.

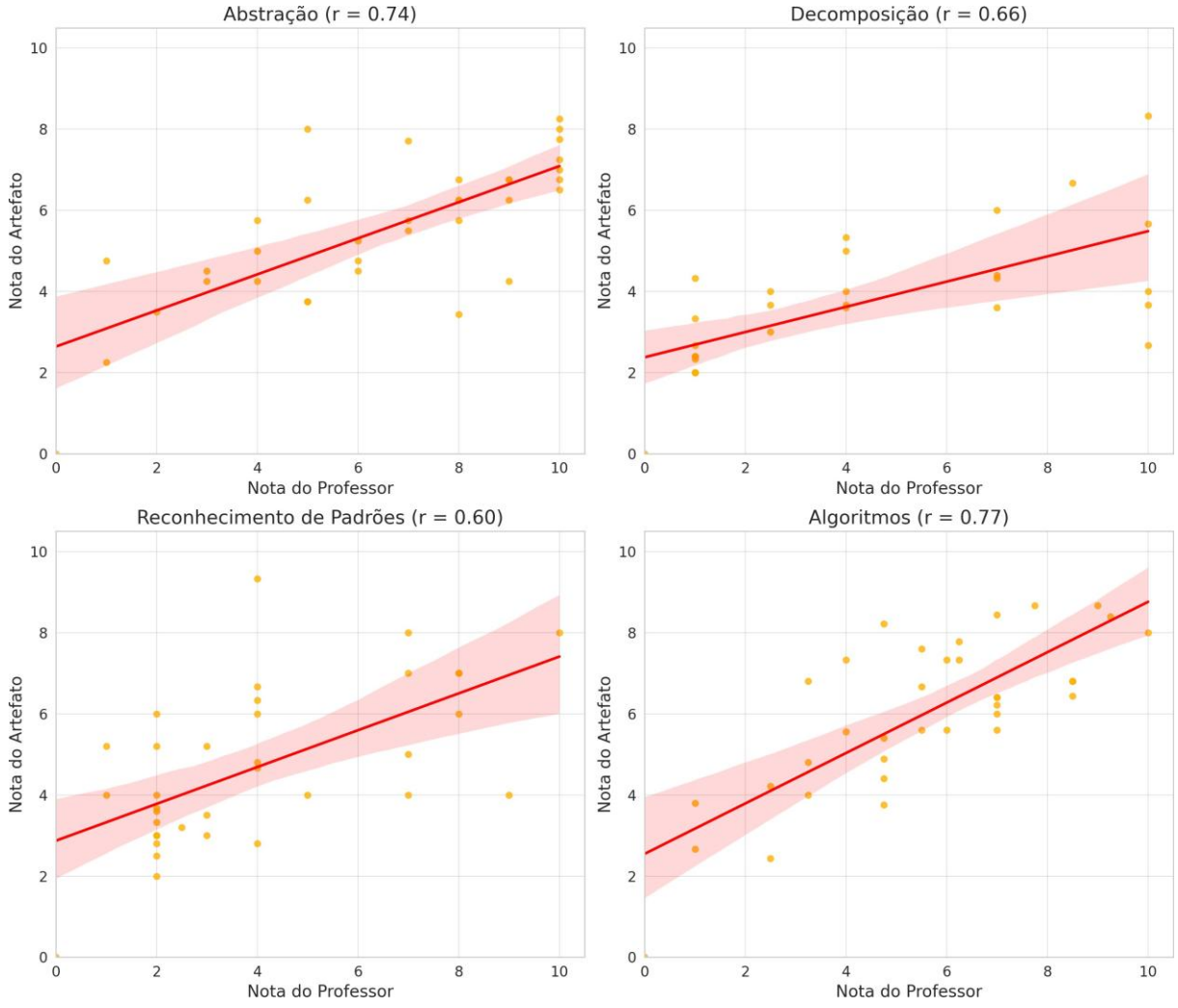
De forma agregada, os resultados mostram que, no total de casos analisados como "Não se Aplica", o sistema obteve **61,76% de Concordância Total**, **27,94% de Concordância Parcial** e apenas **10,29% de Discordância Total**. Esses indicadores reforçam a competência do artefato AutoEval-CT, sobretudo na prevenção de avaliações indevidas em situações de não aplicabilidade, o que é fundamental para a produção de *feedbacks* consistentes e confiáveis aos usuários.

9.2.5 Análise de Correlação entre as Avaliações

Com o objetivo de investigar o grau de concordância entre as avaliações geradas automaticamente pelo artefato *AutoEval-CT* e aquelas atribuídas por um professor especialista, realizou-se uma análise de correlação entre os respectivos escores, segmentada pelas quatro habilidades do PC: Abstração, Decomposição, Reconhecimento de Padrões e Algoritmos.

A Figura 41 apresenta os gráficos de dispersão com regressão linear para cada uma das quatro habilidades do PC, evidenciando o grau de alinhamento entre as avaliações atribuídas pelo professor especialista e aquelas geradas pelo artefato.

Figura 41 – Gráficos de dispersão com regressão linear para as habilidades do pensamento computacional



Fonte: elaborada pelo autor (2025).

A Tabela 6 apresenta os valores de *correlação de Pearson*, *Erro Médio Absoluto*

(MAE) e *Raiz do Erro Quadrático Médio* (RMSE) para cada habilidade avaliada.

Tabela 6 – Síntese das métricas por habilidade do pensamento computacional

Habilidade	Correlação (*r*)	MAE	RMSE
Abstração	0,74 (forte)	1,99	2,31
Decomposição	0,66 (moderada)	2,04	2,74
Reconhecimento de Padrões	0,60 (moderada)	1,69	2,11
Algoritmos	0,77 (forte)	1,38	1,69

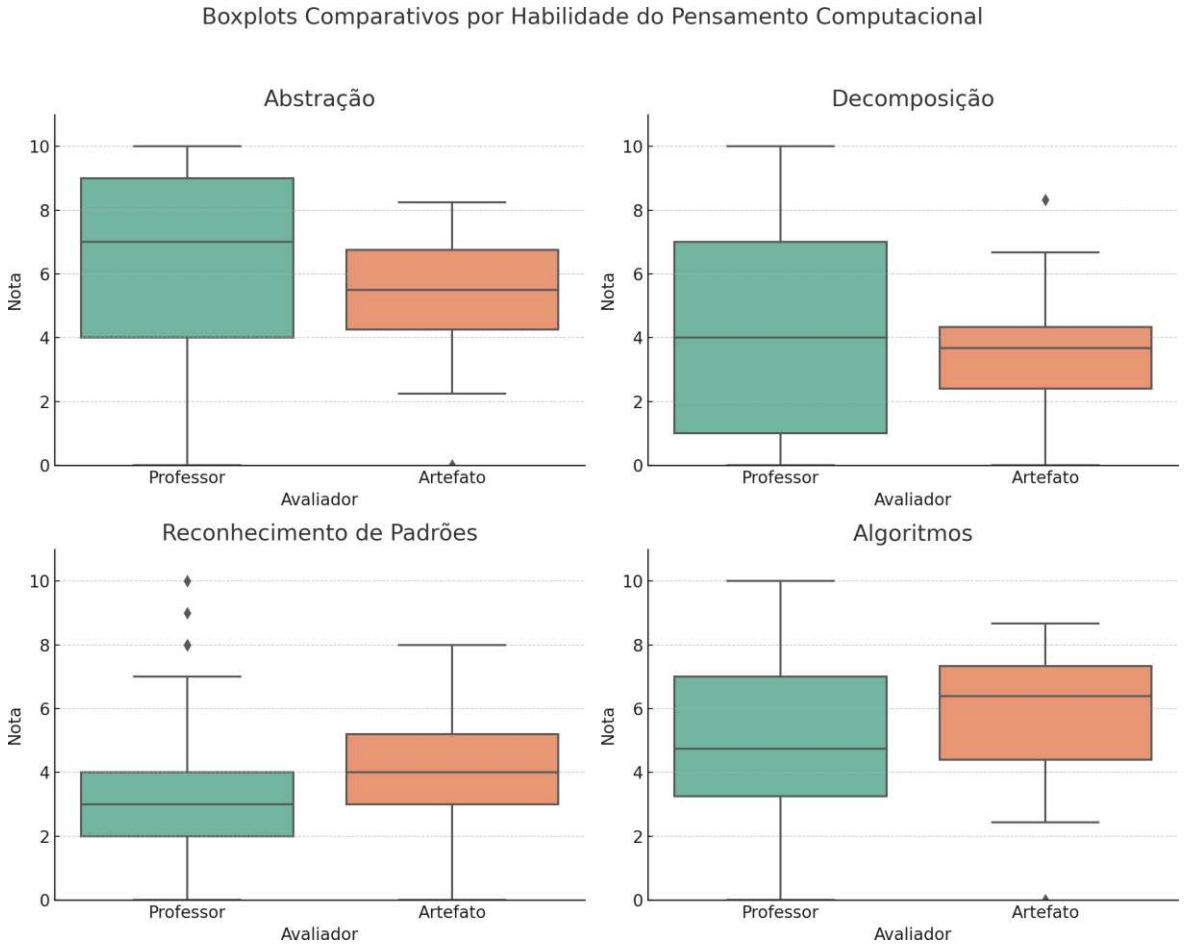
Fonte: elaborada pelo autor (2025).

- **(A) Abstração**: apresenta uma correlação forte ($r = 0{,}74$), embora com uma dispersão, especialmente nas faixas de notas intermediárias e altas. Observa-se uma tendência do artefato em suavizar as avaliações atribuídas pelos professores, resultando nos maiores valores de erro entre todas as dimensões (*MAE* = 1,99; *RMSE* = 2,31).
- **(B) Decomposição**: exibe uma correlação moderada para forte ($r = 0{,}66$), com os dados razoavelmente alinhados à linha de tendência. Trata-se da dimensão com o maior erro quadrático médio (*RMSE* = 2,74), e o maior erro médio absoluto (*MAE* = 2,04), o que revela discrepâncias relevantes em alguns casos.
- **(C) Reconhecimento de Padrões**: com correlação moderada ($r = 0{,}60$), apresenta uma dispersão nos pares de avaliação. Apesar do coeficiente razoável, os erros (*MAE* = 1,69; *RMSE* = 2,11) indicam que o artefato encontra dificuldades para reproduzir com precisão o julgamento humano nessa habilidade.
- **(D) Algoritmos**: é a dimensão com a maior correlação ($r = 0{,}77$) e também os menores valores de erro absoluto (*MAE* = 1,38) e quadrático (*RMSE* = 1,69). Os resultados apontam para um desempenho bastante satisfatório do artefato em reproduzir o padrão avaliativo do professor nessa competência.

A análise da correlação de Pearson indica que as habilidades de Algoritmos e Abstração apresentaram os maiores índices de correlação entre as avaliações humanas e as geradas pelo artefato, enquanto Reconhecimento de Padrões e Decomposição exibiram correlações mais baixas. Apesar desses níveis de correlação, os dados também evidenciam variações importantes nos erros de predição. As dimensões de Abstração e Decomposição demonstraram maior variabilidade nas avaliações e apresentaram as maiores margens de erro, tanto absoluto quanto quadrático. Já a dimensão Reconhecimento de Padrões apresentou desempenho intermediário, com correlação moderada e erros relevantes, sugerindo inconsistências na reprodução do julgamento humano.

Por outro lado, a habilidade de Algoritmos se destacou como a mais bem avaliada pelo artefato, combinando a maior correlação observada com os menores valores de erro. Esses resultados indicam que, embora o artefato seja capaz de se aproximar do padrão avaliativo humano em algumas competências, ainda enfrenta limitações relevantes em habilidades mais abstratas e subjetivas, como a Decomposição e a Abstração.

Figura 42 – Boxplots comparativos por habilidade do pc



Fonte: elaborada pelo autor (2025).

A Figura 42 apresenta os boxplots que evidenciam as diferenças entre as avaliações atribuídas pelo professor e aquelas geradas pelo artefato AutoEval-CT para cada uma das habilidades do Pensamento Computacional.

Na dimensão de **Abstração**, observa-se maior dispersão nas notas atribuídas pelo professor, com uso mais amplo da escala de 1 a 10. Em contraste, o artefato tende a concentrar suas avaliações em uma faixa mais restrita, revelando uma possível suavização das avaliações .

Para a habilidade de **Decomposição**, as diferenças entre as avaliações são mais

acentuadas. As notas atribuídas pelos professores apresentam assimetria e variabilidade considerável, enquanto o artefato distribui suas avaliações de forma mais regular, embora com menor alinhamento em relação ao padrão humano.

Na dimensão de **Reconhecimento de Padrões**, as duas avaliações atribuem notas relativamente baixas, porém com elevada variabilidade. O artefato demonstra dificuldade em captar com precisão os critérios adotados pelo docente, o que resulta em uma dispersão maior das notas.

Já na habilidade de **Algoritmos**, verifica-se o alinhamento mais consistente entre as distribuições das notas. Embora o artefato ainda tenda a evitar os extremos, concentrando suas avaliações em torno de faixas centrais, os padrões gerais entre os dois avaliadores são visivelmente semelhantes.

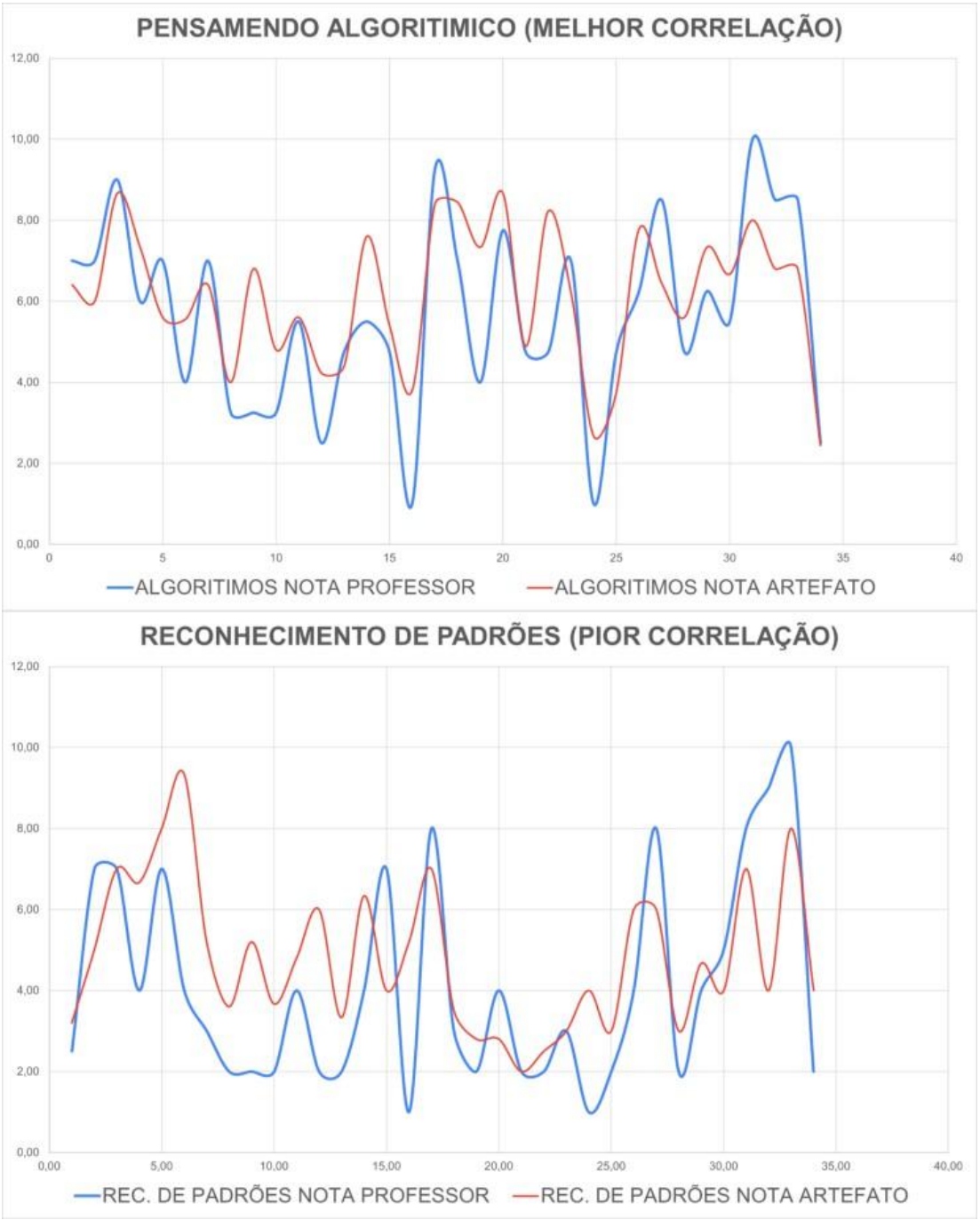
A Figura 43 apresenta os gráficos de dispersão que comparam, ponto a ponto, as notas atribuídas pelo *professor especialista* com aquelas geradas automaticamente pelo artefato. A subfigura superior refere-se à dimensão **Pensamento Algorítmico**, a qual apresentou o mais elevado coeficiente de correlação de Pearson entre as avaliações humanas e semi-automatizadas. Esse resultado pode estar associado ao fato de que a formulação algorítmica, por sua natureza logicamente estruturada e sequencial, favorece critérios objetivos de análise, sendo, portanto, mais facilmente interpretada por modelos computacionais. Por outro lado, a subfigura inferior representa a dimensão **Reconhecimento de Padrões**, que apresentou a menor correlação, indicando maior divergência entre os escores atribuídos. Tal discrepância pode decorrer de ambiguidades na definição dos *indícios* dessa dimensão, especialmente nas Evidências 6 e 7, conforme será discutido como sugestão de aprimoramento futuro na Seção 10.3.1.

Apesar das variações nos coeficientes de correlação, observa-se, de modo geral, uma **tendência de alinhamento** entre as curvas de notas atribuídas pelo professor e pelo artefato, inclusive na dimensão com menor correlação. Essa convergência é ainda mais pronunciada na dimensão com maior correlação, o que reforça a confiabilidade do artefato na análise da habilidade de Pensamento Algorítmico e indica seu potencial de aplicação como ferramenta de apoio à avaliação do Pensamento Computacional.

9.2.6 Análise de Correlação Consolidada

A Figura 44 apresenta uma visão consolidada da relação entre as notas atribuídas pelos professores especialistas e aquelas geradas automaticamente pelo artefato AutoEval-CT,

Figura 43 – Comparativo notas professor e artefato - melhor e pior correlação

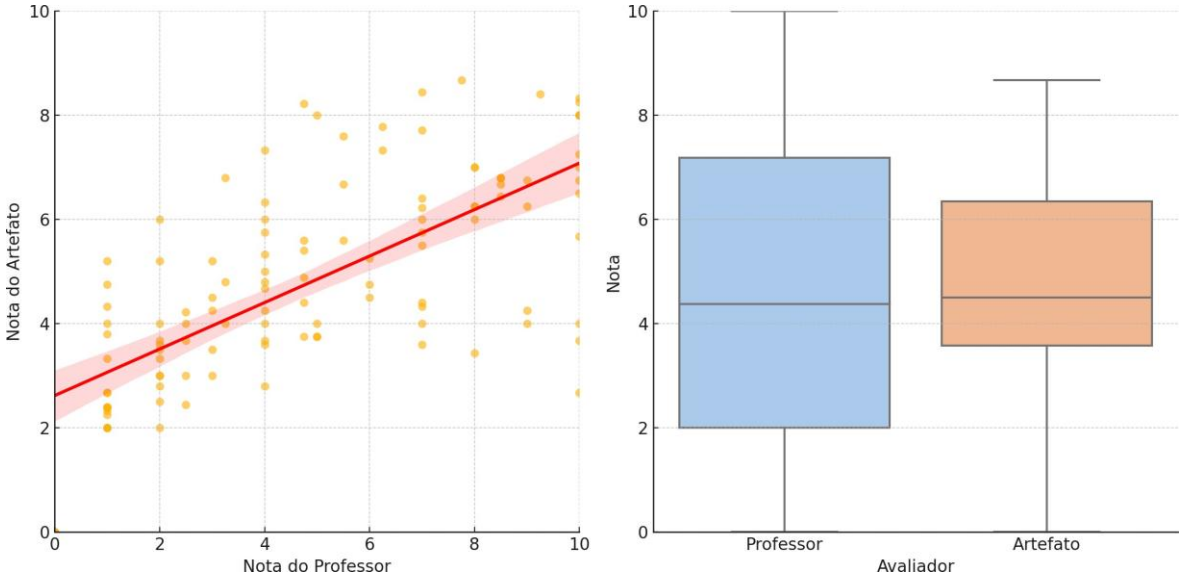


Fonte: elaborada pelo autor (2025).

considerando conjuntamente todas as habilidades do Pensamento Computacional.

No painel **(A)**, o gráfico de dispersão (com regressão linear) evidencia uma correlação moderada a forte ($r = 0{,}69$) entre as avaliações, com uma tendência linear positiva clara. A distribuição dos pontos, embora alinha-se razoavelmente à linha de regressão, revela certa variabilidade, especialmente nas faixas intermediárias da escala, sugerindo que o artefato tende a

Figura 44 – (A) Gráfico de dispersão consolidado (B) boxplot consolidado



Fonte: elaborada pelo autor (2025).

suavizar as notas extremas (altas ou baixas).

No painel **(B)**, o boxplot consolidado permite observar diferenças nas distribuições das notas. Nota-se que o artefato apresenta menor dispersão e menor presença de outliers, com suas avaliações mais concentradas ao redor da mediana. Já as avaliações dos professores demonstram maior amplitude, reforçando a ideia de que o julgamento humano contempla nuances e variações mais amplas, enquanto o modelo automatizado adota um padrão de avaliação mais conservador.

Apesar dessas diferenças na dispersão, observa-se que a **mediana e a média geral das notas** atribuídas por ambos os avaliadores são bastante próximas — *média do professor* = 4,89, *mediana do professor* = 4,38; *média do artefato* = 4,80, *mediana do artefato* = 4,50 — o que reforça a consistência geral do artefato na atribuição de conceitos em uma perspectiva agregada.

9.2.7 Síntese das Comparações

A comparação entre as avaliações dos professores e do artefato AutoEval-CT revelou diferentes níveis de alinhamento conforme a habilidade do Pensamento Computacional analisada.

A maior convergência foi observada na habilidade de **Algoritmos**, que apresentou o maior coeficiente de correlação ($r = 0{,}78$) e os menores erros (*MAE* = 1,38; *RMSE* = 1,69), indicando maior precisão do artefato. A dimensão de **Abstração** também demonstrou boa

correlação ($r = 0{,}73$), mas com erros mais elevados, sugerindo uma suavização das avaliações sugeridas pelo artefato.

As habilidades de **Decomposição** e **Reconhecimento de Padrões** apresentaram menor alinhamento, com correlações moderadas ($r = 0{,}66$ e $r = 0{,}63$, respectivamente) e maiores dispersões nas notas. Esses resultados indicam maior dificuldade do artefato em replicar os critérios subjetivos adotados pelos avaliadores humanos nessas dimensões.

De forma geral, o AutoEval-CT demonstrou desempenho satisfatório, especialmente em **Algoritmos**, mas ainda requer aprimoramentos para lidar com algumas habilidades.

9.3 Avaliação em Ambiente Escolar

Esta seção apresenta a avaliação em ambiente escolar realizada na Escola Estadual de Educação Profissional Monsenhor José Aloysio Pinto, no Estado do Ceará, cujo objetivo principal foi avaliar, em ambiente real, a aplicabilidade do artefato AutoEval-CT no apoio à avaliação do Pensamento Computacional (PC) no Ensino Médio.

9.3.1 Procedimentos do Avaliação em Ambiente Escolar

A avaliação foi conduzida com 23 estudantes do Ensino Médio Técnico, devidamente autorizada pela direção da escola e aprovada pelo Comitê de Ética em Pesquisa da Universidade Estadual Vale do Acaraú (UVA). A Carta de Anuência da direção escolar e o parecer do Comitê de Ética em Pesquisa encontram-se no Anexo A.

Durante a atividade, os alunos resolveram desafios de programação em linguagem Python, utilizando o ambiente computacional da escola. As atividades ocorreram em aula regular no contra turno, acompanhadas pelo professor da disciplina e pelo pesquisador responsável. As soluções desenvolvidas pelos estudantes foram submetidas ao artefato AutoEval-CT, que processou as respostas e forneceu relatórios automáticos de avaliação. A Figura 45 apresenta algumas fotos registradas durante a atividade de resolução dos desafios de programação.

A coleta de dados na avaliação em ambiente escolar se deu por dois instrumentos principais:

1. **Registros do Artefato:** métricas de desempenho, sugestões de conceitos atribuídos aos inícios do PC e feedbacks automáticos gerados para cada código submetido.
2. **Entrevista semi-estruturada:** realizada com o professor responsável pela turma, após a

Figura 45 – Avaliação realizada em ambiente escolar

Fonte: fotos registradas pelo pesquisador (2025).

execução da avaliação em sala de aula, para explorar percepções pedagógicas, técnicas e sugestões de melhorias para o artefato.

A participação dos alunos foi voluntária, mediante assinatura do Termo de Consentimento Livre e Esclarecido (TCLE) pelos responsáveis legais, respeitando a legislação vigente e os princípios éticos da pesquisa com seres humanos.

Disponibilidade da Entrevista: Para fins de transparência científica, o material da entrevista está disponível em dois formatos:

- **Vídeo da Entrevista:** disponível em Link para vídeo.
- **Transcrição textual completa:** disponível no Apêndice H desta tese.

9.3.2 Aspectos Éticos

Todos os procedimentos metodológicos foram cuidadosamente planejados de forma a garantir o respeito aos princípios éticos aplicáveis à pesquisa com seres humanos, assegurando o **anonimato, a confidencialidade dos dados e a participação voluntária dos alunos**, mediante assinatura do *Termo de Consentimento Livre e Esclarecido (TCLE)* pelos responsáveis legais, além do *Termo de Assentimento Livre e Esclarecido (TALE)* para os estudantes menores de idade.

Destaca-se que o projeto foi submetido, apreciado e aprovado pelo *Comitê de Ética em Pesquisa (CEP) da Universidade Estadual Vale do Acaraú (UVA)*, sob o **CAAE nº 88541225.6.0000.5053**, conforme registrado no **Parecer nº 7.722.336**. Todo o processo atendeu às recomendações estabelecidas na *Resolução nº 510/2016 do Conselho Nacional de Saúde*, bem como aos princípios da *Lei Geral de Proteção de Dados (LGPD)*, garantindo que todos os cuidados éticos, legais e metodológicos sejam devidamente observados.

O desenvolvimento deste estudo não implicou em qualquer risco físico, clínico ou social aos participantes. Eventuais desconfortos que tenham ocorrido durante o processo, foram de natureza exclusivamente pedagógica, decorrentes do próprio processo de resolução dos desafios propostos, sendo minimizados pela natureza formativa das atividades, pelo acompanhamento constante do docente e do pesquisador e pelo caráter não punitivo das avaliações realizadas.

9.3.3 Análise Qualitativa da Entrevista

Para analisar a entrevista realizada com o professor responsável, foi empregada a técnica de **Análise de Conteúdo**, conforme proposto por Bardin (BARDIN, 2016). Trata-se de um procedimento sistemático que possibilita identificar, categorizar e interpretar sentidos subjacentes às falas do entrevistado, estruturando-as em categorias temáticas.

O processo seguiu três etapas principais:

1. Pré-Análise

Foi realizada uma leitura flutuante da transcrição da entrevista, buscando familiarização inicial com o material e identificação preliminar de temas recorrentes. Delimitou-se o corpus textual a ser submetido à análise, composto pelas falas do professor registradas no roteiro semi-estruturado.

2. Exploração do Material

Foram definidos núcleos de sentido, extraídos de palavras, expressões e trechos significativos. A codificação foi realizada por meio de categorias temáticas construídas a posteriori, emergentes dos próprios dados, conforme orientação metodológica de Bardin. As principais categorias identificadas foram:

- **Impacto pedagógico da ferramenta;**

- **Qualidade e coerência dos feedbacks;**
- **Tempo e eficiência na avaliação;**
- **Sugestões de melhorias na apresentação dos relatórios;**
- **Possibilidades de expansão para outras áreas.**

3. Tratamento e Interpretação dos Resultados

Os núcleos de sentido foram agrupados e interpretados à luz dos objetivos da pesquisa, buscando compreender as percepções do professor sobre o uso do artefato AutoEval-CT.

9.3.4 Resultados da Entrevista

A seguir, apresentam-se os principais resultados qualitativos obtidos na entrevista, organizados segundo as categorias identificadas:

Impacto Pedagógico da Ferramenta

O professor destacou o potencial do artefato para otimizar o tempo de aula e melhorar a eficácia pedagógica:

> "Ganharíamos muito tempo, pois os relatórios já trazem os erros mais comuns. A aula ficaria mais objetiva e interativa, permitindo abordar diretamente as dificuldades específicas de cada aluno."

Também ressaltou a utilidade dos relatórios na gestão pedagógica:

> "Se percebo que muitos alunos não conseguiram atingir um objetivo em determinada questão, já sei que preciso revisar o assunto ou explicar de outra forma."

Qualidade e Coerência dos Feedbacks

O professor considerou que, de modo geral, os feedbacks gerados foram coerentes com sua própria análise, mas notou pequenas limitações em tópicos mais conceituais:

> "Percebi coerência. Em algumas atividades, por exemplo, na parte de conceituação de estruturas de repetição, havia feedbacks corretos e outros nem tanto. Mas, de forma

> geral, ajudou a identificar quando o aluno não utilizou corretamente determinada estrutura."

Em termos gerais, não foram relatadas discordâncias significativas entre a avaliação do professor e a do artefato:

> "Achei que a ferramenta relatou perfeitamente o que o aluno havia feito ou deixado de fazer."

Tempo e Eficiência na Avaliação

O docente reconheceu ganhos expressivos de tempo e objetividade:

> "Os relatórios ajudam muito, pois permitem visualizar, de forma mais técnica, os conceitos que os alunos utilizaram nas respostas. Gostei bastante dessa parte."

Sugestões de Melhorias

O professor indicou melhorias desejáveis, sobretudo na apresentação visual dos relatórios:

> "Os relatórios poderiam ter uma apresentação mais visual, pois atualmente são muito literais. Um gráfico ou outra forma visual facilitaria a leitura e a análise."

Também sugeriu incluir comparações entre a solução do aluno e uma questão-modelo:

> "Seria interessante ter uma comparação entre a solução do aluno e uma questão-modelo, mostrando quais pontos eram esperados e quais não foram atendidos."

Possibilidades de Expansão

Por fim, o professor identificou potencial para uso do artefato em outras áreas além da programação:

> "Vejo aplicação possível em disciplinas como cálculo ou física, onde também existem diferentes métodos para resolver problemas."

9.3.5 Síntese dos Resultados da Avaliação em Ambiente Escola

A análise de conteúdo evidenciou percepções amplamente positivas por parte do professor quanto ao uso do AutoEval-CT. Os principais benefícios relatados incluíram economia de tempo, apoio à mediação pedagógica e potencial para personalização do ensino. Os feedbacks foram considerados coerentes, embora tenham sido indicadas sugestões de aprimoramento, sobretudo no aspecto visual dos relatórios e na comparação com soluções-modelo. Tais resultados corroboram a eficácia do artefato como ferramenta de apoio à avaliação do Pensamento Computacional no Ensino Médio e sinalizam oportunidades para seu aprimoramento técnico e pedagógico.

9.4 Conclusão do Capítulo

Os resultados da fase de experimentação demonstram a viabilidade técnica e pedagógica do artefato AutoEval-CT, apresentando bons índices de concordância com avaliação humana e desempenho satisfatório das LLMs testadas. As percepções qualitativas indicam que, embora existam pontos a aprimorar, a ferramenta possui grande potencial para integrar práticas de ensino e avaliação no contexto educacional brasileiro.

10 CONSIDERAÇÕES FINAIS

Este capítulo traz as principais conclusões desta pesquisa, organizadas em três seções complementares: **Conclusões**, que apresenta as principais conclusões obtidas a partir dos resultados alcançados; **Contribuições**, que descreve os potenciais avanços teóricos, metodológicos e práticos decorrentes do desenvolvimento desta tese; e, por fim, **Trabalhos Futuros**, seção dedicada à discussão das possibilidades de continuidade, aprofundamento e expansão das investigações, com base nos resultados obtidos.

10.1 Conclusões

A presente tese teve como objetivo desenvolvimento e a analise uma abordagem avaliativa e um artefato destinados a auxiliar professores do Ensino Médio na avaliação da aprendizagem do PC. O trabalho fundamentou-se na premissa de que a avaliação do PC é um processo complexo, que envolve múltiplas habilidades, tais como abstração, decomposição, reconhecimento de padrões e pensamento algorítmico, e que, muitas vezes, carece de instrumentos sistemáticos para sua realização.

A pesquisa demonstrou que o AutoEval-CT é capaz de identificar evidências e indícios de PC em códigos de programação produzidos por estudantes, empregando modelos de *Large Language Models* (LLM) para análise semi-automatizada e geração de relatórios detalhados. Os resultados obtidos, tanto na comparação entre avaliações quanto no experimento controlado realizado em ambiente escolar, evidenciaram bons níveis de concordância, confirmando a viabilidade do uso de ferramentas baseadas em inteligência artificial para apoiar o trabalho docente na avaliação do PC.

De modo mais específico, os resultados permitiram responder às questões de pesquisa desta tese:

1. **De que forma um artefato de avaliação baseado na análise de códigos de programação pode auxiliar o professor no processo de avaliação do PC?**
   O estudo confirmou que o artefato auxilia o trabalho docente, oferecendo relatórios estruturados que capturam dimensões do PC além da mera correção sintática, permitindo ao professor focar em aspectos mais pedagógicos do processo avaliativo.
2. **Como os professores percebem as contribuições advindas do uso de uma plataforma de suporte à avaliação do PC?**

As entrevistas realizadas indicaram percepções positivas: o artefato contribuiu para economia de tempo, maior objetividade na correção e apoio à personalização do ensino. Os docentes também apontaram sugestões de aprimoramento, sobretudo na visualização dos relatórios e na comparação com soluções-modelo.

3. **De que forma a análise semi-automatizada de códigos contribui para uma avaliação mais focada das diferentes dimensões do PC?**

   Os experimentos mostraram que o AutoEval-CT conseguiu capturar múltiplos indicadores — abstração, decomposição, reconhecimento de padrões e algoritmos — em níveis satisfatórios. Ainda que algumas dimensões, como a abstração, apresentem maior complexidade, os resultados confirmaram a abordagem quali-quantitativa hierárquica proposta.

4. **Qual o grau de precisão ou concordância das avaliações semi-automatizadas em relação às avaliações humanas?**

   As análises de correlação apontaram índices satisfatórios de concordância entre o artefato e especialistas/professores, sobretudo em dimensões como algoritmos e reconhecimento de padrões. Esses resultados atestam a assertividade das avaliações semi-automatizadas, ao mesmo tempo em que revelam pontos a serem refinados.

5. **Como diferentes modelos de linguagem de grande escala (LLM) se comportam no processo de avaliação do PC?**

   O estudo comparativo demonstrou que distintos modelos de linguagem apresentam desempenhos variáveis em termos de tempo de execução, estabilidade e qualidade das análises. Ainda assim, todos se mostraram aptos a apoiar a avaliação do PC, sendo possível identificar modelos mais eficientes para contextos educacionais específicos.

Portanto, esta pesquisa confirma a hipótese central de que uma abordagem avaliativa hierárquica, fundamentada na análise semi-automatizada de códigos com suporte de LLM, proporciona suporte concreto aos professores do Ensino Médio na avaliação do PC. Além disso, abre caminhos para o aprimoramento contínuo de práticas avaliativas em contextos educacionais mediados por tecnologias inteligentes.

Adicionalmente, o estudo revelou contribuições importantes do artefato não apenas para a prática avaliativa, mas também para o avanço metodológico e tecnológico no campo do ensino de PC. A abordagem quali-quantitativa hierárquica adotada nesta tese permitiu mensurar indicadores de desempenho dos estudantes e fornecer *feedbacks* pedagógicos personalizados, apoiando os docentes na avaliação do PC do estudantes.

10.2 Contribuições

A principal contribuição desta pesquisa consiste no desenvolvimento e análise de uma abordagem de avaliação e de um artefato que automatiza o processo de correção e *feedback* sobre códigos de programação, voltado especificamente para habilidades do PC no Ensino Médio. Entre os benefícios e contribuições esperadas, destacam-se:

1. **Melhoria no Processo de Avaliação:** Criação de um instrumento que auxilie os professores a analisar, de forma rápida e assertiva, o desempenho dos estudantes em resolução de problemas que envolvem lógica de programação, contribuindo para padronizar e agilizar a atribuição de notas e *feedbacks*.
2. **Otimização do Tempo Docente:** Redução do tempo gasto em tarefas repetitivas de correção, permitindo que o professor possa dedicar-se a atividades mais estratégicas, como planejamento de aulas, desenvolvimento de materiais adicionais e acompanhamento individualizado dos alunos.
3. **Fortalecimento do Pensamento Computacional:** Estímulo à adoção de práticas de ensino de PC no Ensino Médio, assegurando que os estudantes recebam *feedback* imediato sobre seus códigos, o que favorece o desenvolvimento de habilidades de decomposição, abstração, reconhecimento de padrões e raciocínio algorítmico.
4. **Evidências Empíricas para a Comunidade Acadêmica:** Os resultados desta pesquisa, obtidos por meio de estudos experimentais e/ou quasi-experimentais, fornecem evidências quantitativas e qualitativas acerca da eficácia do artefato, possibilitando que a comunidade científica, os gestores educacionais e os docentes compreendam os fatores de sucesso e desafios na adoção de ferramentas para o ensino de PC.
5. **Possível Expansão para Outras Áreas:** Embora direcionado à avaliação de códigos em Python para fins de aprendizagem de PC, o método e a arquitetura do artefato podem ser adaptados para diferentes linguagens de programação e mesmo para tarefas que envolvam raciocínio computacional em áreas correlatas, ampliando o alcance e a aplicabilidade do sistema.

10.3 Trabalhos Futuros

Em função dos resultados alcançados e das limitações identificadas, elencam-se abaixo algumas possibilidades de aprimoramento e expansão do artefato, bem como novas

perspectivas de pesquisa a serem investigadas em estudos futuros:

10.3.1 Ajustes dos Indícios das Evidências 6 e 7

Foi identificado que os indícios inicialmente vinculados às Evidências 6 e 7 necessitam de reestruturação para melhor alinhamento com o escopo pretendido de cada evidência.

O indício 6.1, atualmente vinculado à Evidência 6, deverá compor a Evidência 7, passando a ser denominado Indício 7.3.

O indício 6.2, também atualmente vinculado à Evidência 6, deverá compor a Evidência 7, passando a ser denominado Indício 7.4.

O indício 6.3 será excluído por não estar alinhado ao propósito específico da Evidência 6.

A Evidência 6, cujo foco passa a ser **"Os padrões do desafio foram identificados"**, passa a ter novos indícios definidos, conforme segue:

- **Indício 6.1**: Identificação explícita de repetições ou regularidades nos dados ou no enunciado.
  Ex.: o estudante percebe que há uma sequência que se repete (e.g. "há sempre dois zeros entre cada 1 no vetor").
- **Indício 6.2**: Identificação de regras que se mantêm constantes no problema.
  Ex.: "todos os quadros têm o mesmo tamanho".
- **Indício 6.3**: Uso de estratégias baseadas em padrões percebidos no desafio.
  Ex.: uso de contadores, laços, expressões regulares ou filtros específicos porque o estudante percebeu padrões no texto ou nos dados do problema.
- **Indício 6.4**: Declaração no código ou em comentários sobre padrões observados no desafio.
  Ex.: comentário no código:
  `# todos os nomes seguem o padrão 'Título Nome Sobrenome'.`

Ressalta-se que haverá necessidade de avaliação dessas alterações, visando confirmar sua adequação à prática avaliativa.

10.3.2 Ampliação das Linguagens de Programação Aceitas

Embora o presente trabalho se concentre em Python, há demanda para avaliação automatizada (ou semi-automatizada) em outras linguagens textuais (por exemplo, Java, C++ ou

JavaScript) e também em linguagens de blocos, como Scratch e App Inventor. Essa expansão aumenta o alcance pedagógico e a aplicabilidade do artefato em diversos contextos de ensino de programação e Pensamento Computacional.

10.3.3 Expansão das Habilidades Avaliadas

As habilidades já abordadas (por exemplo, decomposição, abstração, reconhecimento de padrões) podem ser complementadas com outras vertentes do Pensamento Computacional, bem como com elementos relacionados a competências socioemocionais e práticas colaborativas, oferecendo uma análise ainda mais completa do desempenho dos estudantes.

10.3.4 Melhorias na Interface e na Experiência do Usuário (UI/UX)

A usabilidade do sistema e a clareza na apresentação dos resultados são determinantes para a adoção em larga escala. Planeja-se evoluir a interface para torná-la mais intuitiva, responsiva e adaptada às necessidades de distintos perfis de usuários (professores, estudantes e gestores educacionais).

10.3.5 Transformação em um Produto no modelo Software as a Service (SaaS)

Para promover a utilização efetiva e simultânea por múltiplos municípios, escolas, turmas, professores e alunos, uma proposta futura é a disponibilização do artefato como serviço em nuvem, com recursos de gestão de usuários, turmas e relatórios de resultados em tempo real. Essa abordagem facilitaria a escalabilidade, a manutenção contínua do sistema e a integração com plataformas educacionais já existentes.

REFERÊNCIAS

ALEXANDRINO, N. L.; SILVA, C. A.; TARGA, C. N.; CONRADO, D. B. Ps4w: Programa de inclusão jovem e feminina na área tecnológica. **EduComp'21**, Jataí, Goiás, p. 204-210, abr. 2021. Disponível em: https://sol.sbc.org.br/index.php/educomp/article/view/14486/14332. Acesso em: 19 out. 2022.

ALMEIDA, R.; GOMES, A.; BIGOTTE, M. E.; PESSOA, T. iprog: getting started with programming: pilot experiment in two elementary schools. *In*: INTERNATIONAL SYMPOSIUM ON COMPUTERS IN EDUCATION (SIIE), 19., 2018, Lisboa. **Anais** […]. Lisboa: IEEE, 2017. p. 1-5.

ALMEIDA, R.; PESSOA, T.; GOMES, A. Learning to think like a trainer: bringing scratch for educational sciences professional's formation. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 48., 2018, San Jose. **Anais** […]. San Jose: IEEE, 2018. p. 1-7.

ALVES, N. D. C.; WANGENHEIM, C. G. V.; HAUCK, J. C. Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. **Informatics in Education**, v. 18, n. 1, p. 17, 2019. Publisher: Institute of Mathematics and Informatics.

AMBROSE, G.; HARRIS, P. **Design thinking:** Coleção design básico. [*S. l.*]: Bookman, 2016.

ANDERSON, L. W.; KRATHWOHL, D. R. **A taxonomy for learning, teaching, and assessing**: A revision of Bloom's taxonomy of educational objectives: complete edition. [*S. l.*]: Addison Wesley Longman, Inc., 2001.

ANISTYASARI, Y.; EKOHARIADI, E.; BUDITJAHJANTO, I. P. A.; HIDAYATI, S. C. Exploring the psychometric properties of computational thinking assessment in introductory programming. *In*: INTERNATIONAL E-ENGINEERING EDUCATION SERVICES CONFERENCE (E-ENGINEERING), 1., 2021, on-line. **Anais** […]. [*S. l.*]: IEEE, 2021. p. 88–93.

ARAUJO, A. L.; ANDRADE, W.; GUERRERO, D. Um mapeamento sistemático sobre a avaliaçao do pensamento computacional no brasil. *In*: WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 5., 2016, Uberlândia. **Anais** [...]. Uberlândia: SBC, 2016. p. 1147.

ARAUJO, A. L. S. O.; SANTOS, J. S.; ANDRADE, W. L.; GUERRERO, D. D. S.; DAGIENĖ, V. Exploring computational thinking assessment in introductory programming courses. **IEEE Frontiers in education conference (FIE)**, Indianapolis, USA, p. 1–9, 2017. Disponível em: https://ieeexplore.ieee.org/abstract/document/8190652. Acesso em: 19 out. 2022.

ASLINA, Y. R.; MULYANTO, A.; NIWANPUTRI, G. S. Designing "bebras" serious games interaction for indonesian upper elementary school students. *In*: INTERNATIONAL CONFERENCE ON ADVANCE INFORMATICS: CONCEPTS, THEORY AND APPLICATIONS (ICAICTA), 7., 2020, Tokoname. **Anais** […]. Tokoname: IEEE, 2020. p. 1–6.

AUSUBEL, D. P.; NOVAK, J. D.; HANESIAN, H. **Psicologia educacional**. [*S. l.*]: Interamericana, 1980.

BARBOSA, L. L. da S.; MALTEMPI, M. V. Matemática, pensamento computacional e bncc: desafios e potencialidades dos projetos de ensino e das tecnologias na formação inicial de professores. **Revista Brasileira de Ensino de Ciências e Matemática**, v. 3, n. 3, 2020.

BARDIN, L. **Análise de Conteúdo**. [*S. l.*]: Edições 70, 2010.

BARDIN, L. **Análise de conteúdo**. Tradução de Luís Antero Reto e Augusto Pinheiro. São Paulo: Edições 70, 2016.

BARR, V.; STEPHENSON, C. Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community? **Acm Inroads**, ACM New York, v. 2, n. 1, p. 48–54, 2011.

BARR, V.; STEPHENSON, C. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? **Acm Inroads**, ACM New York, v. 2, n. 1, p. 48-54, 2011.

BASAWAPATNA, A.; KOH, K. H.; REPENNING, A.; WEBB, D. C.; MARSHALL, K. S. Recognizing computational thinking patterns. *In*: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 42., 2011, Dallas. **Anais** […]. Dallas: ACM, 2011. p. 245–250.

BELLONI, M. L. Ensaio sobre a educação a distância no brasil. **Educação & sociedade**, SciELO Brasil, v. 23, p. 117–142, 2002.

BNCC, B. **Base Nacional Comum Curricular**. 2018. Disponível em: https://www.gov.br/mec/pt-br/escola-em-tempo-integral/BNCC_EI_EF_110518_versaofinal.pdf. Acesso em: 12 abr. 2025.

BOGDANCHIKOV, A.; ZHAPAROV, M.; SULIYEV, R. Python to learn programming. **Journal of Physics**: Conference Series, [*S. l.*], v. 423, n. 1, p. 012027, 2013.

BORELA, R.; LIDING, Z.; MCDANIEL, M. Enhancing cs1 education through experiential learning with robotics projects. *In*: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 56., 2025, Pittsburgh, EUA. **Anais** [...]. Pittsburgh: ACM, 2025. p. 144–150.

BRACKMANN, C. P. **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. 2017. 167 f. Tese (Doutorado em Informática na Educação) – Programa de Pós-Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

BRUNER, J. S. "The process of education" revisited. **The Phi Delta Kappan**, JSTOR, v. 53, n. 1, p. 18–21, 1971.

BUNDY, A. Computational thinking is pervasive. **Journal of Scientific and Practical Computing**, v. 1, n. 2, 2007.

BUNDY, A. Computational thinking is pervasive. **Journal of Scientific and Practical Computing**, v. 1, n. 2, p. 67–69, 2007.

BURLESON, W. S.; HARLOW, D. B.; NILSEN, K. J.; PERLIN, K.; FREED, N.; JENSEN, C. N.; LAHEY, B.; LU, P.; MULDNER, K. Active learning environments with robotic tangibles: Children's physical and virtual spatial programming experiences. **IEEE Transactions on Learning Technologies**, IEEE, v. 11, n. 1, p. 96–106, 2017.

CABALLERO-GONZÁLEZ, Y.-A.; MUÑOZ, L.; MUÑOZ-REPISO, A. G.-V. Pilot experience: Play and program with bee-bot to foster computational thinking learning in young children. *In*: INTERNATIONAL ENGINEERING, SCIENCES AND TECHNOLOGY CONFERENCE (IESTEC), 7., 2019, Panamá. **Anais** […]. Panamá: IESTEC, p. 601–606.

CACHERO, C.; BARRA, P.; MELIÁ, S.; LÓPEZ, O. Impact of programming exposure on the development of computational thinking capabilities: An empirical study. **IEEE Access**, IEEE, v. 8, p. 72316–72325, 2020.

CAMARGO, C.; BRANCALIÃO, L.; GONÇALVES, J.; LIMA, J.; RAMOS, M.; FERNANDES, L.; TROVISCO, M.; CONDE, M. Emídio garcia school pilot description: A robosteam erasmus+ project activity based on a challenge based learning approach. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 4., 2020, Porto. **Anais** […]. Porto: IEEE, 2020. p. 290-294.

CASTOLDI, R.; POLINARSKI, C. A. A utilização de recursos didático-pedagógicos na motivação da aprendizagem. *In*: SIMPÓSIO NACIONAL DE ENSINO DE CIÊNCIA E TECNOLOGIA, 1., 2009, Rio de Janeiro. **Anais** […]. Rio de Janeiro: PPGECT, p. 684.

CHAI, X.; SUN, Y.; LUO, H.; GUIZANI, M. Dwes: A dynamic weighted evaluation system for scratch based on computational thinking. **IEEE Transactions on Emerging Topics in Computing**, IEEE, v. 10, n. 2, p. 917–932, 2021.

CHEN, B.-H.; HUANG, T.-F.; CHOU, S.-C. Research on teaching effectiveness of computational thinking based on service learning. *In*: INTERNATIONAL CONFERENCE ON TECHNOLOGIES AND APPLICATIONS OF ARTIFICIAL INTELLIGENCE (TAAI), 26., 2021, Taichung. **Anais** […]. Taichung: IEEE, 2021. p. 155–161.

CHOLLET, F. **Deep Learning With Python Manning Publications Company**. [*S. l.*]: Manning Publications Company, 2017.

CODE.ORG. **Sobre nós Code.Org**. 2022. Disponível em: https://code.org/international/about. Acesso em: 23 out. 2022.

COOPER, S.; PÉREZ, L. C.; RAINEY, D. K–12 computational learning. **Communications of the ACM**, New York, v. 53, n. 11, p. 27–29, 2010.

CRONBACH, L. J. Coefficient alpha and the internal structure of tests. **Psychometrika**, Springer, v. 16, n. 3, p. 297–334, 1951.

CURASMA, R. P.; JARA, N. J.; CURASMA, H. P.; ORNETTA, V. C. Assessment of

computational thinking in regular basic education: case ietp “josé obrero”. *In*: INTERNATIONAL CONFERENCE ON ELECTRONICS, ELECTRICAL ENGINEERING AND COMPUTING (INTERCON), 26., 2019, Arequipa. **Anais** […]. Arequipa: IEEE, 2019. p. 1-4.

CUTUMISU, M.; GUO, Q. Using topic modeling to extract pre-service teachers’ understandings of computational thinking from their coding reflections. **IEEE Transactions on Education**, IEEE, v. 62, n. 4, p. 325–332, 2019.

DAGIENĖ, V.; FUTSCHEK, G. Bebras international contest on informatics and computer literacy: Criteria for good tasks. *In*: INTERNATIONAL CONFERENCE ON INFORMATICS IN SECONDARY SCHOOLS-EVOLUTION AND PERSPECTIVES, 3., 2008, Torun Poland. **Anais** […]. Torun Poland: ISSEP, 2008. p. 19–30.

DIAZ, N. V. M.; MEIER, R.; TRYTTEN, D. A.; YOON, S. Y. Computational thinking growth during a first-year engineering course. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 50., 2020, on-line. **Anais** […]. *[S. l.]*: IEEE, 2020. p. 1-7.

DRESCH, A.; LACERDA, D. P.; ANTUNES JUNIOR, J. A. V. **Design science research**: método de pesquisa para avanço da ciência e tecnologia. [*S. l.*]: Bookman, 2020.

DUAN, Z.; RUSSELL, I.; JUNG, A. Active learning strategies: A computing course for undergraduates. *In*: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE & EDUCATION (ICCSE), 16., 2021, Lancaster. **Anais** […]. Lancaster: IEEE, 2021. p. 301–306.

DURAK, H. Y.; SARITEPECI, M. Analysis of the relation between computational thinking skills and various variables with the structural equation model. **Computers & Education**, Elsevier, v. 116, p. 191-202, 2018.

EGUÍLUZ, A.; GUENAGA, M.; GARAIZAR, P.; OLIVARES-RODRíGUEZ, C. Exploring the progression of early programmers in a set of computational thinking challenges via clickstream analysis. **IEEE Transactions on Emerging Topics in Computing**, v. 8, n. 1, p. 256–261, 2020.

ESTEVES, A. M. da S.; SANTANA, A. L. M.; LYRA, R. Use of augmented reality for computational thinking stimulation through virtual. *In*: SYMPOSIUM ON VIRTUAL AND AUGMENTED REALITY (SVR), 21., 2019, Rio de Janeiro. **Anais** […]. Rio de Janeiro: IEEE, 2019. p. 102–106.

FANCHAMPS, N.; SLANGEN, L.; SPECHT, M.; HENNISSEN, P. The effect on computational thinking using sra-programming: Anticipating changes in a dynamic problem environment. **IEEE Transactions on Learning Technologies**, v. 15, n. 2, p. 213–222, 2022.

FANCSALI, C.; KLEVAN, S.; MIRAKHUR, Z. Making research practice partnerships work: An assessment of the maker partnership. *In*: CONFERENCE ON RESEARCH IN EQUITABLE AND SUSTAINED PARTICIPATION IN ENGINEERING, COMPUTING, AND TECHNOLOGY (RESPECT), 6., 2021, on-line. **Anais** […]. *[S. l.]*: IEEE, 2021. p. 1–6.

FANG, X. The teaching innovation and research of python programming in financial and economic universities. *In*: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE & EDUCATION (ICCSE), 15., 2020, on-line. **Anais** […]. *[S. l.]*: IEEE, 2020. p. 600–603.

FARIAS, E. J.; CARVALHO, W. V. de; MATOS, M. E. G. de; RODRIGUES, G.; CASTRO, J. M.; SANTOS, A. D. dos. Pensamento computacional e a ação computacional por ensino remoto: Um relato de experiência de uso do appinventor em meio a pandemia de covid-19. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 31., 2020, Natal. **Anais** [...]. Natal: SBC, 2020. p. 1523–1532.

FARIAS, E. J. P.; LOPES, P. R.; CARVALHO, W. V. de; PORFÍRIO, E. M. Análise da adoção de pensamento computacional no contexto escolar brasileiro: Um mapeamento sistemático da literatura. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 34., 2023, Passo Fundo. **Anais** [...]. Passo Fundo: SBC, 2020. p. 1625-1636.

FARIAS, F. L. de O.; MELO, E.; OLIVEIRA, J.; MADEIRA, C.; CAMPOS, A. Finance math game: Uma proposta lúdica interdisciplinar para ensino de educação financeira com scratch. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 25., 2019, Brasília. **Anais** [...]. Brasília: SBC, 2019. p. 1359–1363.

FENNELL, H. W.; LYON, J. A.; MAGANA, A. J.; REBELLO, S.; REBELLO, C. M.; PEIDRAHITA, Y. B. Designing hybrid physics labs: Combining simulation and experiment for teaching computational thinking in first-year engineering. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 49., 2019, on-line. **Anais** […]. *[S. l.]*: IEEE, 2019. p. 1–8.

FERNANDES, K. T.; ARANHA, E. H. da S.; LUCENA, M. J. N. R.; FERNANDES, G. L. de S. Developing computational thinking and reading and writing skills through an approach for creating games. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 50., 2020, on-line. **Anais** […]. *[S. l.]*: IEEE, 2020. p. 1-7.

FRANÇA, C.; SILVA, C. G. Identificação de critérios para avaliação do pensamento computacional aplicado. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 31., 2020, Natal. **Anais** [...]. Natal: SBC, 2020. p. 1493–1502.

FRANÇA, R.; TEDESCO, P. Pensamento computacional sob a perspectiva de licenciandos em computação. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 23., 2017, Recife. **Anais** [...]. Recife: SBC, 2017. p. 795–804.

FREIRE, P. Pedagogy of the oppressed. **Toward a Sociology of Education**, Routledge, p. 374–386, 2020.

FREITAS, C. A. de; PEREIRA, L. G.; NASCIMENTO, F. M. do; ALBUQUERQUE, M. A. de A.; ARAUJO, M. I. de. Impacto da inteligência artificial na avaliação acadêmica: transformando métodos tradicionais de avaliação no ensino superior. **Revista Ibero-Americana de Humanidades, Ciências e Educação**, v. 11, n. 1, p. 2736–2752, 2025.

FREITAS, M.; MORAIS, P. Possibilidade de desenvolvimento do pensamento computacional por meio do code. org: aplicado ao ensino fundamental (anos iniciais). *In*: WORKSHOP DE

INFORMÁTICA NA ESCOLA, 25., 2019, Brasília. **Anais** [...]. Brasília: SBC, 2019. p. 1219-1223.

FURBER, S. **Shut down or restart?** The way forward for computing in UK Schools. London, England: The Royal Society, 2012.

GALVÃO, T. F.; PANSANI, T. d. S. A.; HARRAD, D. Principais itens para relatar revisões sistemáticas e meta-análises: A recomendação prisma. **Epidemiologia e serviços de saúde**, SciELO Public Health, v. 24, p. 335–342, 2015.

GAMMA, E. Design patterns: elements of reusable object-oriented software. **Person Education Inc**, 1995.

GAO, P.; LU, M.; ZHAO, H.; LI, M. A new teaching pattern based on pbl and visual programming in computational thinking course. *In*: IEEE INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE & EDUCATION (ICCSE), 14., 2019, on-line. **Anais** […]. [*S. l.*]: IEEE, 2019. p. 304–308.

GERALDES, W. B. *et al.* **O pensamento computacional no ensino profissional e tecnológico**. Brasília: Universidade Católica de Brasília, 2017.

GOMES, A. J.; MENDES, A. J. À procura de um contexto para apoiar a aprendizagem inicial de programação. **Educação, Formação & Tecnologias**, v. 8, n. 1, p. 13–27, 2015.

GONZÁLEZ, M. R. Computational thinking test: Design guidelines and content validation. *In*: EDULEARN, 15., 2015, Barcelona. **Anais** [...]. Barcelona: IATED, 2015. p. 2436–2444.

GOU, P.; HAN, Y.; ZHANG, W. Diversified teaching evaluation of python programming based on obe concept. *In*: IEEE INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE & EDUCATION (ICCSE), 16., 2021, on-line. **Anais** […]. *[S. l.]*: IEEE, 2021. p. 402–406.

GREIFENSTEIN, L. Effective feedback on elementary school scratch programs. *In*: ACM CONFERENCE ON INTERNATIONAL COMPUTING EDUCATION RESEARCH, 17., 2021, Charleston. **Anais** […]. Charleston: AMC, 2021.p. 417-418.

GROVER, S.; BASU, S.; BIENKOWSKI, M.; EAGLE, M.; DIANA, N.; STAMPER, J. A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. **ACM Transactions on Computing Education (TOCE)**, New York, v. 17, n. 3, p. 1–25, 2017.

GROVER, S.; PEA, R. Computational thinking in k–12: A review of the state of the field. **Educational researcher**, Sage Publications Sage CA: Los Angeles, CA, v. 42, n. 1, p. 38–43, 2013.

GUSMÃO, A.; FRANÇA, R. Pensamento computacional em atividades de robótica pedagógica livre no ensino médio. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 25., 2019, Brasília. **Anais** [...]. Brasília: SBC, 2019. p. 1129–1133.

HADI, M. E.; ATIQOH, K. S. N. *et al.* Improving students' mathematical computational thinking using scratch program through project based learning: A development research during

pandemic covid-19. *In*: IEEE INTERNATIONAL CONFERENCE ON CYBER AND IT SERVICE MANAGEMENT (CITSM), 9., 2021, Bengkulu. **Anais** [...]. Bengkulu: IEEE, 2021. p. 1–5.

HAMIDI, A. Turning computational thinking right side up. *In*: INTERNATIONAL SYMPOSIUM ON DIGITAL TRANSFORMATION, 3., 2024, Kalmar. **Anais** [...]. Kalmar: IEEE, 2024. p. 1–5.

HEINTZ, F.; MANNILA, L.; FÄRNQVIST, T. A review of models for introducing computational thinking, computer science and computing in k-12 education. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 46., 2016, Erie. **Anais** […]. Erie: IEEE, 2016. p. 1–9.

HEVNER, A. R.; MARCH, S. T.; PARK, J.; RAM, S. Design science in information systems research. **MIS quarterly**, JSTOR, p. 75–105, 2004.

HODGES, C.; MOORE, S.; LOCKEE, B.; TRUST, T.; BOND, A. The difference between emergency remote teaching and online learning. **Educause Review**, v. 27, 2020.

HOEGH, A.; MOSKAL, B. M. Examining science and engineering students' attitudes toward computer science. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 39., 2009, Erie. **Anais** […]. Erie: IEEE, 2009. p. 1–6.

HU, C.-C.; CHEN, M.-H.; YUADI, I.; CHEN, N.-S. The effects of constructing robot-based storytelling system on college students' computational thinking skill and technology comprehension. *In*: IEEE INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY (ICACT), 24., 2022, PyeongChang. **Anais** […]. PyeongChang: IEEE, 2022. p. 496–500.

HU, C.-C.; TSENG, H.-T.; CHEN, M.-H.; IMM, A. G. P.; CHEN, N.-S. Comparing the effects of robots and iot objects on stem learning outcomes and computational thinking skills between programming-experienced learners and programming-novice learners. *In*: IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES (ICALT), 20., 2020, Tartu. **Anais** […]. Tartu: IEEE, 2020. p. 87–89.

ISVIK, A.; CATETÉ, V.; BARNES, T. Investigating the impact of computing vs pedagogy experience in novices creation of computing-infused curricula. *In*: ACM CONFERENCE ON INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION, 26., 2021, on-line. **Anais** […]. [*S. l.*]: ACM, 2021. p. 255–261.

ITO, H.; SHIMAKAWA, H.; HARADA, F. Comprehension analysis considering programming thinking ability using code puzzle. *In*: IEEE CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS (FEDCSIS), 15., 2020, Toronto. **Anais** […]. Toronto: IEEE, 2020. p. 609–618.

KALOS, M. H.; WHITLOCK, P. A. **Monte carlo methods**. [*S. l.*]: John Wiley & Sons, 2009.

KARLING, D. A.; LISBÔA, E. S. Desenvolvimento do pensamento computacional no ensino superior. **Olhares & Trilhas**, v. 21, n. 1, p. 58–69, 2019.

KELLEHER, C.; PAUSCH, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. **ACM Computing Surveys (CSUR)**, New York, v. 37, n. 2, p. 83–137, 2005.

KITCHENHAM, B. *et al.* **Guidelines for performing systematic literature reviews in software engineering**. [*S. l.*]: Keele, UK, 2007.

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, P. **Evidence-based software engineering and systematic reviews**. v. 4. [*S. l.*]: CRC press, 2015.

KOHLER, L. P. de A.; MATTOS, M. M.; LOPES, M. C.; FRONZA, L.; SILVEIRA, H. U. C. da; FIBRANTZ, G.; ROSA, V. L. da; SON, L. H. L. Análise dos resultados de um estudo de caso aplicando pensamento computacional no ensino fundamental com foco na produção de algoritmos. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 27., 2021, on-line. **Anais** [...]. [*S. l.*]: SBC, 2021. p. 106–115.

KORKMAZ, Ö.; ÇAKIR, R.; ÖZDEN, M. Y. A validity and reliability study of the computational thinking scales (cts). **Computers in human behavior**, Elsevier, v. 72, p. 558–569, 2017.

LANZILOTTI, R.; PICCINNO, A.; ROSSANO, V.; ROSELLI, T. Social robot to teach coding in primary school. *In*: IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES (ICALT), 21., 2021, on-line. **Anais** […]. *[S. l.]*: IEEE, 2020. p. 102–104.

LI, X.; GU, C. Teaching reform of programming basic course based on spoc blended teaching method. *In*: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE & EDUCATION (ICCSE), 15., 2020, on-line. **Anais** […]. *[S. l.]*: IEEE, 2020. p. 411–415.

LIMA, R. A. de; SÁ, E. F. de; PORTO, A. P.; RODRIGUES, R. L.; SILVA, J. A. da. Relato de experiência sobre a criação de um clube de desenvolvimento de jogos com foco em habilidades de pensamento computacional. *In*: CONGRESSO SOBRE TECNOLOGIAS NA EDUCAÇÃO, 4., 2019, Recife. **Anais** [...]. Recife: SBC, 2019. p. 106–115.

LIN, C.-C.; CHAO, P.-Y.; LIN, E.-T.; TZENG, H.-L. Exploring the role of visual programming activities in computational thinking. *In*: IEEE INTERNATIONAL COGNITIVE CITIES CONFERENCE (IC3), 1., 2018, Okinawa. **Anais** [...]. Okinawa: IEEE, 2018. p. 135–138.

LIU, R.; LUO, F.; ISRAEL, M. What do we know about assessing computational thinking? a new methodological perspective from the literature. *In*: ACM CONFERENCE ON INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION, 26., 2021, on-line. **Anais** [...]. [*S. l.*]: ACM, 2021. p. 269–275.

LIU, Y.; ROJAS, J. Evaluation of the root robot system and curriculum to improve computational thinking in chinese children. *In*: IEEE R10 HUMANITARIAN TECHNOLOGY CONFERENCE (R10-HTC), 2019, Depok. **Anais** [...]. Depok: IEEE, 2019. p. 126–131.
LIUKAS, L. **Hello Ruby**: adventures in coding. v. 1. [*S. l.*]: Feiwel & Friends, 2015.

LOCKWOOD, T. **Design thinking**: Integrating innovation, customer experience, and brand value. [*S. l.*]: Simon and Schuster, 2010.

LUCKESI, C. C. **Avaliação da aprendizagem**: componente do ato pedagógico. [*S. l.*]: São Paulo: Cortez, 2011.

LUCKESI, C. C. **Avaliação em educação**: questões epistemológicas e práticas. [*S. l.*]: Cortez, 2022.

LUDOVICO, L. A.; MALCHIODI, D.; ZECCA, L. A multimodal lego®-based learning activity mixing musical notation and computer programming. *In*: ACM SIGCHI INTERNATIONAL WORKSHOP ON MULTIMODAL INTERACTION FOR EDUCATION, 1., 2017, New York. **Anais** [...]. New York: ACM, 2017. p. 44-48. Disponível em: https://doi.org/10.1145/3139513.3139519. Acesso em: 19 out. 2022.

LUNA, S. M. M. Manual práctico para el diseño de la escala likert. **Xihmai**, v. 2, n. 4, 2007.
MAIORANA, F.; GIORDANO, D.; MORELLI, R. Quizly: A live coding assessment platform for app inventor. *In*: IEEE BLOCKS AND BEYOND WORKSHOP (BLOCKS AND BEYOND), 1., 2015, Atlanta. **Anais** [...]. Atlanta: IEEE, 2015. p. 25–30.

MARCHISIO, M.; MARGARIA, T.; SACCHET, M. Automatic formative assessment in computer science: Guidance to model-driven design. *In*: ANNUAL COMPUTERS, SOFTWARE, AND APPLICATIONS CONFERENCE (COMPSAC), 44., 2020, Madrid. **Anais** [...]. Madrid: IEEE, 2020. p. 201–206.

MARIN, V. J.; PEREIRA, T.; SRIDHARAN, S.; RIVERO, C. R. Automated personalized feedback in introductory java programming moocs. *In*: IEEE INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE), 33., 2017, San Diego. **Anais** [...]. San Diego: IEEE, 2017. p. 1259–1270.

MARQUES, D. L.; COSTA, L. F. S.; SILVA, M. A. de A.; REBOUÇAS, A. D. D. S. Atraindo alunos do ensino médio para a computação: Uma experiência prática de introdução à programação utilizando jogos e python. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 17., 2011, Aracaju. **Anais** [...]. Aracaju: SBC, 2011. p. 1138–1147.

MARTINS-PACHECO, L. H.; WANGENHEIM, C. A. G. von; ALVES, N. da C. Assessment of Computational Thinking in K-12 Context: Educational Practices, Limits and Possibilities-A Systematic Mapping Study. *In*: INTERNATIONAL CONFERENCE ON COMPUTER SUPPORTED EDUCATION (CSEDU 2019), 11., 2019, Heraklion. **Anais** [...]. Heraklion: IEEE, 2019. p. 292–303.

MEC. **Anuário Brasileiro da Educação Básica** - Todos pela Educação. 2021. Disponível em: https://todospelaeducacao.org.br/wordpress/wp-content/uploads/2021/07/Anuario_21final.pdf. Acesso em: 19 out. 2022.

MEERBAUM-SALANT, M. A. O.; BEN-ARI, M. M. Learning computer science concepts with scratch. **Computer Science Education**, Routledge, v. 23, n. 3, p. 239–264, 2013. Disponível em: https://doi.org/10.1080/08993408.2013.832022. Acesso em: 19 out. 2022.

MIN, W.; FRANKOSKY, M. H.; MOTT, B. W.; ROWE, J. P.; SMITH, A.; WIEBE, E.; BOYER, K. E.; LESTER, J. C. Deepstealth: Game-based learning stealth assessment with deep neural networks. **IEEE Transactions on Learning Technologies**, v. 13, n. 2, p. 312–325, 2019.

MITAPPINVENTOR. **Acerca do Scratch**. 2022. Disponível em: https://appinventor.mit.edu/about-us. Acesso em: 23 out. 2022.

MOHER, D. *et al.* Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. **International journal of surgery**, Elsevier, v. 8, n. 5, p. 336–341, 2010.

MORENO-LEÓN, J.; ROBLES, G. Dr. scratch: A web tool to automatically evaluate scratch projects. *In*: WORKSHOP IN PRIMARY AND SECONDARY COMPUTING EDUCATION, 10., 2015, Londres. **Anais** [...]. Londres: Wipsce, 2015. p. 132–133.

MÜHLING, A.; RUF, A.; HUBWIESER, P. Design and first results of a psychometric test for measuring basic programming abilities. *In*: WORKSHOP IN PRIMARY AND SECONDARY COMPUTING EDUCATION, 10., 2015, Londres. **Anais** [...]. Londres: Wispce, 2015. p. 2–10.

MUNOZ, R.; VILLARROEL, R.; BARCELOS, T. S.; RIQUELME, F.; QUEZADA, A.; BUSTOS-VALENZUELA, P. Developing computational thinking skills in adolescents with autism spectrum disorder through digital game programming. **IEEE Access**, v. 6, p. 63880–63889, 2018.

NAGUMO, H.; OOMORI, Y.; TAKEMURA, Y. Mutual improvement between teaching materials and assessment tools for k-12 programming education. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 51., 2021, on-line. **Anais** […]. [*S. l.*]: IEEE, 2021. p. 1–8.

NAKAMURA, T.; KAWASAKI, T. Computer science unplugged for developing computational thinking and mathematical thinking. *In*: IEEE INTERNATIONAL JOINT CONFERENCE ON INFORMATION, MEDIA AND ENGINEERING (IJCIME), 1., 2021, Fuzhou. **Anais** […] Fuzhou: IEEE, 2019. p. 305–308.

NASCIMENTO, E. S. d. O.; OLIVEIRA, C. H.; CASTRO, M. F. de. Uso da inteligência artificial no ensino e avaliação do pensamento computacional: um mapeamento sistemático da literatura. **Revista Novas Tecnologias na Educação**, v. 22, n. 1, p. 295–307, 2024.

NASIR, J.; NORMAN, U.; BRUNO, B.; DILLENBOURG, P. When positive perception of the robot has no effect on learning. *In*: IEEE INTERNATIONAL CONFERENCE ON ROBOT AND HUMAN INTERACTIVE COMMUNICATION (RO-MAN), 29., 2020, on-line. **Anais** […]. [*S. l.*]: IEEE, 2020. p. 313–320.

NG, A. K.; ATMOSUKARTO, I.; CHEOW, W. S.; AVNIT, K.; YONG, M. H. Development and implementation of an online adaptive gamification platform for learning computational thinking. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 51., 2021, on-line. **Anais** […]. [*S. l.*]: IEEE, 2021. p. 1–6.

NUNES, D. J. Ciência da computação na educação básica. **Jornal da Ciência**, v. 9, n. 9, 2011.
NUNEZ, N. A.; CORNEJO-MEZA, G.; SÁNCHEZ, S. A. Comparing computational thinking skills of engineering students in urban and rural areas of peru. *In*: IEEE ANDESCON, 1., 2020, Equador. **Anais** […]. Equador: IEEE, 2020. p. 1–5.

OLIVEIRA, A. L. S.; ANDRADE, W. L.; GUERRERO, D. D. S.; MELO, M. R. A. How

do bebras tasks explore algorithmic thinking skill in a computational thinking contest? *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 51., 2021, on-line. **Anais** […]. [*S. l.*]: IEEE, 2021. p. 1–7.

OLIVEIRA, C. M.; PEREIRA, R. Desenvolvimento do pensamento computacional no ensino superior em ciência da computação. *In*: WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 8., 2019, Brasília. **Anais** [...]. Brasília: SBC, 2019. p. 1502.

OLIVEIRA, C. M.; PEREIRA, R. Coleta de evidências do exercício do pensamento computacional no ensino superior em computação: um artefato de apoio. *In*: SIMPÓSIO BRASILEIRO DE EDUCAÇÃO EM COMPUTAÇÃO, 3., 2023, on-line. **Anais** [...]. [*S. l.*]: SBC, 2023. p. 300-309.

OLIVEIRA, E.; BITTENCOURT, R.; TRINDADE, R. Designing and evaluating a computational thinking course for k-12 brazilian educators. *In*: WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 8., 2019, Brasília. **Anais** [...]. Brasília: SBC, 2019. p. 1094.

OLIVEIRA, E. C.; CORREIA, R. C.; BITTENCOURT, R. A. Evaluating a computational thinking and computing attitudes instrument for educational purposes. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 5., 2021, Viena. **Anais** […]. Viena: IEEE, 2021. p. 748–754.

ONG, S. L.; LING, J. P. W. Low-cost educational robotics car promotes stem learning and 21 st century skills. *In*: IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT, AND LEARNING FOR ENGINEERING (TALE), 4., 2020, Takamatsu. **Anais** [...]. Takamatsu: IEEE, 2020. p. 467–473.

OOMORI, Y.; TSUKAMOTO, H.; NAGUMO, H.; TAKEMURA, Y.; IIDA, K.; MONDEN, A.; MATSUMOTO, K.-i. Algorithmic expressions for assessing algorithmic thinking ability of elementary school children. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 49., 2019, on-line. **Anais** […]. *[S. l.]*: IEEE, 2019. p. 1–8.

OROZCO-GARCIA, L.; GONZALEZ, C.; MONTANO, J.; MONDRAGON, C.; TOBAR-MUNOZ, H. A formative assessment tool to support computational thinking in the classroom. *In*: IEEE INTERNATIONAL CONFERENCE ON VIRTUAL REALITY AND VISUALIZATION (ICVRV), 1., 2019, Hong Kong. **Anais** […]. Hong Kong: IEEE, 2019. p. 185–188.

OTA, G.; MORIMOTO, Y.; KATO, H. Ninja code village for scratch: Function samples/function analyser and automatic assessment of computational thinking concepts. *In*: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 1., 2016, Cambridge. **Anais** […]. Cambridge: IEEE, 2016. p. 238–239.

PALTS, T.; PEDASTE, M. A model for developing computational thinking skills. **Informatics in Education**, Vilnius University Institute of Data Science and Digital Technologies, v. 19, n. 1, p. 113–128, 2020.

PAPERT, S. **Mindstorms**: children, computers, and powerful ideas. [*S. l.*]: Basic Books, 1980.

PARMAR, D.; LIN, L.; DSOUZA, N.; JOERG, S.; LEONARD, A. E.; DAILY, S. B.; BABU, S. How immersion and self-avatars in vr affect learning programming and computational thinking in middle school education. **IEEE Transactions on Visualization and Computer Graphics**, v. 29, n. 8, p. 3698-3713, 2022.

PEDERSEN, B. K. M. K.; ANDERSEN, K. E.; KÖSLICH, S.; SHERZAI, F.; NIELSEN, J. *et al.* Towards playful learning and computational thinking—developing the educational robot bricko. *In*: IEEE INTEGRATED STEM EDUCATION CONFERENCE (ISEC), 1., 2018, Princeton. **Anais** […]. Princeton: IEEE, 2018. p. 37–44.

PÉREZ, A. D. F.; VALLADARES, G. M. Development and assessment of computational thinking: A methodological proposal and a support tool. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 2., 2018, Santa Cruz de Tenerife. **Anais** […]. Porto: IEEE, 2020. p. 787–795.

PERRENOUD, P. **Dez novas competências para ensinar**. Porto Alegre: Artmed, 2015.
PHETSRIKRAN, T.; MASSAGRAM, W.; PHOKA, T.; HARFIELD, A. A feasibility study of arducation bot: An educational robotics and mobile application kit for computational thinking skills. *In*: IEEE INTERNATIONAL COMPUTER SCIENCE AND ENGINEERING CONFERENCE (ICSEC), 22., 2018, Hohhot. **Anais** […].Hohhot: IEEE, 2018. p. 1–6.
PIAGET, J. **The child's conception of the world**. [*S. l.*]: Littlefield, Adams, 1960.

PIMENTEL, M.; FILIPPO, D.; SANTORO, F. M. Design science research: fazendo pesquisas científicas rigorosas atreladas ao desenvolvimento de artefatos computacionais projetados para a educação. **Metodologia de Pesquisa em Informática na Educação**: Concepção da Pesquisa. Porto Alegre: SBC, 2019.

PIRES, F.; LIMA, F. M. M.; MELO, R.; BERNARDO, J. R. S.; FREITAS, R. de. Gamification and engagement: Development of computational thinking and the implications in mathematical learning. *In*: IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES (ICALT), 19., 2019, Maceió. **Anais** […]. Maceió: IEEE, 2019. p. 362–366.

POMBO, N.; LAMAS, D. Game-based learning for young children: A case study. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 6., 2022, Tunis. **Anais** […]. Tunis: IEEE, 2022. p. 133–138.

PROMPOLMAUENG, W.; WETMAHA, A.; JAMSRI, P. A game development to promote computational thinking. *In*: IEEE INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING (ICITEE), 13., 2021, Chiang Mai. **Anais** […].Chiang Mai: IEEE, 2021. p. 116–121.

RAABE, A. L. A.; ZORZO, A. F.; FRANGO, I.; RIBEIRO, L.; GRANVILLE, L.; SALGADO, L.; CRUZ, M.; BIGOLIN, N.; CAVALHEIRO, S.; FORTES, S. **Referenciais de formação em computação**: Educação básica. Porto Alegre: Sociedade Brasileira de Computação, 2017.

RAHMAN, M. M.; SHARKER, M. H.; PAUDEL, R. An effective approach to teach an introductory computer science course with computational thinking and flow-chart based visual programming. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 50., 2020, on-line. **Anais** […]. *[S. l.]*: IEEE, 2020. p. 1-7.

RAMADHANI, N. R.; MULYANTO, A.; NIWANPUTRI, G. S. Designing interaction and user interface of computational thinking digital game for children using user-centered design approach. *In*: INTERNATIONAL CONFERENCE ON ADVANCE INFORMATICS: CONCEPTS, THEORY AND APPLICATIONS (ICAICTA), 7., 2020, Tokoname. **Anais** […]. Tokoname: IEEE, 2020. p. 1–6.

REIMER, Y. J.; COE, M.; BLANK, L. M.; BRAUN, J. Effects of professional development on programming knowledge and self-efficacy. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 48., 2018, San Jose. **Anais** […]. San Jose: IEEE, 2018. p. 1–8.

RELKIN, E.; BERS, M. Techcheck-k: A measure of computational thinking for kindergarten children. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 5., 2021, Viena. **Anais** […]. Viena: IEEE, 2021. p. 1696–1702.

ROBERT, C. P.; CASELLA, G.; CASELLA, G. **Monte Carlo statistical methods**. v. 2. [*S. l.*]: Springer, 1999.

ROMÁN-GONZÁLEZ, M.; MORENO-LEÓN, J.; ROBLES, G. Combining assessment tools for a comprehensive evaluation of computational thinking interventions. **Computational thinking education**, Springer, Singapore, p. 79-98, 2019.

ROMÁN-GONZÁLEZ, M.; PÉREZ-GONZÁLEZ, J.-C.; JIMÉNEZ-FERNÁNDEZ, C. Which cognitive abilities underlie computational thinking? criterion validity of the computational thinking test. **Computers in Human Behavior**, Elsevier, v. 72, p. 678–691, 2017.

ROWE, G.; WRIGHT, G. Expert opinions in forecasting: the role of the delphi technique. **Principles of forecasting**: A handbook for researchers and practitioners, Springer, p. 125–144, 2001.

SANDRI, M. *et al.* **Telegram bots in science education**: Applications in the context of astrophysics. New Perspectives in Science Education, 2025. Disponível em: https://conference.pixel-online.net/files/npse/ed0014/FP/9760-ESTR7156-FP-NPSE14.pdf. Acesso em: 19 ago. 2025.

SANTIAGO, C. P.; PASSOS, B. S.; NOBRE, K. A.; STEDILE, A. M. de A.; COSTA, T. M. da; LIMA, J. R. C.; AQUINO, F. J. A. de; MENEZES, J. W. M. **Um MOOC de Pensamento Computacional para Alunas do Ensino Médio**. 2024. Disponível em: https://cynthiapinheiro.ifce.edu.br/index.php/2024/11/12/um-mooc-de-pensamento-computacional-para-alunas-do-ensino-medio/. Acesso em: 19 ago. 2025.

SANTO, A. D.; FARAH, J. C.; MARTÍNEZ, M. L.; MORO, A.; BERGRAM, K.; PUROHIT, K.; FELBER, P.; GILLET, D.; HOLZER, A. Promoting computational thinking skills in non-computer-science students: Gamifying computational notebooks to increase student engagement. **IEEE Transactions on Learning Technologies**, v. 15, n. 3, p. 392–405, 2022.

SANTOS, F. A.; SEGUNDO, P.; TELVINA, M. Codeteacher: Uma ferramenta para correção automática de trabalhos acadêmicos de programação em java. *In*: WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6., 2017, Fortaleza. **Anais** [...]. Fortaleza: SBC, 2017. p. 1152.

SANTOS, P. S.; ARAUJO, L. G. J.; BITTENCOURT, R. A. A mapping study of computational thinking and programming in brazilian k-12 education. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 48., 2018, San Jose. **Anais** […]. San Jose: IEEE, 2018. p. 1–8.

SARMENTO, A.; FARIAS, M.; SANTOS, H. Relato de experiência do pibid: Promovendo o ensino de computação de forma interdisciplinar com português no ensino fundamental. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 23., 2017, Recife. **Anais** [...]. Recife: SBC, 2017. p. 313–322.

SAWADA, C.; TERAZONO, M.; HAGINO, T.; HATTORI, T. An analysis of k-12 programming education utilizing the minecraft metaverse amidst the covid-19 pandemic. *In*: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 56., 2025, Pittsburgh, EUA. **Anais** [...]. Pittsburgh: ACM, 2025. p. 1615–1616.

SBC. Sociedade Científica sem fins lucrativos. **Manifesto da SBC pela Inserção de Computação na Educação Básica**. 2018.

Disponível em: https://www.sbc.org.br/noticias/10-slideshow-noticias/ 2079-manifesto-da-sbc-pela-insercao-de-computacao-na-educacao-basica. Acesso em: 19 out. 2019.

SBCOPENLIB. **Sobre a SBC OpenLib**. 2022. Disponível em: https://sol.sbc.org.br/index.php/indice/about. Acesso em: 10 out. 2022.

SCAICO, P. D.; LIMA, A. A. de; AZEVEDO, S.; SILVA, J. B. B. da; RAPOSO, E. H.; ALENCAR, Y.; MENDES, J. P.; SCAICO, A. Ensino de programação no ensino médio: Uma abordagem orientada ao design com a linguagem scratch. **Revista Brasileira de Informática na Educação**, v. 21, n. 2, p. 92, 2013.

SCHNEIDER, G.; BERNARDINI, F.; BOSCARIOLI, C. Teaching ct through internet of things in high school: Possibilities and reflections. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 50., 2020, on-line. **Anais** […]. *[S. l.]*: IEEE, 2020. p. 1-8.

SCHOEFFEL, P.; MOSER, P.; VARELA, G.; DURIGON, L.; ALBUQUERQUE, G. C. de; NIQUELATTI, M. Uma experiência no ensino de pensamento computacional para alunos do ensino fundamental. *In*: WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 4., 2015, Maceió. **Anais** [...]. Maceió: SBC, 2019. p. 1474.

SCRATCH. **Acerca do Scratch**. 2022. Disponível em: https://scratch.mit.edu/about. Acesso em: 23 out. 2022.

SCRIVEN, M. The methodology of evaluation. social science education consortium. **Publication**, v. 110, p. 61, 1966.

SHOAIB, H.; CARDELLA, M.; MADAMANCHI, A.; UMULIS, D. An investigation of undergraduates' computational thinking in a sophomore-level biomedical engineering course. *In*: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 49., 2019, on-line. **Anais** […]. *[S. l.]*: IEEE, 2019. p. 1–4.

SHYAMALA, C.; VELAYUTHAM, C. S.; PARAMESWARAN, L. Teaching computational thinking to entry-level undergraduate engineering students at amrita university. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 1., 2017, Atenas. **Anais** […]. Atenas: IEEE, 2017. p. 1731–1734.

SILVA, A.; MELO, R. F. de; SOUSA, R. P. de; NASCIMENTO, K. Estimulando o pensamento computacional em alunos do ensino médio com o uso do scratch for arduino. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 25., 2019, Brasília. **Anais** [...]. Brasília: SBC, 2019. p. 783–791.

SILVA, N.; SANTOS, I.; ORLEANS, L. Ensino inclusivo de pensamento computacional: um relato de experiência. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 27., 2019, Belém. **Anais** [...]. Belém: SBC, 2019. p. 81–90.

SINGH, M.; SINGH, A.; KAPOOR, A. Exploring ways to develop computational thinking in pre-service teachers. **This paper has appeared in The Academic**, v. 2, n. 6, p. 386–394, 2024.

SOBREIRA, P. de L.; ABIJAUDE, J. W.; VIANA, H. D. G.; SANTIAGO, L. M. S.; GUEMHIOUI, K. E.; WAHAB, O. A.; GREVE, F. Usability evaluation of block programming tools in iot contexts for initial engineering courses. *In*: IEEE WORLD CONFERENCE ON ENGINEERING EDUCATION (EDUNINE), 1., 2020, on-line. **Anais** […]. [*S. l.*]: IEEE, 2020. p. 1–5.

SOFTEX. Mercado de trabalho e formação de mão de obra em ti. **Cadernos Temáticos do Observatório**, p. 7-127, jan. 2018. Disponível em: file:///Users/usuario/Downloads/cadernos_tematico_mercado_de_trabalho.pdf. Acesso em: 19 out. 2022.

SOUSA, L. de L.; FARIAS, E. J.; CARVALHO, W. V. de. Programação em blocos aplicada no ensino do pensamento computacional: Um mapeamento sistemático. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 31., 2020, Natal. **Anais** [...]. Natal: SBC, 2020. p. 1513–1522.

SOUZA, D.; GOULART, M.; GUARDA, G.; GOULART, I. Lightbot logicamente: um game lúdico amparado pelo pensamento computacional e a matemática. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 24., 2018, Fortaleza. **Anais** [...]. Fortaleza: SBC, 2018. p. 61–69.

SRISANGNGAM, P.; DECHSURA, C. Stem education activities development to promote computational thinking's students. *In*: INTERNATIONAL STEM EDUCATION CONFERENCE (ISTEM-ED), 5., 2020, Sapporo. **Anais** [...]. Sapporo: IEEE, 2020. p. 103-105.

STERBINI, A.; TEMPERINI, M. Automatic correction of c programming exercises through unit-testing and aspect-programming. *In*: INTERNATIONAL CONFERENCE ON EDUCATION AND INFORMATION SYSTEMS: TECHNOLOGIES AND APPLICATIONS (EISTA'04), 1., 2004, Orlando. **Anais** [...]. Orlando: IIIS, 2004.

SWAID, S.; SUID, T. Computational thinking education: Who let the dog out? *In*: IEEE INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND

COMPUTATIONAL INTELLIGENCE (CSCI), 1., 2019, Las Vegas. **Anais** [...]. Las Vegas: IEEE, 2019. p. 788–792.

TANIOKA, H.; YANO, R. Development and evaluation of quizzes aimed at quantifying computational thinking. *In*: INTERNATIONAL CONGRESS ON ADVANCED APPLIED INFORMATICS (IIAI-AAI), 10., 2021, on-line. **Anais** [...]. [*S. l.*]: IEEE, 2021. p. 188–191.
TAVARES, T. E.; MARQUES, S. G.; CRUZ, M. K. da. Plugando o desplugado para ensino de computaçao na escola durante a pandemia do sars-cov-2. *In*: SIMPÓSIO BRASILEIRO DE EDUCAÇÃO EM COMPUTAÇÃO, 1., 2021, Jataí. **Anais** [...]. Jataí: SBC, 2021. p. 263–271.

TERRAGNI, V.; GIACAMAN, N. A system-level testing framework for automated assessment of programming assignments allowing students object-oriented design freedom. **International Conference on Software Testing**, 2025. Disponível em: https://valerio-terragni.github.io/assets/pdf/terragni-icstEdu-2025.pdf. Acesso em: 12 ago. 2025.

THIOLLENT, M. J. M.; COLETTE, M. M. Pesquisa-ação, universidade e sociedade. **Revista Mbote**, v. 1, n. 1, p. 042–066, 2020.

TISSENBAUM, M.; SHELDON, J.; ABELSON, H. From computational thinking to computational action. **Communications of the ACM**, New York, v. 62, n. 3, p. 34–36, 2019.

TRINDADE, G. M.; FERNANDES, F. P.; BARBOSA, L. S. de O.; SOUZA, D. R. de. O uso do jogo digital minecraft para estimular o pensamento computacional e a aprendizagem colaborativa no ensino fundamental i: Um relato de experiência. *In*: WORKSHOP DE INFORMÁTICA NA ESCOLA, 26., 2020, on-line. **Anais** [...]. [*S. l.*]: SBC, 2020. p. 219-228.

UNPLUGGED, C. **Computer Science without a computer**. 2022. Disponível em: https://www.csunplugged.org/en/. Acesso em: 15 out. 2022.

UTESCH, M. C.; FAIZAN, N. D.; KRCMAR, H.; HEININGER, R. Pic2program-an educational android application teaching computational thinking. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 4., 2020, Porto. **Anais** […]. Porto: IEEE, 2020. p. 1493-1502.

VANACORE, K.; PANKIEWICZ, M.; BAKER, R. Unpacking the impact of generative ai feedback: Divergent effects on student performance and self-regulated learning. **ResearchGate**, 2024. Disponível em: https://www.um.es/ead/red/46/moreno_robles.pdf. Acesso em: 19 out. 2022.

VIANNA, H. M. Avaliação educacional: problemas gerais e formação do avaliador. **Estudos em Avaliação Educacional**, Fundação Carlos Chagas, v. 25, n. 60, p. 74–84, 2014.

VIHAVAINEN, A.; PAKSULA, M.; LUUKKAINEN, M. Extreme apprenticeship method in teaching programming for beginners. *In*: TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 42., 2011, Dallas. **Anais** […]. Dallas: ACM, 2011. p. 93-98.

VINAYAKUMAR, R.; SOMAN, K.; MENON, P. Alg-design: facilitates to learn algorithmic thinking for beginners. *In*: International Conference on Computing, Communication and

Networking Technologies (ICCCNT), 9., 2018, Bengaluru. **Anais** […].Bengaluru: IEEE, 2018. p. 1–6.

VLAHU-GJORGIEVSKA, E.; VIDENOVIK, M.; TRAJKOVIK, V. Computational thinking and coding subject in primary schools: Methodological approach based on alternative cooperative and individual learning cycles. *In*: IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT, AND LEARNING FOR ENGINEERING (TALE), 2., 2018, Wollongongo. **Anais** [...]. Wollongongo: IEEE, 2018. p. 77–83.

WANG, L.; SY, A.; LIU, L.; PIECH, C. Learning to represent student knowledge on programming exercises using deep learning. **International Educational Data Mining Society**, p. 324-329, 2017. Disponível em: https://educationaldatamining.org/EDM2017/proc_files/papers/paper_129.pdf. Acesso em: 19 out. 2022.

WANG, Y.; ZHANG, Y.; MAO, A.; WANG, J.; LI, N. The research of programming teaching in primary school on the cultivation of computational thinking. *In*: INTERNATIONAL CONFERENCE OF EDUCATIONAL INNOVATION THROUGH TECHNOLOGY (EITT), 9., 2020, Porto. **Anais** [...]. Porto: IEEE, 2020. p. 250–255.

WANGENHEIM, C. G. V.; HAUCK, J. C.; DEMETRIO, M. F.; PELLE, R.; ALVES, N. da C.; BARBOSA, H.; AZEVEDO, L. F. Codemaster–automatic assessment and grading of app inventor and snap! programs. **Informatics in Education**, Vilnius University Institute of Mathematics and Informatics, Lithuanian, v. 17, n. 1, p. 117–150, 2018.

WANGENHEIM, C. G. von; NUNES, V. R.; SANTOS, G. D. D. Ensino de computaçao com scratch no ensino fundamental–um estudo de caso. **Revista Brasileira de Informática na Educação**, v. 22, n. 03, p. 115, 2014.

WEINTROP, D.; SHEPHERD, D. C.; FRANCIS, P.; FRANKLIN, D. Blockly goes to work: Block-based programming for industrial robots. *In*: BLOCKS AND BEYOND WORKSHOP (B.B.), 1., 2017, Raleigh. **Anais** [...]. Raleigh: IEEE, 2017. p. 29–36.

WENG, X.; WONG, G. K. Integrating computational thinking into english dialogue learning through graphical programming tool. *In*: IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT, AND LEARNING FOR ENGINEERING (TALE), 1., 2017, Hong Kong. **Anais** [...].Hong Kong: IEEE, 2018. p. 320–325.

WIERUCH, R. **The road to React**. [*S. l.*]: Robin Wieruch, 2020.

WING, J. Research notebook: Computational thinking - what and why. **The link magazine**, v. 6, p. 20–23, 2011.

WING, J. M. Computational thinking. **Communications of the ACM**, New York, v. 49, n. 3, p. 33–35, 2006.

WING, J. M. Computational thinking. **Commun. ACM**, Association for Computing Machinery, New York, v. 49, n. 3, p. 33–35, mar. 2006. Disponível em: https://doi.org/10.1145/1118178.1118215. Acesso em: 19 out. 2022.

WING, J. M. Computational thinking and thinking about computing. **Philosophical Transactions of the Royal Society A**: Mathematical, Physical and Engineering Sciences, v. 366, n. 1881, p. 3717–3725, 2008.

WING, J. M.; STANZIONE, D. Progress in computational thinking, and expanding the hpc community. **Communications of the ACM**, ACM New York, v. 59, n. 7, p. 10–11, 2016.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [*S. l.*]: Springer Science & Business Media, 2012.

WONG, G. K.; JIANG, S. Computational thinking education for children: Algorithmic thinking and debugging. *In*: IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT, AND LEARNING FOR ENGINEERING (TALE), 2., 2018, Wollongongo. **Anais** [...]. Wollongongo: IEEE, 2018. p. 328–334.

XU, C.; FENG, Z.; QI, P.; SUN, Y. An automatic analysis tool based on computational thinking for blockpy programs. *In*: INTERNATIONAL WIRELESS COMMUNICATIONS AND MOBILE COMPUTING (IWCMC), 21., 2020, Abu Dhabi. **Anais** [...]. Abu Dhabi: IEEE, 2020. p. 2045–2050.

YADAV, A.; MAYFIELD, C.; ZHOU, N.; HAMBRUSCH, S.; KORB, J. T. Computational thinking in elementary and secondary teacher education. **ACM Transactions on Computing Education (TOCE)**, New York, v. 14, n. 1, p. 1-16, 2014.

YANG, K.-H.; LIN, H.-Y. Exploring the effectiveness of learning scratch programming with code. org. *In*: IEEE INTERNATIONAL CONGRESS ON ADVANCED APPLIED INFORMATICS (IIAI-AAI), 8., 2019, Toyama. **Anais** [...]. Toyama: IEEE, 2019. p. 1057–1058.

YETT, B.; SNYDER, C.; HUTCHINS, N.; BISWAS, G. Exploring the Relationship Between Collaborative Discourse, Programming Actions, and Cybersecurity and Computational Thinking Knowledge [Badanie zwia˛zków mie˛dzy dyskursem opartym na współpracy, działaniami programistycznymi a bezpieczeństwem cybernetycznym i wiedza˛ z zakresu mys´lenia komputacyjnego]. *In*: IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT, AND LEARNING FOR ENGINEERING (TALE), 4., 2020, Takamatsu. **Anais** [...]. Takamatsu: IEEE, 2020. p. 213-220.

YULIANA, I.; OCTAVIA, L. P.; SUDARMILAH, E.; MATAHARI, M. Introducing computational thinking concept learning in building cognitive capacity and character for elementary student. *In*: IEEE INTERNATIONAL SYMPOSIUM ON COMMUNICATIONS AND INFORMATION TECHNOLOGIES (ISCIT), 19., 2019, Ho Chi Minh. **Anais** [...].Ho Chi Minh: IEEE, 2019. p. 549–554.

ZAPATA-CÁCERES, M.; MARTÍN-BARROSO, E.; ROMÁN-GONZÁLEZ, M. Computational thinking test for beginners: Design and content validation. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 4., 2020, Porto. **Anais** […]. Porto: IEEE, 2020. p. 1905–1914.

ZAPATA-CÁCERES, M.; MARTÍN-BARROSO, E.; ROMÁN-GONZÁLEZ, M. Collaborative

game-based environment and assessment tool for learning computational thinking in primary school: a case study. **Transactions on Learning Technologies**, IEEE, v. 14, n. 5, p. 576–589, 2021.

ZARKADOULA, K.; XENOS, M. Evaluating usability and educational effectiveness of a serious game for programmers using alternative interfaces and types of activities. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 6., 2022, Tunis. **Anais** […]. Tunis: IEEE, 2022. p. 644–651.

ZHANG, S.; CUI, C. Implementing blended learning in k-12 programming course: Lesson design and student feedback. *In*: INTEGRATED STEM EDUCATION CONFERENCE (ISEC), 16., 2021, on-line. **Anais** […]. [*S. l.*]: IEEE, 2021. p. 38-44.

ZHANG, S.; WONG, G. K.; PAN, G. Computational thinking test for lower primary students: Design principles, content validation, and pilot testing. *In*: IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT, AND LEARNING FOR ENGINEERING (TALE), 5., 2021, Hubei. **Anais** [...]. Hubei: IEEE, 2021. p. 345–352.

ZIMMERMAN, D. W.; ZUMBO, B. D. Relative power of the wilcoxon test, the friedman test, and repeated-measures anova on ranks. **The Journal of Experimental Education**, Taylor & Francis, v. 62, n. 1, p. 75–86, 1993.

ZINN, B. The future of technical education: Current research focuses on educational evaluation, digitalization, interdisciplinarity, and the shortage of educators. **Journal of Technical Education (JOTED)**, v. 13, n. 1, p. 1–8, 2025.

ZORZO, A. F.; NUNES, D.; MATOS, E.; STEINMACHER, I.; ARAUJO, R. M.; CORREIA, R.; MARTINS, S. **Referenciais de Formação para os Cursos de Graduação em Computação**. [*S. l.*]: SBC, 2017.

APÊNDICE A – PC E A AÇÃO COMPUTACIONAL

A.1 Pensamento Computacional e a Ação Computacional por Ensino Remoto: Um relato de experiência de uso do *AppInventor* em meio a pandemia de COVID-19

A.2 Abstract e Resumo

A.2.1 Abstract

This paper presents a proposal for teaching Computational Thinking, combining the use of Computational Action to Design Thinking. We evaluated it in a course taught to students in the first semester of Computer Science, which also used the concepts of Emergency Remote Education. At the end of the course, each student developed a mobile application, which had as its central theme: " Fighting the pandemic of COVID 19 ". The applications implemented in the course were designed using Design Thinking principles and developed on the block programming platform App Inventor. The results indicate that after the course, the students showed a visible increase in knowledge about Computational Thinking and improved their motivation to learn more about the subject.

A.2.2 Resumo

Este artigo apresenta uma proposta de ensino de Pensamento Computacional (PC) mesclando o uso de Ação Computacional e *Design Thinking*. Ela foi avaliada em um curso ministrado para alunos de primeiro semestre de Ciência da Computação que também usou os conceitos do Ensino Remoto Emergencial. Ao final, cada aluno desenvolveu um aplicativo para *smartphone* que tinha como tema central: "O combate a pandemia do COVID 19". Os aplicativos implementados foram projetados usando *Design Thinking* e desenvolvidos na plataforma de programação em blocos *App Inventor*. Os resultados do curso indicam que os estudantes apresentaram um visível aumento no conhecimento a respeito do PC e na motivação em aprender mais sobre o assunto.

A.3 Introdução

Com o aumento da inserção da tecnologia digital nos diversos setores da economia mundial, a procura por profissionais de Ciência da Computação é intensa. De acordo com a

Fundação *Code.org*, por exemplo, até o fim de 2020 haverá mais de um milhão de vagas para programadores em aberto apenas nos Estados Unidos. A necessidade da formação de um grande número de profissionais dessa área continua, portanto, em evidência.

No entanto, o Curso de Ciência da Computação conta historicamente com elevadas taxas de desistência nos semestres iniciais (VIHAVAINEN *et al.*, 2011), o que indica que o ensino dos conteúdos básicos do Curso não é tarefa fácil. Alguns pesquisadores que já abordaram o tema atribuem tal cenário ao foco excessivo na sintaxe das linguagens de programação (CASTOLDI; POLINARSKI, 2009) e ao uso de materiais didáticos pouco interessantes (GOMES; MENDES, 2015).

Neste contexto, o Pensamento Computacional (PC), pode ser entendido como uma abordagem prática na resolução de problemas, habilidade de projetar sistemas e entender a relação entre o pensamento humano e os conceitos que fundamentam a Ciência da Computação (ROMÁN-GONZÁLEZ *et al.*, 2017). Dessa forma, ensinar PC para alunos iniciantes de Ciência da Computação apresenta-se como uma estratégia interessante, visto que pode melhorar a capacidade de reconhecer, analisar, compreender e solucionar problemas.

O aprendizado do PC requer que o estudante adquira um conjunto de conhecimentos, desenvolva habilidades e internalize atitudes (WANG *et al.*, 2017). Uma abordagem metodológica voltada especificamente para o ensino de PC é a Ação Computacional, que consiste na proposta de atividades de ensino situadas na realidade das vidas e comunidades dos estudantes (TISSENBAUM *et al.*, 2019).

Este trabalho detalha a concepção, aplicação e avaliação de uma abordagem de ensino de PC embasada na metodologia da Ação Computacional. Essa abordagem foi avaliada durante um curso de construção de aplicativos para *smartphone* usando o *APP Inventor* seguindo princípios do *Design Thinking*. 19 estudantes de primeiro semestre de Ciência da Computação participaram do curso. O curso foi avaliado - em caráter exploratório - quantitativamente, e foram encontrados resultados positivos com relação ao seu impacto no conhecimento e na motivação dos estudantes acerca do PC, bem como na boa qualidade dos códigos realizados pelos estudantes ao final do curso. Espera-se que os resultados detalhados neste artigo possam ajudar a inspirar pesquisadores e professores a analisar ou mesmo adotar o ensino de PC em semestre iniciais de Ciência da Computação, utilizando metodologias emergentes como a Ação Computacional.

O curso estava planejado para ser realizado de forma presencial. Com a pandemia de COVID-19, o mesmo foi adaptado para a modalidade remota. Os ajustes realizados seguiram

os conceitos e ferramentas propostos pelo Ensino Remoto de Emergência (ERE) (HODGES *et al.*, 2020). Ressalta-se que o formato no qual o curso foi efetivado abre espaço para a discussão de ampliá-lo para um público mais amplo no futuro já que diminuíram as restrições físicas (i.e., laboratório) de execução do mesmo.

O restante deste trabalho está organizado na seguinte forma: a seção A.4 apresenta a metodologia da pesquisa utilizada neste trabalho e em A.5 são apresentados e discutidos os resultados obtidos. A seção A.6 apresenta pesquisas relacionadas ao presente trabalho e, por fim, a seção B.8 apresenta as conclusões extraídas, destacando os principais resultados obtidos, ameaças a validade e propondo alguns trabalhos futuros.

A.4 Metodologia

Este artigo relata a análise de um caso prático de uso da abordagem proposta em um curso de ensino de Pensamento Computacional. A inovação aqui proposta consiste em integrar conceitos de Ação Computacional e princípios de *Design Thinking* no ensino de PC para estudantes iniciantes de Ciência da Computação. A Ação Computacional, propõe que, ao mesmo tempo em que aprendem sobre computação, os jovens também devem ter oportunidade de fazer com que a computação tenha impacto direto em suas vidas e em suas comunidades(TISSENBAUM *et al.*, 2019). Já o *Design Thinking* é um processo iterativo não linear usado para entender os usuários, desafiar suposições, redefinir problemas e criar, prototipar e testar soluções inovadoras (LOCKWOOD, 2010). Ao mesclar essas duas estratégias, podemos tornar o ensino do PC mais inclusivo e motivador para os jovens aprendizes. A avaliação deste curso baseou-se em uma estratégia quantitativa de pesquisa, de caráter exploratório. Esta seção detalha os procedimentos metodológicos adotados.

A.4.1 Questões de Pesquisa

Para guiar a concepção do curso, foram definidas as seguintes questões de pesquisa:

Q1. A abordagem proposta é capaz de impactar no conhecimento dos estudantes sobre o pensamento computacional?

Q2. A motivação dos participantes em conhecer mais sobre a pensamento computacional é modificada após o uso dessa abordagem no curso?

Q3. Qual a qualidade dos códigos apresentados pelos participantes do curso?

Desta forma, as atividades propostas e a avaliação realizada visaram coletar indícios para investigar estas perguntas levantadas.

A.4.2 Contexto e Perfil dos participantes

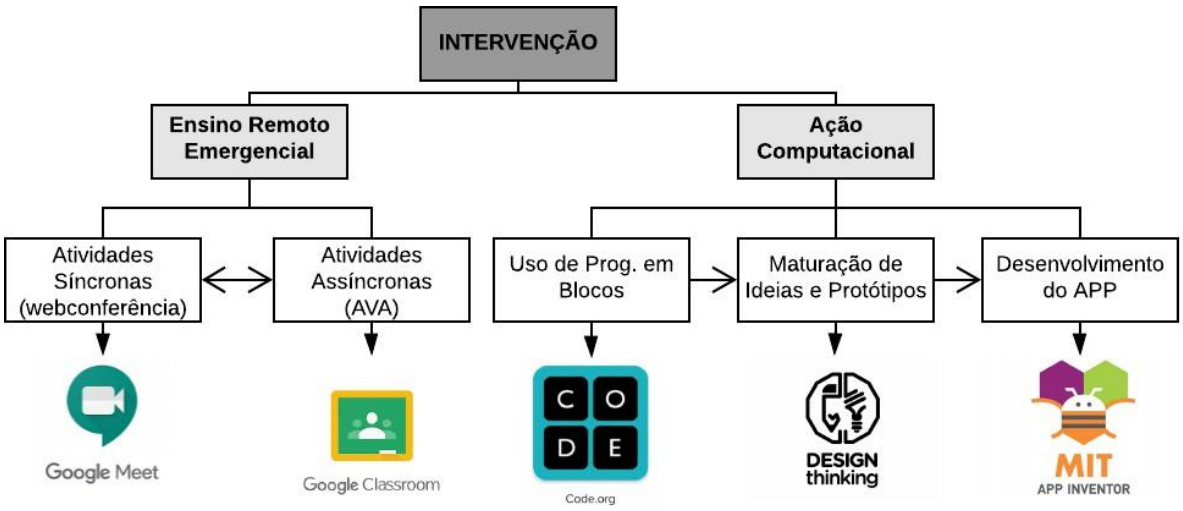
O curso foi ministrado para estudantes de uma turma do primeiro semestre do Curso de Ciência da Computação (CC) de uma universidade pública brasileira de uma cidade do interior do Brasil. É importante salientar que o curso foi aplicado antes que os estudantes tivessem contato com os conceitos básicos de programação no curso de CC. Inscreveram-se inicialmente para o curso um total de 25 estudantes, contudo, apenas 19 estudantes passaram por todas as etapas e apenas esses serão levados em consideração para a avaliação dos resultados. A média de idade dos estudantes que finalizaram o curso é igual a 18,23 anos, com desvio padrão de 0,6. Sendo 68,42% de homens e 31,58% de mulheres.

Como dito na Introdução, inicialmente o curso tinha um escopo presencial, com o uso de um laboratório de informática para o apoio ao desenvolvimento dos APPs. Contudo, por conta da pandemia do Covid-19, o projeto de intervenção precisou passar por algumas adaptações com base nos conceitos do Ensino Remoto Emergencial (HODGES *et al.*, 2020). No ERE, as aulas remotas de maneira síncrona são priorizadas (como no ensino presencial), usando, por exemplo, sistemas de webconferência. Os materiais são disponibilizados e as atividades e avaliações seguem usando um ambiente virtual de aprendizagem (AVA) de forma assíncrona. O uso do ERE, que inicialmente não estava planejado, possibilitou a utilização de tecnologias digitais em todo o curso, o que oferece uma perspectiva mais escalável da proposta para as suas próximas versões.

A.4.3 Intervenção

O curso teve duração de oito semanas e foi ministrado nos meses de Maio e Junho de 2020. O organograma mostrado na Figura 46 apresenta graficamente como o Ensino Remoto Emergencial e a Ação Computacional foram utilizados no projeto de intervenção proposto. Usou-se *Google Meet* para as aulas síncronas, o *Google Classroom* como AVA, o *Code.org* como plataforma inicial para programação em blocos, o *Design Thinking* para organização das ideias e prototipação dos projetos e o *APP Inventor* como plataforma de programação dos aplicativos para *smartphone*. A Figura 46 ressalta a relação entre os objetivos de cada tecnologia, o fluxo de aplicação e as ferramentas usadas.

Figura 46 – Arquitetura e ferramentas utilizadas no projeto de intervenção



Fonte: elaborada pelo autor (2025).

O Quadro 1 apresenta um resumo de como se deu a organização do curso e descreve os encontros síncronos que aconteceram durante as oito semanas. Como pode-se observar, foram feitos dois encontros em cada semana, com duração média de duas horas cada. Todas as aulas foram gravadas e dispostas para a turma no *Google Classroom*, assim como os slides e materiais de apoio, como vídeos externos, links para sites e artigos. O AVA também foi usado para o recebimentos das tarefas que eram propostas no decorrer da semana, e para receber o código do aplicativo desenvolvido por cada estudante.

Na primeira semana, apresentou-se aos estudantes a formatação do curso e os conceitos iniciais sobre o que é o PC. Nas semanas 2 e 3, foi proposto aos estudantes que solucionassem desafios que tinham relação direta com o aprendizado dos principais aspectos do PC (Pensamento Algorítmico, Abstração, Divisão de Problemas e Reconhecimento de Padrões) (GROVER; PEA, 2013). Neste período, com o objetivo de familiarizar os estudantes com a sintaxe de uma linguagem de programação em blocos, foi proposta a participação em um curso disponibilizado na plataforma *Code.org*. Em resumo, as 3 semanas iniciais, viu-se as bases teórico-práticas requeridas para a continuação do curso.

Da semana 4 até a semana 8, o curso se concentrou na criação de um projeto e na implementação de aplicativos para *smartphone*. Seguindo os conceitos da Ação Computacional, foi conduzida a construção de protótipos em torno da temática: “Enfrentamento da Pandemia causada pelo Covid-19". Nos projetos, cada um dos estudantes desenvolveu e apresentou um aplicativo para ajudar no enfrentamento da pandemia do novo coronavírus, inserido na realidade

Quadro 1 – Organização do curso

	CONTEÚDO	OBJETIVOS
SEM. 1	Aula 1 – Apresentação do Curso	• Apresentar o formato do curso para os estudantes • Entender o conceito de pensamento computacional
	Aula 2 - Conceitos de Pensamento Computacional	
SEM. 2	Aula 3 - Programação em Blocos e desafios práticos de pensamento computacional.	• Entendimento teórico e prática da programação em blocos • Adquirir competências básicas do pensamento computacional
	Aula 4 - Demonstração prática de programação em blocos	
SEM. 3	Aula 5 - Resolução dos desafios da semana 2 e tira dúvida sobre Code.org	• Sedimentar o entendimento e a prática do pensamento computacional • Esclarecer conceitos básicos do pensamento algorítmico
	Aula 6 - Apresentação de conceitos de Algoritmos (Sequência, Seleção e Repetição)	
SEM. 4	Aula 7 - *Computational Action*	• Entendimento do conceito do *Computational Action* • Apresentar a ferramenta de desenvolvimento do APP
	Aula 8 - Conceitos de App Inventor	
SEM. 5	Aula 9 - APP Inventor - Switch de Ferramentas	• Conhecer quais ferramentas o APP inventor disponibiliza • Facilitar a organização das ideias
	Aula 10 - *Design Thinking*	
SEM. 6	Aula 11 - Reunião de Criação de Ideias para APP	• Possibilitar a maturação das ideias e a criação dos protótipos
	Aula 12- Workshop de co-criação	
SEM. 7	Aula 13 - Recebimento de protótipos	• Realizar o desenvolvimento dos Aplicativos
	Aula 14 - Tira dúvidas sobre a produção dos APP	
SEM. 8	Aula 15 - Reunião de finalização dos APP	
	Aula 16 - Entrega dos Aplicativos e fechamento do curso	

Fonte: elaborado pelo autor (2025).

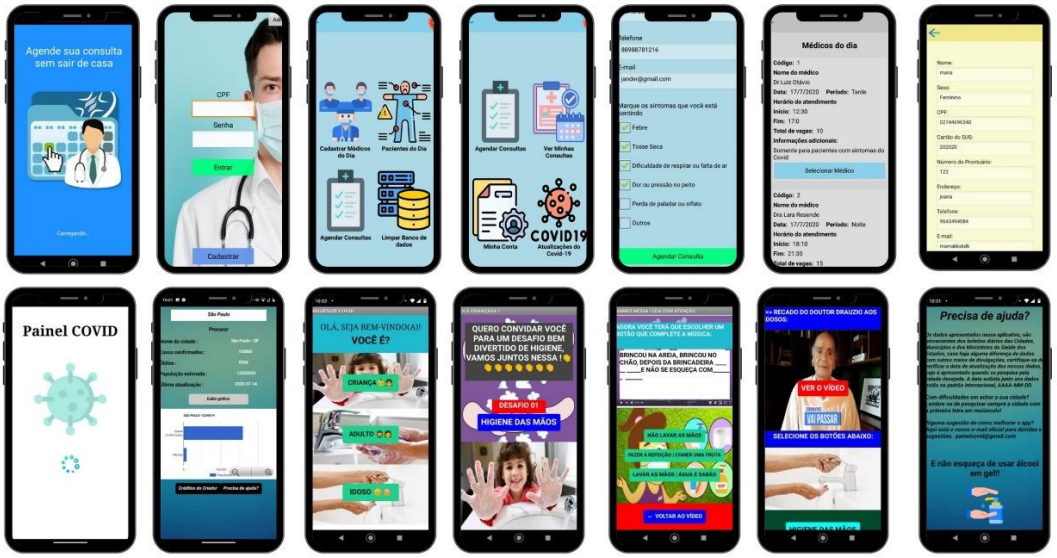
das suas comunidades. A Figura 47 mostra as telas de alguns aplicativos desenvolvidos pelos estudantes no decorrer do curso.

A.4.4 Materiais e Métodos

Para investigar as questões de pesquisa **Q1** e **Q2**, utilizou-se um questionário adaptado de (OLIVEIRA *et al.*, 2019), que foi concebido para avaliar o conhecimento e a motivação de estudantes acerca do PC. Este questionário - cujas respostas são expressas em Escala Likert - foi aplicado no início e no fim do curso com o objetivo de detectar se o conhecimento ou a motivação dos estudantes foi alterada.

Para a análise dos dados, é importante entender que o questionário é composto por uma categoria relacionada ao conhecimento do estudante (logo, relacionado à Q1) que é “definição e sala de aula” e outras três categorias relacionadas à motivação do estudante (logo, relacionado à Q2), que são: “confiança", “interesse"e “utilidade".

Figura 47 – Telas dos projetos desenvolvidos pelos alunos



Fonte: elaborada pelo autor (2025).

- **Definição e sala de aula** inclui itens que forneciam definições de PC e da visão do seu uso futuro em sala de aula (YADAV *et al.*, 2014);
- A **confiança** dos estudantes nas suas capacidades de aprender habilidades relacionadas a Ciência da Computação (HOEGH; MOSKAL, 2009);
- O **interesse** dos estudantes na Ciência da Computação (HOEGH; MOSKAL, 2009);
- A crença dos estudantes na **utilidade** de aprender Ciência da Computação (HOEGH; MOSKAL, 2009).

Como forma de reforçar a resposta à **Q2**, foi aplicado, ao fim do curso, um questionário motivacional que também foi utilizado em (OLIVEIRA *et al.*, 2019). Neste questionário - cujas respostas também usavam Escala Likert - as perguntas estavam divididas em 4 categorias: Atenção, Relevância, Confiança e Satisfação.

- A capacidade de captar e sustentar a **atenção** dos estudantes;
- A percepção da **relevância** do conteúdo por parte dos estudantes;
- A **confiança** dos estudantes nas suas capacidades;
- A **satisfação** dos estudantes em ver seus objetivos atingidos.

Para a aplicação e coleta de dados dos questionários, utilizou-se a ferramenta *Google Forms*. Os questionários usados na pesquisa podem ser encontrados em https://cutt.ly/4dwOj8x. A confiabilidade e consistência interna dos questionários foram medidas com o auxílio do coeficiente Alfa de Cronbach (CRONBACH, 1951).

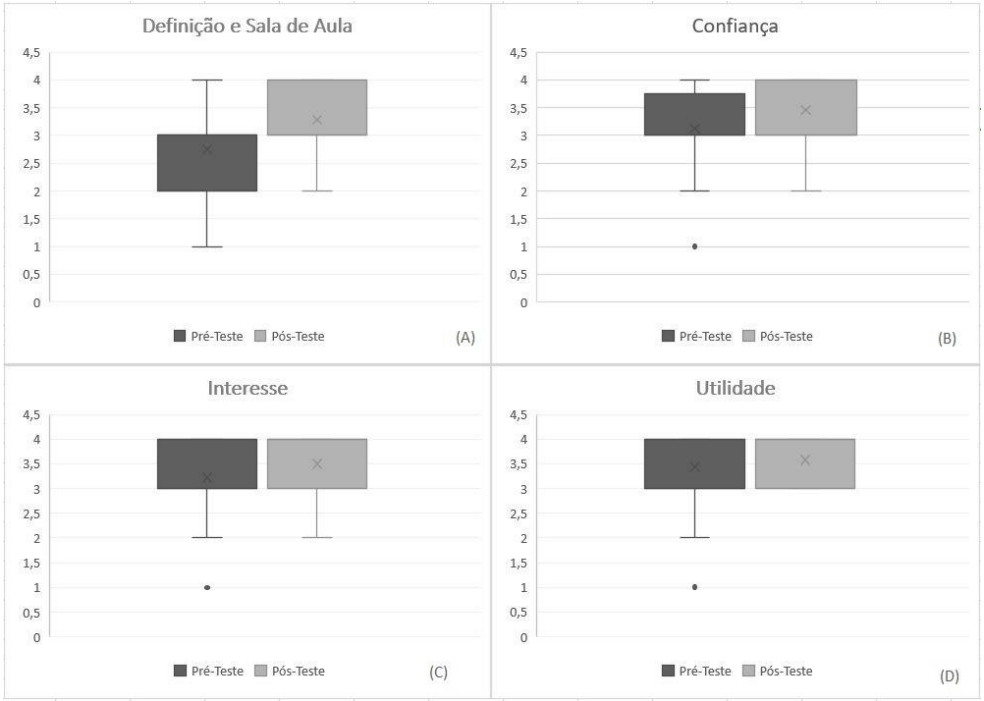
Com relação à **Q3**, o software *CodeMaster* propiciou suporte a avaliação dos códigos apresentados pelos estudantes. O *CodeMaster* é um aplicativo gratuito que permite avaliar e classificar automaticamente projetos programados com o *App Inventor* (WANGENHEIM *et al.*, 2018). A escolha de uma ferramenta de análise automática de código é importante, pois descarta qualquer tipo de subjetividade e enviesamento na avaliação. A opção pelo *CodeMaster* ocorreu também por este realizar a verificação dos códigos submetidos levando em conta os conceitos do PC. A ferramenta utiliza rubricas que analisam indiretamente as competências com base na medição de indicadores do resultado da aprendizagem do PC (WANGENHEIM *et al.*, 2018).

A.5 Resultados e Discussões

A.5.1 Conhecimento sobre o PC

A questão de pesquisa **Q1** foi investigada nos 06 itens do pré/pós-teste que referem-se à categoria Definição e Sala de Aula. Neste contexto, ressalta-se que os questionários de pré/pós-teste apresentaram um valor de Alpha Cronbach de 0,8949 e 0,9344, o que indica confiabilidade da consistência interna dos questionários, ou seja, os itens são suficientemente diferentes e medem elementos distintos (CRONBACH, 1951).

Figura 48 – Declarações de atitudes - pré e pós-teste

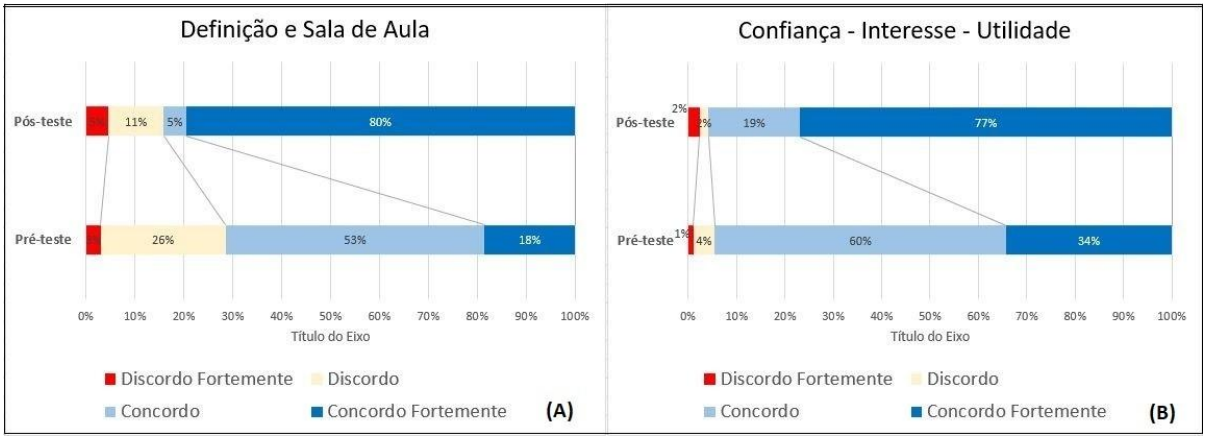


Fonte: elaborada pelo autor (2025).

Como pode-se verificar na Figura 48A, houve uma melhora no entendimento dos participantes do curso a respeito da definição e do conceito de PC. O pré-teste apresentou mediana igual a 3,0, média igual a 2,7560 e Amplitude Interquartílica (AI) igual a 1, com primeiro quartil (Q1) igual a 2 e terceiro quartil (Q3) igual a 3. Já o pós-teste, apesar de também apresentar mediana igual a 3 e uma AI igual a 1, apresentou uma média de 3,2778, Q1 igual a 3 e Q3 igual a 4. Esse resultado demonstra uma evolução de *score* na categoria avaliada. Outra informação relevante é que o pré-teste apresentou o valor mínimo para as respostas igual a 1 (que é o valor mais baixo utilizado na escala Likert adotada), demostrando pouco conhecimento de alguns participantes a respeito da definição de PC. Este resultado indica preliminarmente uma resposta positiva à Q1, necessitando entretanto de validações estáticas com um número maior de estudantes e de aplicações.

O gráfico apresentado na Figura 49A, ajuda a elucidar os resultados. Pode-se notar que houve uma migração das respostas para categorias mais altas na escala Likert adotada. No pré-teste, por exemplo, 53% das respostas foram “Concordo", já no pós-teste, a 80% das respostas foram “Concordo Fortemente".

Figura 49 – Respostas aos questionários pré/pós-teste (escala Likert)



Fonte: elaborada pelo autor (2025).

A.5.2 Motivação sobre o PC

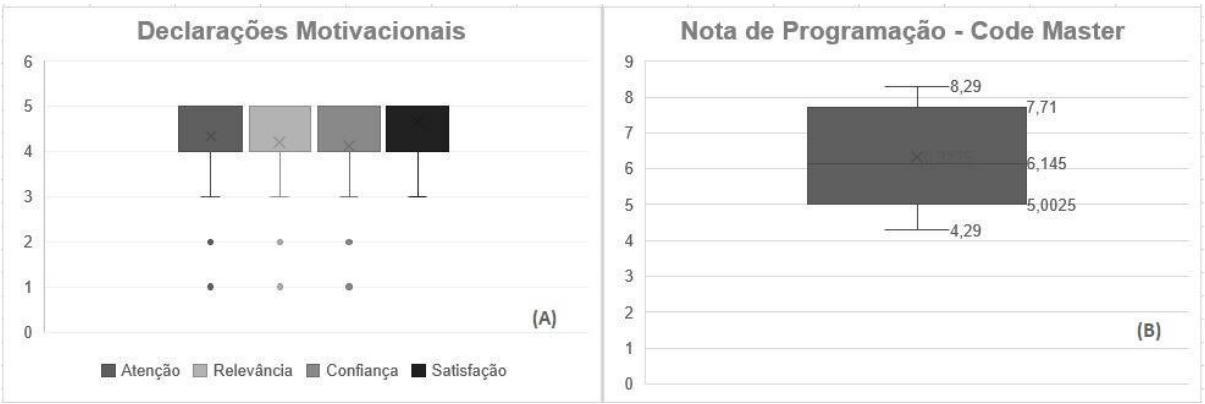
Em relação a Q2, a motivação dos estudantes foi medida por meio das categorias Confiança (08 itens), Interesse (10 itens) e Utilidade (06 itens) do questionário pré/pós teste, bem como pelos 36 itens questionário motivacional aplicado ao final do curso.

Com relação ao questionário pré/pós-teste, apesar da mediana dos resultados da

Confiança dos estudantes serem iguais a 3 em pré e pós-teste (Figura 48B), houve pequena melhora na média que passou de 3,1205 para 3,4583, e no no valor do Q3, que passou de 3,75 para 4. As avaliações de Interesse (Figura 48C) e Confiança (Figura 48D), tiveram o valor da mediana aumentada de 3 para 4. Nota-se também a ausência de *outliers* de valores baixos para os pós-testes, o que indica que nenhum estudante obteve pontuações negativamente fora do padrão. O gráfico apresentado na Figura 49B, faz uma compilação das respostas dadas aos questionários pré/pós-teste nas categorias Confiança, Interesse e Utilidade. No pré-teste 60% das respostas estavam concentradas em “Concordo", já no pós-teste, essa maior concentração se deu na categoria mais alta da escola, o “Concordo Fortemente", que obteve 77% das respostas.

O formulário de declarações motivacionais aplicado ao fim do curso, apresentou confiabilidade com um valor de Alpha Cronbach de 0,7989. Como pode ser observado na Figura 50 (A), as categorias Atenção, Relevância e Satisfação obtiveram mediana igual a 5 (que é a pontuação máxima na escala Likert utilizada para este questionário) e a categoria confiança obteve mediana igual a 4. Os resultados apresentados tanto pelo pré/pós-teste quanto pelo questionário motivacional indicam que o curso pode ter afetado positivamente a motivação dos participantes. Conjectura-se que os resultados expressivos observados - especialmente pelo questionário motivacional - tem relação com a metodologia adotada da Ação Computacional que visa especificamente em motivar os estudantes com atividades de ensino situadas na realidade das suas vidas cotidianas.

Figura 50 – Declarações motivacionais - pós-teste (A) e notas do CodeMaster (B)



Fonte: elaborada pelo autor (2025).

A.5.3 Qualidade dos códigos

A Figura 50 (B) apresenta o gráfico *Box Plot* da pontuação de programação atribuída pelo *CodeMaster* aos códigos dos APPs desenvolvidos pelos alunos no fim do curso. O gráfico, responde de maneira positiva a Q3. Nota-se que, apesar de ter sido o primeiro aplicativo desenvolvido por todos os participantes do curso, o resultado final foi positivo. A média e mediana das notas atribuídas aos códigos é igual a 6,3225 e 6,145, respectivamente, com desvio padrão igual a 1,4586, o que podem ser consideradas boas notas para um primeiro código, pois o resultado apresentado pelo *CodeMaster* possue uma escala de 0 a 10. É válido salientar que o gráfico ainda apresentou o terceiro quartil igual a 7,71 e valor máximo igual a 8,29.

Pode-se ainda afirmar que os códigos desenvolvidos pelos estudantes possuem características que remetem ao aprendizado do PC. Essa afirmação decorre do fato de o *CodeMaster* medir a complexidade dos programas com base nas várias dimensões do PC, como abstração, sincronização, paralelismo, noções algorítmicas de controle de fluxo, interatividade do usuário e representação de dados (WANGENHEIM *et al.*, 2018). A avaliação completa obtida pelos estudantes pode ser encontrada em https://bit.ly/30iIvkW.

A.6 Trabalhos Relacionados

Em (OLIVEIRA; PEREIRA, 2019), o autor apresenta uma proposta de melhorar o aprendizado do PC utilizando uma abordagem baseada na resolução de problemas reais. Essa pesquisa teve como público alvo alunos iniciantes do curso de Ciência da Computação, tentando compreender e diminuir a alta taxa de evasão do curso. Por se tratar apenas de uma proposta esse estudo não apresentou resultados.

Já (KARLING; LISBÔA, 2019) dispõe de um estudo voltado ao desenvolvimento do PC em discentes do Curso de Licenciatura em Computação, utilizado também a plataforma *App Inventor*, com a finalidade de avaliar a construção de aplicativos. Nesse estudo, optou-se pela avaliação por pares, onde os próprios alunos julgaram os trabalhos dos colegas. Adicionalmente, os alunos do 6º semestre puderam aprimorar a habilidade da docência durante 4 semanas de aulas com vinte alunos do 1º período. O estudo apresentou resultados positivos, tando para os estudantes de 1º quanto para os do 6º semestre.

A principal diferença entre os trabalhos acima citados e o presente estudo, está na metodologia do ensino do PC. Uma vez que neste trabalho, utilizou-se a Ação Computacional e o

Design Thinking para estimular o desenvolvimento de projetos de aplicativos móveis como maior significado para os estudantes. Além disso, sua aplicação se deu por meio de ensino remoto.

A.7 Considerações Finais

O ensino do PC envolve fazer com que os estudantes adquiram a habilidade de resolver problemas, projetar sistemas e entender o comportamento humano, baseando-se nos conceitos fundamentais de CC (WING, 2006a). Neste sentido, pode-se concluir que a a abordagem proposta de ensino de Pensamento Computacional com a utilização da Ação Computacional obteve resultados promissores. Essa conclusão decorre do fato de que todas as avaliações propostas apresentaram resultados satisfatórios, juntamente com a qualidade dos APPs desenvolvidos durante o processo. Em resposta as questões de pesquisa apresentadas na seção A.4.1, pode-se afirmar que:

- Os conhecimentos dos participantes em relação ao pensamento computacional foram alteradas positivamente;
- Os participantes estão motivados em conhecer mais sobre a pensamento computacional depois do curso;
- Os aplicativos desenvolvidos pelos estudantes durante o curso, apresentaram, de acordo com os instrumentos utilizados, uma boa qualidade de código.

É importante destacar que os resultados reportados são preliminares, e atendem apenas ao caráter exploratório desta pesquisa. Há limitações quanto à população e amostra, uma vez que não foram seguidos os critérios de aleatoriedade na escolha da amosta, e à ausência de um grupo de controle. Além disso, para dar maior liberdade aos alunos no momento em que estivessem respondendo aos questionários, optou-se pelo anonimato nas respostas, o que impossibilitou a rastreabilidade entre os testes individuais aplicados antes e depois do curso. Apesar destes fatores, espera-se que os dados analisados ajudem a inspirar pesquisadores a abordar o PC em seus cursos, utilizando metodologias emergentes como a Ação Computacional.

Este artigo apresentou um caso da aplicação da Ação Computacional em um curso para o semestre inicial de Ciência da Computação. A continuidade desta pesquisa pode ser contemplada com a adaptação e aplicação do curso para outros públicos-alvo. Para tanto, a estratégia planejada é incremental: aplicar para cursos de STEM (Ciências, Tecnologia, Engenharia e Matemática) exceto Computação, posteriormente estender para outros Cursos de Graduação de outras áreas. Esta estratégia está alinhada com pesquisas recentes que indicam

que o uso do PC pode transformar o comportamento de diversas disciplinas, profissões e setores (WING; STANZIONE, 2016).

APÊNDICE B – PROGRAMAÇÃO EM BLOCOS: MSL

B.1 Programação em Blocos Aplicada no Ensino do Pensamento Computacional: Um Mapeamento Sistemático

B.2 Abstract e Resumo

B.2.1 Abstract

In recent years, Computational Thinking (CT) has been gaining prominence on the world stage, and many researches point to the block programming paradigm as one of the main alternatives for teaching PC to people who have no programming experience. In this sense, this work proposes the realization of a Systematic Mapping Studies (SMS) on the use of block programming in the teaching of CT. During the SMS some important information was revealed, such as: which stages of formal education are studies most applied to, which are the most used tools, which types of assessment are adopted and in which countries this type of study is focused on.

B.2.2 Resumo

Nos últimos anos, o Pensamento Computacional (PC) vem ganhando destaque no cenário mundial e, muitas pesquisas apontam o paradigma de Programação em Blocos (PB) como uma das principais alternativas de ensino do PC para pessoas que não tenham experiência com programação. Nesse sentido, este trabalho propõe a realização de um Mapeamento Sistemático da Literatura (MSL) sobre o uso de PB no ensino do PC. No decorrer do MSL foram reveladas algumas informações importantes, como: em quais etapas da educação formal os estudos são mais aplicados, quais são as ferramentas mais usadas, quais os tipos de avaliação adotadas e em quais países esse tipo de estudo se concentra.

B.3 Introdução

Atualmente a computação está presente em todos os lugares. Nesse contexto, não é mais possível esperar que os jovens tenham contato com conceitos relacionados a esse tema apenas quando entrarem na universidade (BARR; STEPHENSON, 2011b). Os jovens de hoje irão se deparar com empregos que nem sequer existem ainda, e, provavelmente muitos deles

estarão relacionados, direta ou indiretamente, com a computação (BARR; STEPHENSON, 2011b) (SCAICO *et al.*, 2013).

Alguns conceitos relacionados a computação, embora tenham importância social, ainda são vistos como monótonos pelos alunos. (SCAICO *et al.*, 2013) observa que muitos estudos apontam que o desinteresse por temas relacionados a essa área é causado, inúmeras vezes, devido a percepção que diversos alunos tem a respeito do assunto, considerando-o chato e entediante. Como consequência da falta de interesse, poucos alunos ingressam em cursos relacionados com computação, e com isso, a necessidade por profissionais vem aumentando. Segundo o relatório técnico apresentado por (SOFTEX, 2013), o mercado de TI vai apresentar um deficit de cerca de 408 mil profissionais em 2022 no Brasil. Tal fato, traz a necessidade de ensinar conceitos do Pensamento Computacional (PC) cada vez mais cedo, além da necessidade de que novas metodologias de ensino do PC sejam propostas. Somando a isso, muitos estudos indicam que as diversas áreas do conhecimento estão sofrendo, direta ou indiretamente, influência do PC (WING, 2006b)(BUNDY, 2007b)(BARR; STEPHENSON, 2011b) (SBC, 2018).

Atualmente, a utilização do paradigma de Programação em Blocos (PB), vem apresentando resultados positivos no ensino do PC. Esse paradigma vem sendo adotado por várias metodologias, desde programação de histórias, jogos, vídeos até a programação de robôs. (WEINTROP *et al.*, 2017) (KELLEHER; PAUSCH, 2005).

Visto isso, este trabalho tem como objetivo geral fazer um levantamento do uso da Programação em Blocos (PB) juntamente com o ensino do Pensamento Computacional (PC). Para tentar responder a essa questão, um Mapeamento Sistemático da Literatura (MSL) foi realizado com o propósito de conhecer e classificar as informações existentes, a fim de fornecer dados relevantes que possam embasar novas pesquisas sobre o assunto.

O restante desse trabalho está dividido da seguinte forma: Na seção B.4 é apresentado um referencial teórico; A seção B.5 revela a metodologia utilizada no desenvolvimento do trabalho; Na seção B.6 são apresentados os resultados do trabalho, com foco em revelar as respostas para cada questão de pesquisa; Por fim, a seção B.8 apresenta as considerações finais do estudo, mostrando suas limitações e trabalhos futuros.

B.4 Referencial Teórico

B.4.1 Pensamento Computacional

Segundo, (WING, 2006b) “O PC envolve resolver problemas, projetar sistemas e entender o comportamento humano, baseando-se nos conceitos fundamentais da Ciência da Computação”. Em um outro artigo, a autora enfatiza também que a essência do PC é a abstração, e enxerga que ela é uma ferramenta mental da computação (WING, 2008).

PC é mais que um conhecimento, ele pode ser definido como uma competência. Aprender esse conceito requer que o estudante adquira, além de um conjunto de conhecimentos, desenvolva habilidades e internalize atitudes (WANG *et al.*, 2017). O processo de desenvolvimento dessa competência pode ser melhorado com a utilização de novas metodologias de ensino em combinação com ferramentas computacionais. Neste contexto, o PC, pode ser entendido como uma abordagem prática na resolução de problemas, habilidade de projetar sistemas e entender a relação entre o pensamento humano e os conceitos que fundamentam a Ciência da Computação (ROMÁN-GONZÁLEZ *et al.*, 2017).

B.4.2 Programação em Blocos

Conforme (KELLEHER; PAUSCH, 2005), a programação baseada em blocos (PB) procura abstrair a sintaxe de forma que os estudantes iniciantes mantenham seu foco apenas na semântica. Por isso, esse paradigma é um meio eficiente de atrair estudantes para o aprendizado do PC (WEINTROP *et al.*, 2017).

A PB utiliza ações de arrastar e soltar blocos gráficos ou físicos, que correspondem aos componentes de um programa escrito em linguagem textual. Cada bloco tem encaixes específicos, possibilitando que o usuário consiga visualizar onde cada um pode ser encaixado. Com isso, programas com sintaxes incorretas se tornam quase impossíveis, embora em alguns caso (a depender da ferramenta) seja possível encaixar blocos de forma errônea (WEINTROP *et al.*, 2017) (KELLEHER; PAUSCH, 2005).

B.5 Metodologia

O presente estudo trata-se de um MSL e tem a finalidade de fazer um levantamento do uso da PB no ensino do PC. Este trabalho procura seguir as diretrizes propostas em

(KITCHENHAM *et al.*, 2007).

Como suporte para a realização do mapeamento, foi utilizada a ferramenta StArt (*State of the Art through Systematic Review*), *software* desenvolvido na Universidade Federal de São Carlos (UFSCar).

B.5.1 Protocolo de Revisão

Um protocolo de revisão indica quais os passos serão seguidos para a realização do mapeamento. A criação desse documento se faz necessária para reduzir a possibilidade de viés por parte dos pesquisadores (KITCHENHAM *et al.*, 2007).

O problema de pesquisa desse trabalho é: Entender os diversos cenários em que a Programação em Blocos está sendo usada para auxiliar o ensino e a aprendizagem do Pensamento Computacional.

As questões de pesquisa propostas por este trabalho são:

QP1. Em quais etapas da educação formal os estudos se concentram?

QP2. Quais as ferramentas utilizadas no ensino do Pensamento Computacional que fazem uso de Programação em Blocos?

QP3. Quais os tipos de avaliações de aprendizado foram empregadas nos estudos?

QP4. Em quais países os estudos se concentram?

B.5.1.1 Estratégia de Busca

Foi utilizada a busca automatizada nas seguintes bibliotecas digitais: *Institute of Electrical and Electronics Engineers* (IEEE), *Web of Science*, *Association For Computing Machinery* (ACM), Scopus e *Science Direct*. Apenas artigos escritos na língua inglesa e que foram publicados no período de 2015 a 2019, foram aceitos.

A *string* de busca foi composta seguindo o protocolo PICOC (KITCHENHAM *et al.*, 2007), que divide os termos em cinco blocos distintos: *Population* (População), *Intervention* (Intervenção), *Comparison* (Comparação), *Outcome* (Resultados) e *Context* (Contexto).

Como resultado, obteve-se o seguinte termo de busca: **(“*Block language*” *OR* “*Block programming*” *OR* “*Block-based programming*”) *AND* (*“Computational thinking”*).**

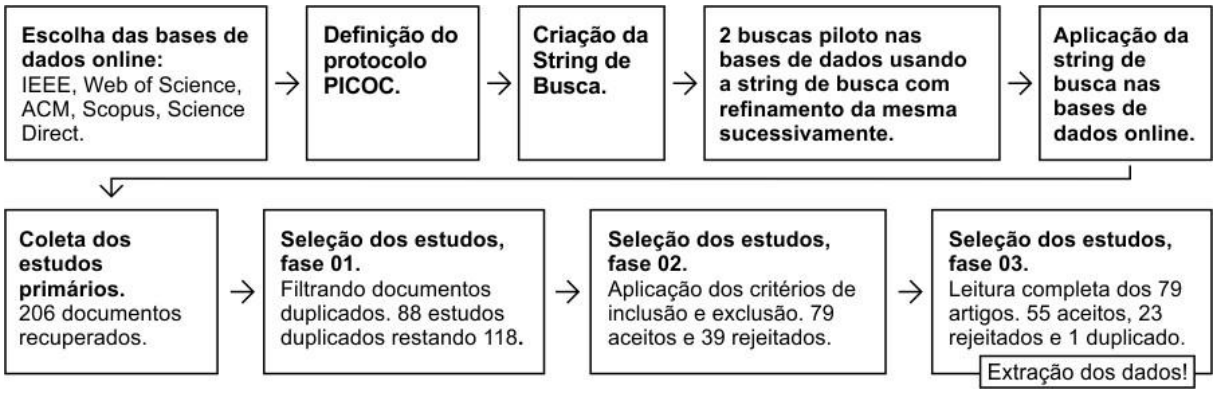
B.5.1.2 Seleção dos Estudos e Extração dos Dados

Foram propostos critérios de inclusão, exclusão e o formulário de extração dos dados. Todos descritos no protocolo de mapeamento sistemático, que foi criado antes da execução desse estudo com o intuito de sistematizar todo o processo e reduzir o viés. Os critérios de inclusão, exclusão e o formulário de extração dos dados podem ser encontrados em http://bit.ly/pc_pb.

B.5.2 Aplicação do Protocolo

A Figura 51 ilustra o processo de busca dos estudos, seleção dos trabalhos e extração dos dados. Esse processo é dividido em 5 fases principais: escolha das bases de dados online; criação da *string* de busca; coleta dos estudos; seleção dos estudos e extração dos dados. As fases secundárias também estão expressas na Figura 51.

Figura 51 – Representação da estratégia de busca e seleção dos estudos primários



Fonte: elaborada pelo autor (2025).

A *string* de busca foi executada e o resultado obtido está descrito na Tabela 7. Nela é possível observar que 206 estudos foram recuperados utilizando a *string* de busca nas bibliotecas *online*. A base de dados Scopus apresentou a maior concentração de estudos, sendo responsável por 35,4% de todos os estudos recuperados. Em contrapartida, a IEEE foi a que menos ofereceu trabalhos, tendo apenas 11,2% de artigos. Depois que o conjunto com 206 estudos primários foi recuperado, eles passaram por três fases de seleção.

Na primeira fase, os estudos duplicados foram detectados com auxílio da ferramenta StArt e excluídos. No total, 88 estudos foram identificados como duplicados sobrando 118 trabalhos. Na segunda fase, os títulos, palavras-chave e resumos de todos os trabalhos restantes foram lidos e julgados segundo os critérios de inclusão e exclusão. Nessa fase, 39 estudos foram

Tabela 7 – Estudos recuperados na busca automatizada

Base de dados	Quantidade de estudos	%
IEEE	23	11,2%
Web of Science	45	21,8%
ACM	38	18,4%
Scopus	73	35,4%
Science Direct	27	13,1%
Total:	**206**	**100%**

Fonte: elaborada pelo autor (2025).

rejeitados e 79 foram aceitos. Na terceira e última fase foi aplicado novamente os critérios de seleção, mas dessa vez com a leitura completa dos 79 artigos retantes. Nessa etapa 55 estudos foram aceitos, 23 rejeitados e 1 foi identificado como duplicado. Ainda na terceira fase de seleção, os dados necessários para responder as questões de pesquisa foram extraídos com auxílio do formulário de extração. A planilha contendo todos os 55 trabalhos selecionados e os 62 rejeitados pode ser acessada através do *link* http://bit.ly/pc_pb.

B.6 Resultados e Discussões

Nesta seção, os dados obtidos no MSL são sintetizados e analisados com foco em responder as questões de pesquisa previamente estabelecidas. Além disso, foi possível extrair algumas informações adicionais.

Pode ser observado na Figura 52, que no período estabelecido para realização desse MSL (2015 a 2019), houve um crescimento de publicações que relacionam PB no ensino do PC. Com mais trabalhos sendo publicados, é esperado que o ensino de PC utilizando PB, se torne uma prática cada vez mais comum.

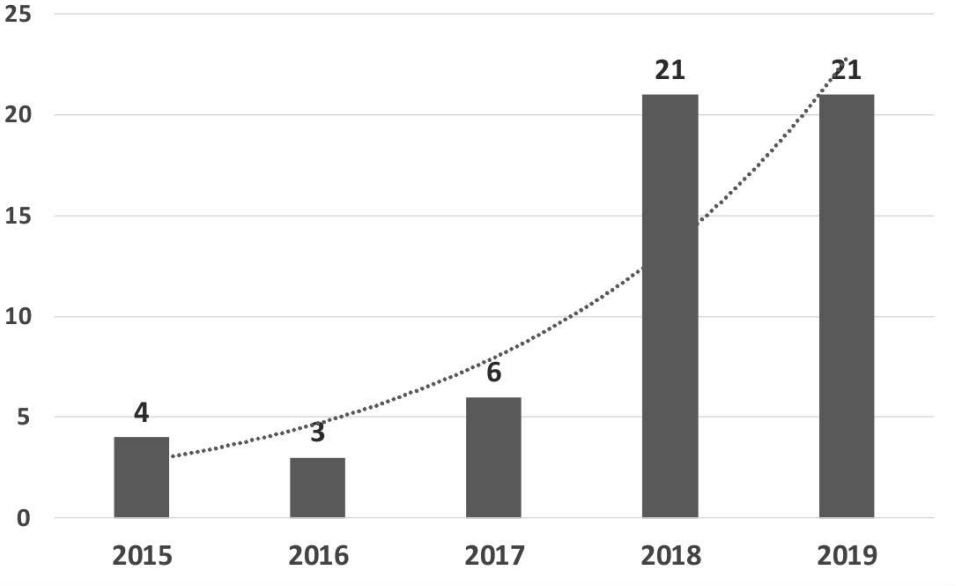
B.6.1 Respostas às Questões de Pesquisa

Esta seção apresenta as respostas para as questões de pesquisa, com base na análise dos dados extraídos dos estudos primários selecionados.

QP1- Em quais etapas da educação formal os estudos se concentram?

A Figura 53 apresenta a quantidade de estudos por etapa da educação formal. Como forma de normalizar todas as etapas, foi utilizado o padrão adotado no Brasil (pré-escola, ensino fundamental, ensino médio e ensino superior), uma vez que os países adotam classificações diferentes para as etapas educacionais. É importante dizer que vários estudos tem seu foco em mais de uma etapa educacional. Dessa forma, alguns estudos foram vinculados a mais de uma

Figura 52 – Ano de publicação dos estudos primários

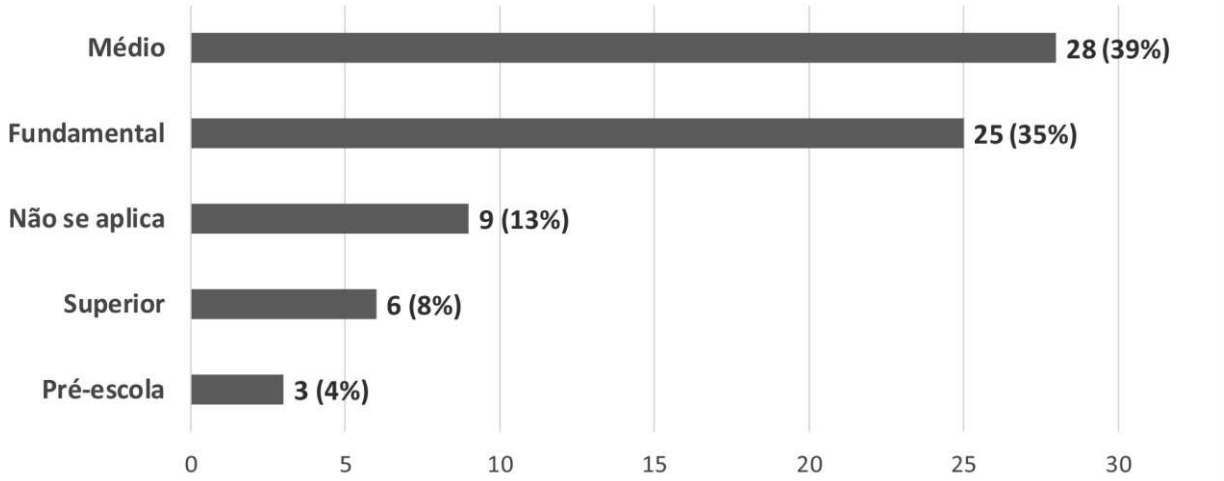


Fonte: elaborada pelo autor (2025).

categoria.

Analisando o gráfico, pode-se notar que a maioria dos estudos está concentrado no ensino fundamental e médio, o que era esperado, pois são nessas etapas da educação que o uso de PB é recomendado. A categoria 'Não se aplica' enumera os trabalhos que não vincularam seus estudos a nenhuma etapa educacional.

Figura 53 – Etapas da educação formal onde os estudos se concentram



Fonte: elaborada pelo autor (2025).

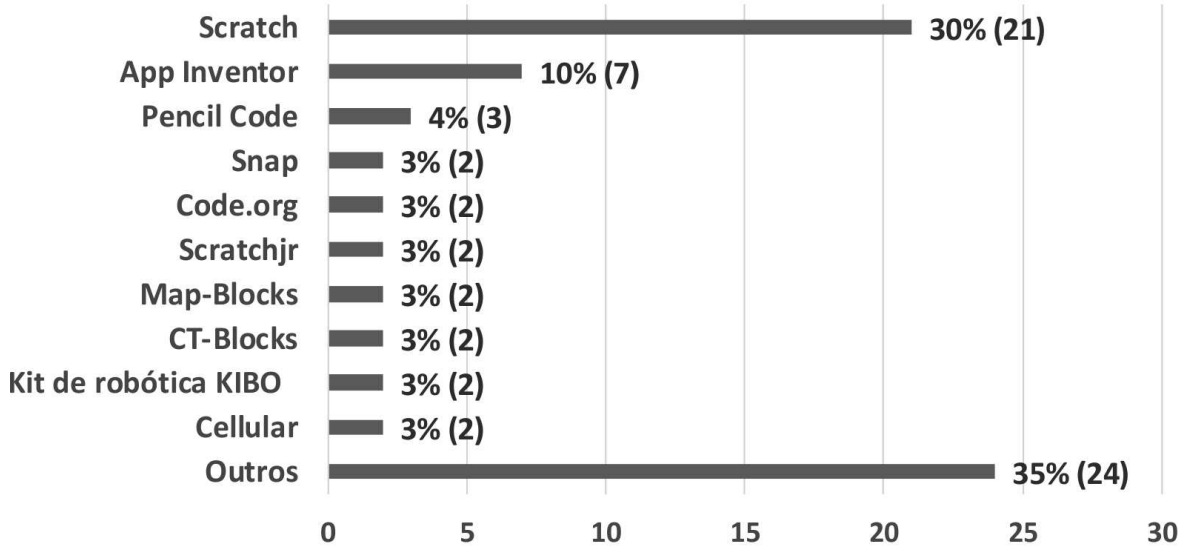
QP2- Quais as ferramentas utilizadas no ensino do Pensamento Computacional que utilizam Programação em Blocos?

Como mostrado na Figura 54, foi detectado um grande número de ferramentas

que usam o paradigma de PB. No total, foram 33 ferramentas diferentes e um trabalho que não identificou a ferramenta usada. A ferramenta mais utilizada foi o *Scratch*, sendo incluída em 23 dos trabalhos analisados, o que resultou em um percentual de 30%. Na sequência, o *App inventor* e o *Pencil Code* ficaram em segundo e terceiro lugar, respectivamente. O restante das ferramentas apareceu em apenas um ou dois trabalhos, o que demonstra uma grande pulverização no uso de ferramentas. A categoria "outros" agrupa as 24 ferramentas que apareceram em apenas um trabalho. O gráfico completo com todas as ferramentas pode ser encontrado em http://bit.ly/pc_pb.

Outra observação interessante é que 59% das ferramentas são aplicações *web*. Esse resultado é apresentado na Figura 55, e pode ser justificado pelo fato de que essas ferramentas possuem maior portabilidade e por serem de fácil distribuição.

Figura 54 – Ferramentas de programação em blocos



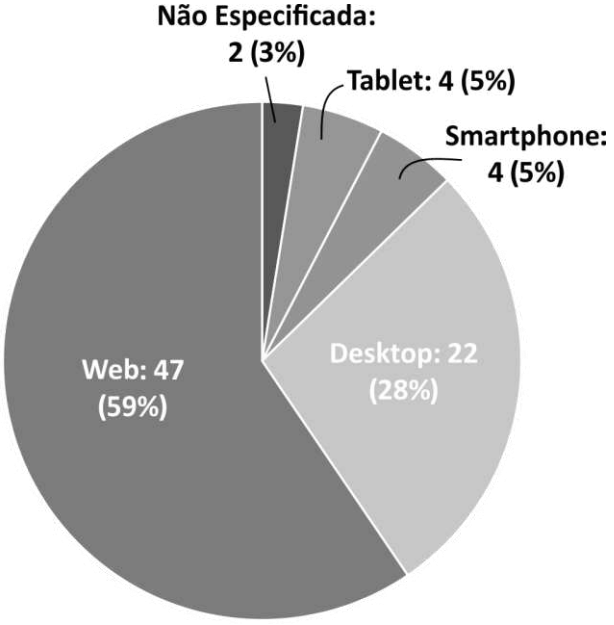
Fonte: elaborada pelo autor (2025).

QP3- Quais os tipos de avaliações de aprendizado que foram empregadas nos estudos?

De forma geral, os trabalhos apresentaram diversos tipos de avaliações. Devido essa granularidade, as avaliações foram distribuídas em categorias para que os dados pudessem ser agrupados e assim melhor representados.

A Figura 56 mostra a distribuição dos trabalhos de acordo com as categorias propostas. A categoria "Teste avaliativo" reúne o conjunto de estudos que utilizaram testes de conhecimento comumente aplicados em salas de aula, como testes escritos, práticos e análise

Figura 55 – Plataformas onde as ferramentas são executadas



Fonte: elaborada pelo autor (2025).

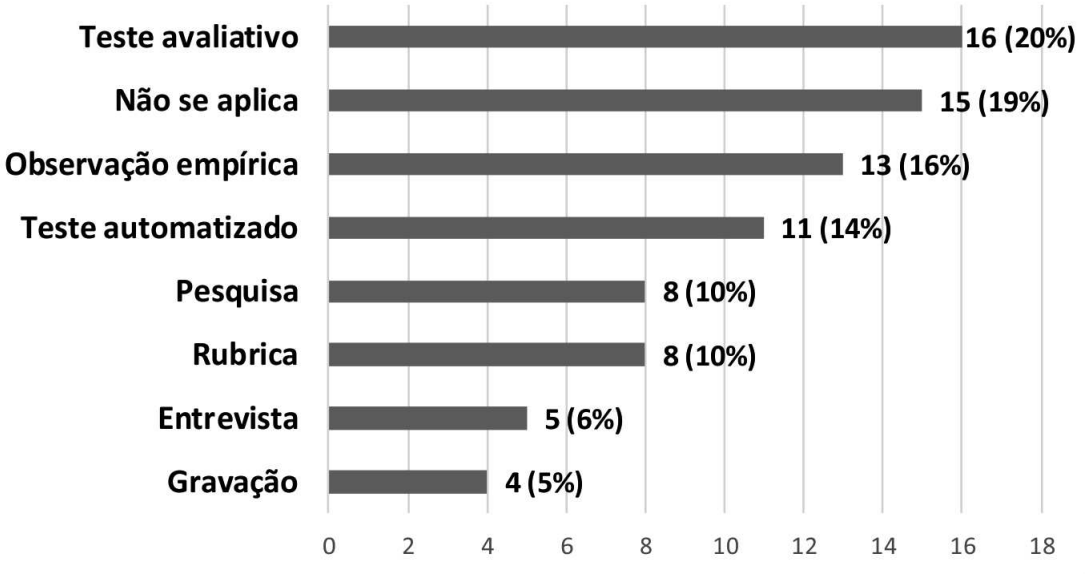
de portfólio, que corresponde a 20% das avaliações mapeadas. A avaliação por "Observação empírica" contém o conjunto de trabalhos que procuraram avaliar os alunos por meio de monitoramento de atividades e esteve presente em 16% dos tipos de avaliações. Logo em seguida com 14% , a categoria "Teste automatizado" agrupa os trabalhos que fizeram uso de ferramentas digitais para medir o nível de PC e outras habilidades dos alunos, como por exemplo, o grau de engajamento nas atividades. As demais categorias propostas foram: "Pesquisa", "Rubrica", "Entrevista" e "Gravação", que aparecerem em 10%, 10%, 6% e 5% das avaliações, respectivamente. A categoria "Não se aplica" apresenta os trabalhos que não fizeram nenhum tipo de avaliação do conhecimento e corresponde a 19% dos trabalhos, o que demonstra que um número muito grande de trabalhos da área não fazem avaliação dos resultados. É preciso deixar claro que um mesmo trabalho pode ter utilizado diferentes formas de avaliação, por isso alguns estudos estão distribuídos em mais de uma categoria.

Para um melhor entendimento de como se deu a categorização dos trabalhos, o Quadro 2 apresenta uma visão geral das subcategorias de avaliações constantes em cada uma das categorias descritas.

QP4- Em quais países os estudos se concentram?

Neste MSL, um total de 19 países apresentam publicações sobre o uso de PB no ensino do PC. Como demonstrado na Figura 57, o país com maior concentração de estudos sobre o assunto é os Estados Unidos, com 20 documentos selecionados para esse MSL, o que

Figura 56 – Tipos de avaliações empregadas



Fonte: elaborada pelo autor (2025).

contabiliza 34,5%, seguido da Índia com 8 (13,8%) e logo após a Coreia do Sul com 6 trabalhos selecionados (10,3%). Apenas 3,4% dos trabalhos foram realizados no Brasil. A categoria "Outros" corresponde a um total de 10 países que tiveram 1 artigo cada. O gráfico completo com todos os países pode ser encontrado em http://bit.ly/pc_pb.

É válido salientar que, 3 artigos foram publicados descrevendo dois países ao mesmo tempo. Esses artigos foram contabilizados como sendo um para cada país, dessa forma o somatório total foi 58 e não 55 trabalhos.

B.7 Trabalhos Relacionados

Existem alguns trabalhos que fizeram uso de um MSL a fim de levantar informações sobre PC. (ALVES *et al.*, 2019) realizou um MSL para levantar quais são as avaliações usadas para avaliar programas de computadores construídos através de PB, no contexto do ensino fundamental e médio. Já (SANTOS *et al.*, 2018) realizou um MSL (tendo como foco o cenário brasileiro) a fim de levantar informações sobre a diversidade de experiências que tratam de pensamento e programação computacional na educação básica. (MARTINS-PACHECO *et al.*, 2019) fez uso de um MSL para levantar informações sobre o que está sendo feito para avaliar o PC no contexto da educação básica.

Este MSL se diferencia dos trabalhos citados pois procura fazer um levantamento de estudos que utilizam PB, juntamente com o ensino do PC no cenário mundial e em todas as fases

Quadro 2 – Categorias e subcategorias dos tipos de avaliação

CATEGORIAS	SUBCATEGORIAS	QUANTIDADE	%
Teste avaliativo	Teste escrito	10	20%
	Teste prático	2	
	Análise de portifólio	1	
	Programas pontuados manualmente	1	
	Testes avaliativos gerais	2	
Não se aplica	-	15	19%
Observaçõe empírica	-	13	16%
Teste automatizado	Ferramenta Digital de Análise de Códigos em Blocos	5	14%
	Rastreamento Ocular	2	
	Captura de árvores de sintaxe abstratas (ASTs)	1	
	Análise de dados de log	3	
Rubrica	-	8	10%
Pesquisa	Formulário de Pesquisa	3	10%
	Pesquisa de Auto-Avaliação	2	
	Bilhetes de saída	1	
	Pesquisa de atitude	2	
Entrevista	Entrevistas Informais	3	6%
	Entrevista Semiestruturada	1	
	Entrevista baseada em artefatos	1	
Gravação	Gravações em sala de aula	1	5%
	Gravação de audio dos alunos	1	
	Gravações de tela de computador	2	
TOTAL:	**-**	**80**	**100%**

Fonte: elaborado pelo autor (2025).

do ensino formal.

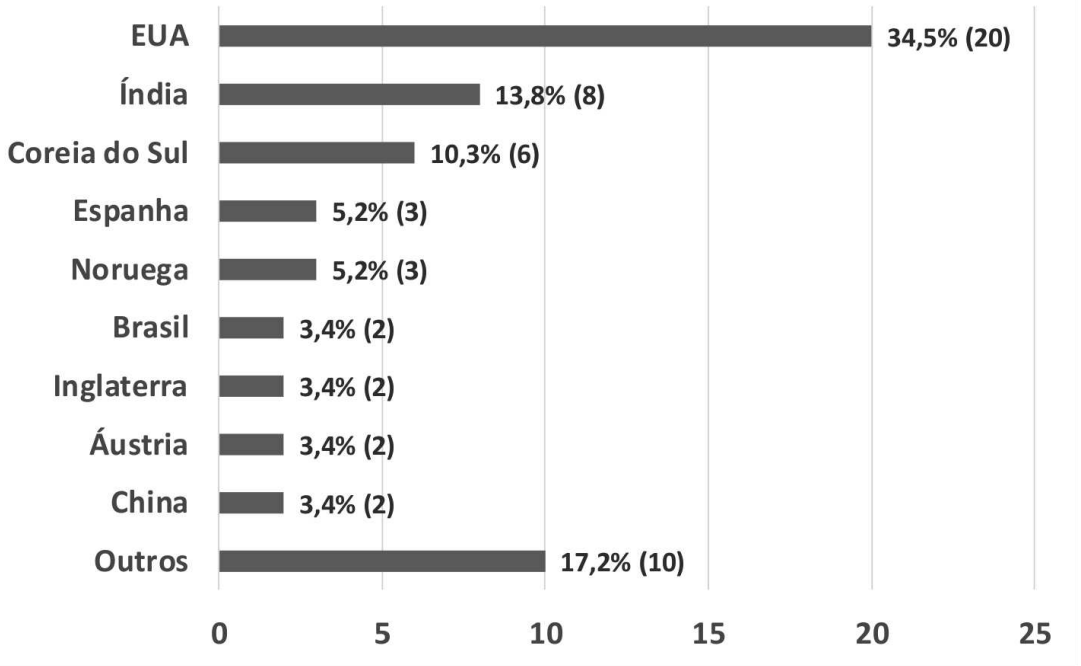
B.8 Considerações Finais

Esta seção apresentará as conclusões extraídas no decorrer deste trabalho, destacando os principais resultados obtidos e propondo alguns trabalhos futuros.

Pode-se perceber que está havendo um crescimento nas publicações a respeito do uso de PB no ensino do PC. Tal tendência é um forte indício do crescimento da área estudada.

Outro resultado interessante, é o grande número de ferramentas utilizadas, e apesar

Figura 57 – Países onde os estudos se concentram



Fonte: elaborada pelo autor (2025).

de o *Scratch* ter aparecido em um número considerável de estudos (30%), a grande quantidade de ferramentas existentes corrobora com o indicativo de que a área esteja em pleno crescimento. O fato de que quase 60% das ferramentas utilizadas serem *web* demonstra que a área segue uma tendência global, que é a substituição de ferramentas *desktop* por ferramentas *web*.

No mapeamento dos tipos de avaliação utilizados, pôde-se perceber que são utilizadas várias formas de avaliação diferentes com uma distribuição quase uniforme, como um leve protagonismo para os testes avaliativos. Contudo, o que chama atenção é que quase 20% dos artigos não fizeram uso, ou pelo menos não informaram, de nenhuma ferramenta de avaliação, o que pode indicar a necessidade de um maior rigor científico para os estudos na área.

Por fim, é válido salientar que a grande maioria dos estudos que se apresentaram para este MSL foram realizados nos Estados Unidos e que apenas 3,4% deles foram feitos no Brasil, indicando que ainda é uma área com forte potencial de estudo em nosso país. É claro que esse resultado deve ser analisado conforme as restrições do MSL, uma vez que apenas estudos escritos na língua inglesa foram selecionado e esse fator pode ter contribuído para justificar essa quantidade.

A pesquisa teve como objetivo geral fazer um levantamento do uso de PB, juntamente com o desenvolvimento do PC. Constatou-se que o objetivo geral foi atendido com sucesso.

Embora este trabalho procure seguir um protocolo pré-estabelecido a fim de fornecer

resultados confiáveis, algumas ameaças a validade podem ser apresentadas. Entre elas, pode-se destacar o fato de que o MSL é feito com base na análise dos pesquisadores, e pode, de alguma forma, introduzir análises enviesadas, mesmo que involuntárias. A *string* de busca também é um fator limitante, pois alguns trabalhos sobre o tema podem não ter sido recuperados e com isso não foram ser selecionados para este MLS.

Como trabalhos futuros pretende-se explorar melhor, realizando uma revisão sistemática de literatura, a utilização de plataformas web que buscam desenvolver o PC usando PB. Visto que foi observado uma grande adesão por tais sistemas.

Pretende-se também realizar um estudo com foco nos testes automatizados que medem o nível de PC. Uma vez que haja adesão do ensino de PC no contexto da escola básica é esperado que os professores nesse cenário não possuam habilidades suficiente para transmitir aos alunos. Testes automatizados podem ser bem úteis nesse processo inicial.

APÊNDICE C – PC NO CONTEXTO ESCOLAR BRASILEIRO: MSL

C.1 Análise da Adoção de Pensamento Computacional no Contexto Escolar Brasileiro: Um Mapeamento Sistemático da Literatura

C.2 Abstract e Resumo

C.2.1 Abstract

In the context of Brazilian education, Computational Thinking (CT) is currently in the phase of defining curriculum guidelines and finding ways to adapt its teaching to the country's diverse realities. This research aims to present a systematic literature mapping of studies that report the adoption of Computational Thinking in the school context, published in the SBC database between August 23, 2017, and August 23, 2022. We analyzed the main characteristics of the studies, such as location, educational levels, pedagogical theories, methodologies, and learning tools. This mapping selected 99 articles for analysis to address the research questions. As a result, it was possible to identify a prominent adoption of CT in schools, particularly in the Northeast region of Brazil. Additionally, a preference for research focused on elementary education was detected, as well as for visual programming tools and unplugged computational activities.

C.2.2 Resumo

No contexto da educação brasileira, o Pensamento Computacional (PC) encontra-se em fase da definição de referenciais curriculares e de formas de adequação do seu ensino às diferentes realidades do país. Esta pesquisa tem como objetivo apresentar um mapeamento sistemático da literatura sobre estudos que relatam a adoção do Pensamento Computacional no contexto escolar, publicados na base da SBC, entre os períodos de 23 de agosto de 2017 a 23 de agosto de 2022. Foram analisadas as principais características dos estudos como, localidade, níveis de escolaridade, teorias pedagógicas, metodologias e ferramentas de aprendizagem. Este mapeamento selecionou 99 artigos para análise, com o objetivo de responder às questões de pesquisa. Como resultado, foi possível identificar um destaque para a região Nordeste na adoção do PC em escolas, além disso detectou-se uma preferência das pesquisas para o Ensino Fundamental, assim como para as ferramentas de programação visual e computação desplugada.

C.3 Introdução

Estudos a respeito das habilidades que circundam o ensino de conceitos computacionais são bem anteriores à popularização do termo pensamento computacional (PC). O pesquisador *Seymour Papert* foi quem primeiro falou sobre o termo, ainda na década de 1980 (PAPERT, 1980). No entanto, após o trabalho de *Jeannette Wing* intitulado "*Computational Thinking*", esse termo se tornou popular. Em seu texto, a autora descreve um conjunto de atitudes e habilidades que ela definiu como Pensamento Computacional (WING, 2006a). Segundo Wing, o PC é um processo que envolve a formulação e resolução de problemas utilizando técnicas providas da Ciência da Computação (CC) e é considerado como fundamental não somente para cientistas, mas para todas as pessoas (RAABE *et al.*, 2017).

Com o propósito de pesquisar sobre a adoção do PC e da CC no currículo escolar, (HEINTZ *et al.*, 2016) analisam a formação dos alunos (ensinos fundamental e médio) de países como Estados Unidos, Inglaterra, Estônia, Finlândia, Noruega, Nova Zelândia, Suécia, Austrália, Coreia do Sul e Polônia. A pesquisa descobriu que cada um desses países encontra-se em um momento distinto da inclusão da CC em sua matriz curricular. Contudo, é perceptível que, de forma geral, existe uma tendência da aplicação do PC dentro dos processos de ensino e de aprendizagem.

Este trabalho, em formato de Mapeamento Sistemático da Literatura (MSL), objetiva avaliar e identificar estudos publicados em Anais e Periódicos da Sociedade Brasileira da Computação nos últimos cinco anos, que descrevam o andamento da adoção do PC no ambiente escolar brasileiro dos últimos anos. Está organizado da seguinte forma: a Seção C.4 apresenta a metodologia aplicada para o desenvolvimento deste MSL, a Seção C.5 é realizada a apresentação dos resultados e discussões, e por fim, na Seção C.6, são apresentadas as considerações finais.

C.4 Metodologia

Esse MSL foi produzido de acordo com as recomendações de PRISMA (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*) (MOHER *et al.*, 2010; GALVÃO *et al.*, 2015).

C.4.1 Questões de Pesquisa

Essa pesquisa foi norteada pela seguinte questão principal: *"Como o pensamento computacional está sendo aplicado nas escolas do Brasil ?"*.

A partir da pergunta norteadora, foram elaboradas outras três questões de pesquisa (QP) mais específicas:

- QP1: *Em que nível da educação formal estão sendo realizadas as pesquisas?*
- QP2: *Quais as teorias e metodologias pedagógicas são utilizadas?*
- QP3: *Quais as ferramentas são mais usadas no ensino do PC? Como essas ferramentas estão sendo utilizadas?*

C.4.2 Protocolo de Mapeamento

Como indicado em (MOHER *et al.*, 2010), um Protocolo de Mapeamento Sistemático foi desenvolvido, contendo todas as indicações de como deveriam ser selecionados os trabalhos para esta pesquisa.

C.4.2.1 Base de dados

Com o propósito de englobar o maior cenário possível de artigos científicos publicados pela Sociedade Brasileira de Computação (SBC), a busca foi realizada na SBC OpenLib (SOL), biblioteca digital que oferece acesso a todo o conteúdo produzido pela SBC (SBCOPENLIB, 2022).

C.4.2.2 Palavra-chave, intervalo de busca e idiomas considerados

O mecanismo de busca aplicado na biblioteca, contou como chave a palavra "Pensamento Computacional" utilizado em um campo nomeado 'Qualquer Lugar'. Dessa forma, retornaram resultados em que o termo estava presente em qualquer parte dos trabalhos. Foram incluídos apenas artigos produzidos nos últimos 5 anos, mais precisamente entre 23 de agosto 2017 e 23 de agosto de 2022, pois, de acordo com (MOHER *et al.*, 2010), este intervalo de tempo é adequado para análise de um MSL. Foram considerados apenas trabalhos que pertencessem às bases de Anais de Eventos ou Periódicos, podendo estar disponibilizados nos idiomas português ou inglês.

Tabela 8 – Anais de eventos e periódicos utilizados na pesquisa

Anais de Eventos e Periódicos	Qtd. de artigos
Workshop de Informática na Escola (WIE)	52
Simpósio Brasileiro de Educação em Computação (EDUCOMP)	10
Workshop sobre Educação em Computação (WEI)	9
Congresso sobre Tecnologias na Educação (CTRL+E)	7
Simpósio Brasileiro de Informática na Educação (SBIE)	6
Escola Regional de Computação Bahia, Alagoas e Sergipe (ERBASE)	4
LATINOWARE; RBIE; SBGAMES	2 cada
ENCOMPIF; ERCOMP-RS; SBESC; WIT; WCBIE	1 cada
Total	**99**

Fonte: elaborada pelo autor (2025).

C.4.2.3 Critérios de Inclusão e Exclusão

Foram considerados apenas trabalhos que atuaram em regiões brasileiras e no ambiente escolar. E desconsideradas pesquisas secundárias, como revisões da literatura e mapeamentos. Também foram descartados estudos primários que, mesmo tendo foco no desenvolvimento do PC, não identificaram o público alvo.

C.4.3 Condução da Pesquisa

Ao aplicar os filtros, o primeiro resultado da pesquisa retornou 248 artigos. Na sequência, ao realizar a leitura de títulos e resumos foram reduzidos a 186 artigos. Por fim, ao realizar a leitura dos capítulos de conclusões e/ou resultados foram selecionados 99 artigos. Nessa etapa, todos os estudos foram lidos na íntegra com o preenchimento de um fichamento, de acordo com as indicações do PRISMA (MOHER *et al.*, 2010; GALVÃO *et al.*, 2015). O protocolo de mapeamento sistemático completo, formulário de extração de dados e a relação de estudos com as informações mapeadas, estão disponíveis no link https://bit.ly/3yxL5o1.

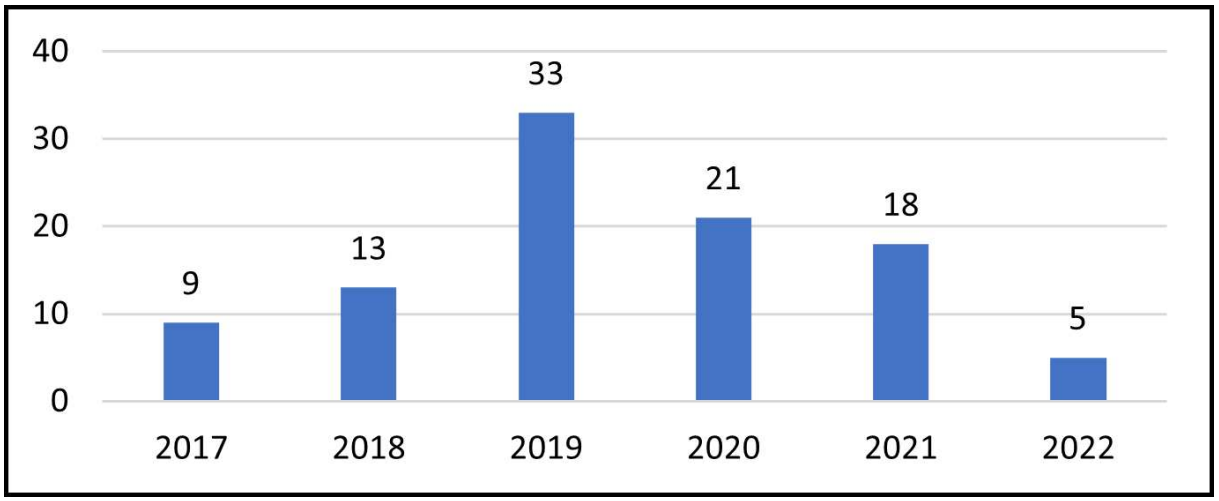
C.4.4 Trabalhos Selecionados

Na Tabela 8, são identificados os anais de eventos e periódicos dos artigos selecionados pela pesquisa, ordenados de forma decrescente pela quantidade de estudos. O evento com maior quantidade de estudos foi o Workshop de Informática na Escola (WIE), com mais de 50% do trabalhos selecionados.

A Figura 58 ilustra a distribuição dos estudos selecionados por ano de publicação. É possível perceber uma crescente evolução no interesse da adoção do PC em sala de aula, partindo

de 9 estudos em 2017 a 13 estudos em 2018 e chegando ao seu pico com 33 artigos em 2019. A diminuição da quantidade de trabalhos de 2020 e 2021 é esperada, uma vez que o impacto da pandemia COVID-19 adicionou diversas dificuldades para esse tipo de ação. Por fim, em 2022 foram selecionados apenas 5 artigos, como essa pesquisa foi realizada em meados de 2022, os principais anais de eventos e periódicos ocorrem nos meses finais do ano, a tendência é que em uma atualização desse estudo o número de estudos de 2022 seja maior.

Figura 58 – Distribuição de estudos por ano de publicação



Fonte: elaborada pelo autor (2025).

A Figura 59 apresenta a distribuição dos trabalhos por estado da federação. Ao todo, foi possível identificar 83 artigos, ficando 16 artigos sem identificação do estado, o que corresponde a um pouco mais de 83% da seleção total. Destaca-se o estado da Bahia com 16 estudos, correspondendo a mais de 16,16% do total. Como ilustrado, a região Nordeste encontra-se com a maior concentração dos estudos, seguido pela região Sul, Sudeste, Norte e Centro-Oeste, respectivamente.

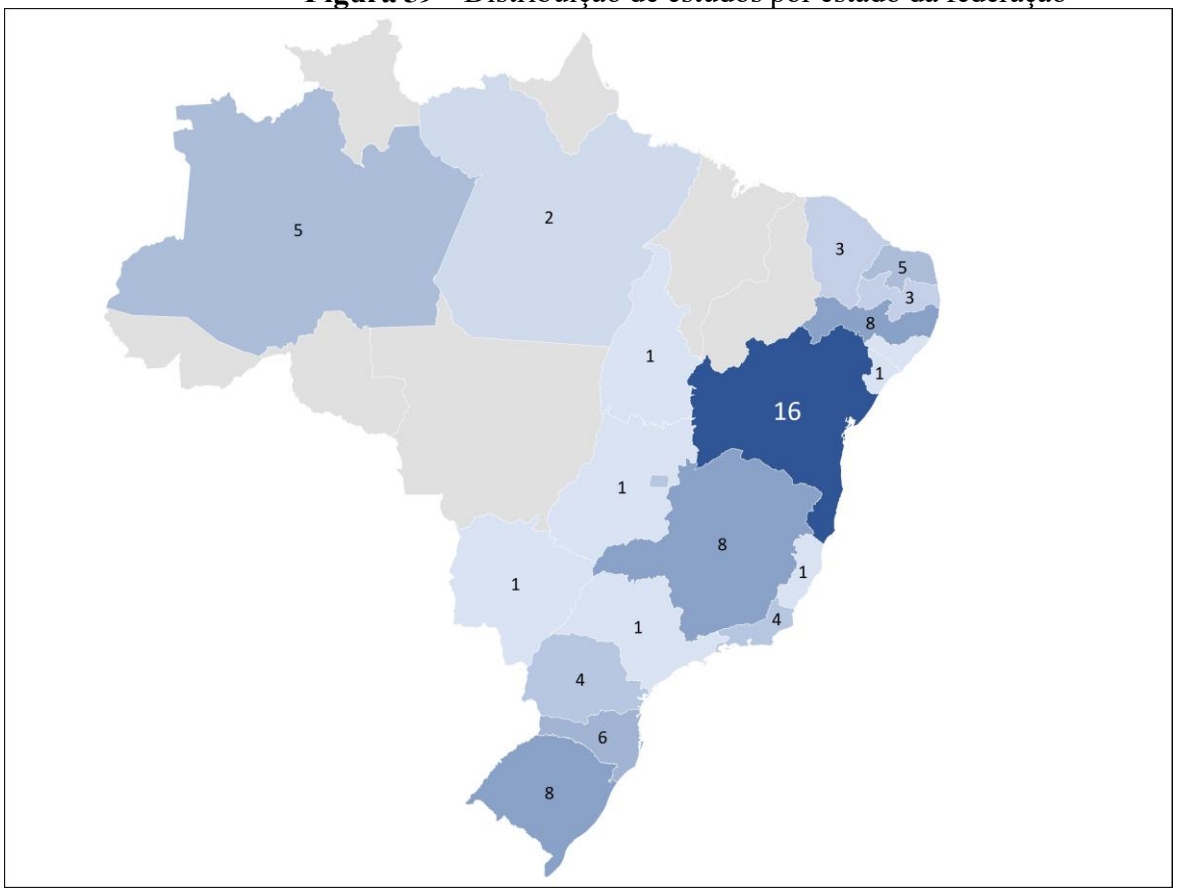
C.5 Resultados e Discussões

Nesta seção, são apresentados os resultados e discussões que auxiliem na obtenção de respostas válidas para cada uma das questões de pesquisa propostas por esse MSL.

C.5.1 Estágio da Educação Formal (QP1)

Essa questão buscou refletir sobre em que estágios da educação formal as pesquisas realizadas no Brasil estão sendo executadas. Para padronização e melhor entendimento, os níveis foram padronizados como Ensino Fundamental, Ensino Médio e Ensino Superior. Durante a

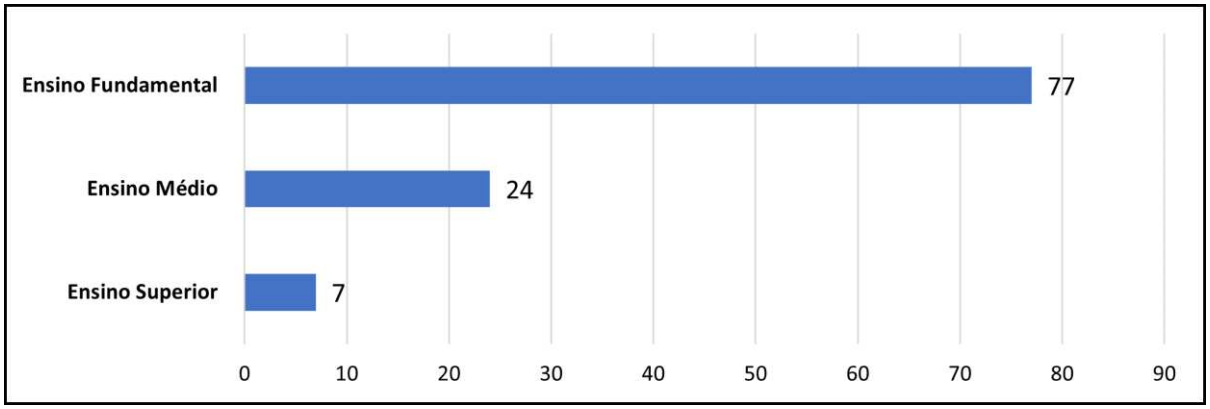
Figura 59 – Distribuição de estudos por estado da federação



Fonte: elaborada pelo autor (2025).

análise verificou-se que alguns trabalhos envolviam mais de um público, dessa forma, o mesmo estudo foi categorizado em mais de um nível.

Figura 60 – Distribuição de estudos por estágio da educação formal



Fonte: elaborada pelo autor (2025).

O ensino fundamental foi o nível educacional mais presente entre os artigos selecionados, sendo encontrado em 77 estudos, seguido pelo ensino médio com 24 e ensino superior com 7. Esses números ratificam os resultados apresentados no estudo de (HEINTZ *et al.*, 2016), que em sua revisão sobre o desenvolvimento de PC em vários países estrangeiros, também resulta

em uma clara tendência da inclusão do ensino do PC ainda na fase inicial da vida escolar dos estudantes.

Para os estudos aplicados no ensino fundamental, os pesquisadores optaram com maior frequência por ferramentas de computação desplugada, presente em mais de 53%, e programação visual, com participação em quase 52%. Uma possível explicação é o fato de que a computação desplugada é comumente desenvolvida utilizando atividades lúdicas, facilitando assim o envolvimento do público infantil.

Nas pesquisas realizadas com público do ensino médio, a maior ênfase está na programação visual. Encontrou-se essa abordagem em 62,5% dos trabalhos, seguidos por computação desplugada e robótica pedagógica ambos com 33,3% de participação. O fato de que os estudantes desse nível são mais maduros e entendem com maior facilidade os conceitos passados, são possíveis indicadores de uma frequência maior do uso da robótica pedagógica.

Como poucos trabalhos mapeados eram do ensino superior, existiu uma dificuldade em analisar mais profundamente a adoção nesse nível de ensino. Dos sete estudos encontrados, destacam-se a computação desplugada e a programação visual com participação em quatro estudos, linguagem de programação e robótica pedagógica com duas participações e jogos digitais com uma.

C.5.2 Teorias e Metodologias Pedagógicas (QP2)

C.5.2.1 Teorias Pedagógicas

As teorias pedagógicas que estavam vinculadas aos trabalhos selecionados foram: o **Construcionismo** de Seymon Papert (PAPERT, 1980), o **Construtivismo** de Jean Piaget e Jerome Bruner (PIAGET, 1960; BRUNER, 1971), a **Aprendizagem Significativa** de David Ausubel (AUSUBEL *et al.*, 1980) e a **Autonomia** de Paulo Freire (FREIRE, 2020). Os estudos foram classificados por definição direta ou de acordo com a semelhança que as intervenções educacionais. Na Figura 61, é ilustrada a distribuição das teorias pedagógicas dos estudos selecionados.

Quase metade dos estudos selecionados utilizaram a teoria do construcionismo. Essa teoria presa pelo protagonismo do aluno em construir seus próprios conhecimentos, o que vai de encontro com os ideias do PC, na medida de que o processo da resolução do problema pode ser encontrada de diversas formas, muitas vezes não obtendo somente uma forma de solução, o que

Figura 61 – Distribuição das teorias pedagógicas

Fonte: elaborada pelo autor (2025).

estimula o estudante a tomar suas próprias conclusões.

A segunda teoria mais encontrada foi a significativa, ela propõe que os seus conhecimentos anteriores devem ser aproveitados para o novo aprendizado, diversos estudos utilizavam de assuntos já trabalhados por disciplinas tradicionais, como matemática, linguagens naturais, ciências naturais e outros. Trabalhar o PC desta maneira é concordar que seu uso não é somente para a área da tecnologia. (WING, 2006a) afirma que futuros profissionais de diferentes áreas deverão interagir com soluções computacionais, através de um pensamento interdisciplinar.

Doze trabalhos utilizaram como teoria pedagógica o construtivismo. Essa teoria defende que as atividades devem despertar o interesse do aluno, fazendo com ele procure o conhecimento e o professor obtenha um papel de facilitador durante o aprendizado. Alguns estudos com intuito de trazer a atenção dos estudantes levantaram a proposta de projetar o desenvolvimento de jogos digitais desde *puzzles* e *quizzes*, à jogos famosos da internet como o conhecido Flappy Bird. Esse tipo de intervenção pedagógica apresenta-se como uma excelente opção, principalmente quando o objetivo é o ensino de programação de softwares.

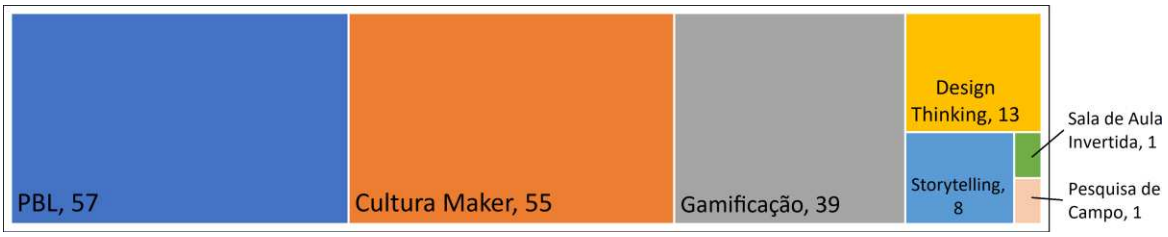
A teoria da autonomia esteve presente em sete dos trabalhos selecionados, ela prega que o aluno deve possuir liberdade durante o processo de aprendizagem, de modo que ele possa tomar decisões conscientes. Desta maneira, alguns trabalhos propuseram, após uma introdução das ferramentas de aprendizagem, que os estudantes construíssem projetos independentes, com somente apoio e sem interferência do professor, instigando a curiosidade e autonomia do aluno.

C.5.2.2 Metodologias de Aprendizagem

O ensino do PC em sala de aula exige do estudante um engajamento na descoberta de soluções para problemas do cotidiano, desta forma, o professor detém o desafio de propiciar experiências que mantenham o interesse do aluno na descoberta de soluções. Para esse fim, as metodologias ativas são ótimas alternativas, uma vez que se baseia em atividades que favorecem

o protagonismo na construção do próprio conhecimento. Na Figura 62, é ilustrada a frequência dos tipos de metodologias nos estudos selecionados, o somatório das metodologias ultrapassa a quantidade de artigos por mais de uma metodologia pode ter sido usada no mesmo trabalho.

Figura 62 – Frequência dos tipos metodologias



Fonte: elaborada pelo autor (2025).

Nenhum dos artigos selecionados apresentou uma metodologia passiva, que envolvem métodos tradicionais em sala de aula, onde o professor é a fonte do conhecimento e os alunos são somente receptores do conhecimento. Entre os tipos de metodologias ativas, o *Prolem Based Learning* (PBL) e a cultura *maker* foram as predominantes, com 57% e 55% de participação, respectivamente.

O aprendizado por problema foi abordado através de desafios, em que o estudante deveria aplicar métodos computacionais propostos pelo professor, esta metodologia tem uma ação direta no uso de PC, visto que geralmente era descrito de forma clara como a atividade iria desenvolver a habilidade aplicada.

A cultura maker foi frequentemente encontrada nos estudos em que o pesquisador colocou o aluno na função de desenvolvedor da atividade, e que o resultado de suas realizações era o reflexo do desempenho do estudante. Este tipo de metodologia estimula a curiosidade do aluno em resolver seu problema, dando autonomia de buscar sua resolução de acordo com que as habilidades do PC eram aprofundadas no curso.

A gamificação é uma metodologia que busca a união do mundo do jovem aprendiz com a educação, visto que traz consigo elementos comuns aos jogos digitais, como desafios, regras e narrativas em geral, com o intuito de exercitar o pensamento analítico. Nos estudos, essa prática apareceu com uma opção para aumentar o engajamento do alunado nas práticas de PC, uma vez que as habilidades a serem desenvolvidas eram feitas em formato de jogos ou atividades gamificadas, fazendo com que o aluno conseguisse se manter motivado durante o ensino.

O *Design Thinking* é uma metodologia que concentra-se na solução de problemas criativos, começando com uma compreensão profunda das necessidades e envolvendo-se no desenvolvimento de soluções inovadoras e atendendo as necessidades. O envolvimento dessa

metodologia nos estudos aconteceu frequentemente em pesquisas que objetivava a construção de projetos, onde era apresentado um problema que pudesse ser resolvido com as habilidades do PC desenvolvidas.

O *Storytelling* é uma metodologia que utiliza-se da narração de histórias para o ensino, a qual trabalha a criatividade, imaginação e memória, além de outras características. Esta prática mostrou-se bem aceita nos estudantes do ensino fundamental, ampliando o interesse com histórias e estreitando a comunicação com os alunos.

As metodologias ativas de sala de aula invertida e pesquisa de campo foram encontradas em apenas um artigo cada. A pesquisa de (LIMA *et al.*, 2019) justifica o uso da sala de aula de invertida com a otimização do tempo, fazendo com que o conteúdo adquirido anteriormente facilite o aprendizado nas atividades propostas em sala. A presença da pesquisa de campo foi detectada de forma subjetiva em (GUSMÃO; FRANÇA, 2019), onde propôs em um ambiente externo uma apresentação e aperfeiçoamento dos projetos dos alunos da pesquisa.

C.5.3 Ferramentas de Aprendizagem (QP3)

As ferramentas de aprendizagem no ensino do pensamento computacional definem como serão desenvolvidos os pilares do PC, para isso um fator importante é que o aluno apresente-se apto para entender o funcionamento da ferramenta, e desenvolva suas habilidades. No Gráfico 63, é ilustrada a frequência das ferramentas aplicadas nos estudos selecionados.

Figura 63 – Frequência das ferramentas de aprendizagem

Fonte: elaborada pelo autor (2025).

Com aproximadamente 53% de participação, linguagem de programação visual e computação desplugada compartilharam a primeira posição entre as ferramentas, seguidas de jogos digitais com quase 27%, robótica pedagógica com 23%, linguagem de programação 13% e animação com somente um estudo abordando essa estratégia.

A linguagem de programação visual é uma ferramenta que usa gráficos para representar conceitos de programação, incluindo blocos de construção de programas, gráficos de fluxo

e outras representações gráficas. Apresenta-se como uma opção interessante para o público infantil, pois seu uso é descrito em muitos trabalhos como intuitivo e fácil de usar, o que torna adequada para trabalhar com iniciantes. As plataformas de programação visual encontradas com maior frequência nos estudos selecionados foram:

- Code.Org (CODE.ORG, 2022);
- Scratch (SCRATCH, 2022);
- MIT App Inventor (MITAPPINVENTOR, 2022).

A computação desplugada por sua vez é uma abordagem que corresponde a atividades que envolvem os conceitos de ciência da computação sem o auxílio do computador. Sua participação esteve presente em mais da metade das pesquisas selecionadas. Isso pode ser explicado pela dificuldade de infraestrutura presente no Brasil, segundo o Anuário de Educação de 2021 realizado pelo (MEC, 2021), no ensino fundamental apenas 40,3% das escolas possuem laboratório de informática, e isso se agrava ainda mais se observamos que apenas 35,6% possuem acesso à internet. Considerando a união dos dois ensinos, fundamental e médio, o percentual de estabelecimentos com laboratório de informática resulta em 47,08%, o que equivale a menos da metade do total. Dessa forma, a computação desplugada é uma ferramenta de grande valor para o desenvolvimento do pensamento computacional no Brasil. Atualmente, diversas atividades desplugadas são disponibilizadas na internet, como exemplo pode-se citar o site *CS Unplugged* (UNPLUGGED, 2022), que fornece vários materiais gratuitos, aulas prontas, vídeos demonstrativos e material de apoio para vários idiomas.

Os jogos digitais ajudam os alunos a aprenderem conteúdos de uma forma mais dinâmica e divertida. Os estudos que abordaram essa estratégia buscaram promover um ambiente mais estimulante e dinâmico, onde as historias do jogo trabalhassem os pilares do pensamento computacional. Diferentes tipos de jogos foram apresentados pelas pesquisas, desde jogos mais robustos como o Minecraft trabalhado por (TRINDADE *et al.*, 2020), a jogos bem didáticos como o Gramaticando aplicado por (SARMENTO *et al.*, 2017) e Furbot aplicado por (KOHLER *et al.*, 2021). Alguns estudos partiram da ideia da própria turma desenvolver seus jogos, como em (FARIAS *et al.*, 2019) que os alunos desenvolveram o projeto Monte seu Evento. Para a utilização direta de conceitos de programação os trabalhos de (SOUZA *et al.*, 2018) e (FREITAS; MORAIS, 2019) aplicaram o software Lightbot, que é um jogo educacional que utiliza-se de comandos que fazem referência a algoritmos computacionais, como estruturas sequências, repetição, e funções, a partir de como o aluno vai avançando as fases os desafios vão ficando

mais difíceis.

O objetivo principal da robótica pedagógica é que os estudantes programarem um mecanismo para executar tarefas. Dessa forma, partindo de instruções simples como programar um robô para seguir uma linha, girar em torno de um objeto ou simplesmente fazer o que for pedido. Muitos estudos indicaram uma dificuldade durante o uso desta ferramenta, uma vez que é necessário de equipamentos para o ensino, para isso (SILVA *et al.*, 2019a) propõe o uso do Arduíno como opção para minimizar essa problemática, pois se trata de uma plataforma de baixo custo para prototipação eletrônica de hardware, junto com a plataforma S4A que é uma plataforma oriunda do Scratch, que simplifica a programação do equipamento através da programação em blocos.

As linguagem de programação exigem uma maior formalidade sintática, pois é necessário especificar uma sequência de etapas a serem executadas por um computador. Nos estudos é possível perceber uma predominância da escolha da linguagem Python para a introdução do estudante, (ALEXANDRINO *et al.*, 2021) faz a escolha desta linguagem "[...] por ter uma sintaxe simples e de fácil aprendizado, se comparada com outras linguagens", em (SILVA *et al.*, 2019b) é enfatizado que sua escolha se baseia na legibilidade e produtividade, obtendo uma rápida e fácil compreensão dos programadores iniciantes.

A ferramenta de animação foi apresentada por (TAVARES *et al.*, 2021), o estudo parte do impasse causado pela pandemia Sars-CoV-2 (COVID-19), que foi a dificuldade de aplicar atividades desplugadas presenciais. A pesquisa desenvolveu animações computacionais que foram transpostas de desafios desplugados, a qual resultou em que 100% dos estudantes assimilassem o uso de algoritmos para soluções de problemas, conjuntamente 82% dos alunos demonstraram ter assimilado os conceitos de pensamento computacional.

C.6 Considerações Finais

O presente estudo teve como objetivo analisar a adoção do pensamento computacional nas escolas brasileiras descrita nas publicações acadêmicas dos últimos cinco anos. Para tal, foi realizado um mapeamento sistemático da literatura envolvendo os principais anais de eventos e periódicos associados à SBC. Os resultados indicam que todas as regiões do Brasil buscam o desenvolvimento do PC em suas escolas, com um destaque para a região Nordeste nas publicações encontradas. Além disso, no artigo, foram listadas as ferramentas que cada região mais utiliza, inferindo em âmbito nacional duas ferramentas predominantes no ensino de PC nas

escolas.

De acordo com a análise da QP1, é possível observar uma tendência de pesquisas no início da fase estudantil, constatando que atualmente o Brasil caminha de acordo com países que possuem maior nível de maturidade no ensino do PC. Os resultados da QP2, por sua vez, mostram que a maioria dos estudos se baseiam no construcionismo, onde o estudante é estimulado a construir seu próprio conhecimento, e na aprendizagem significativa, em que se reutiliza-se saberes adquiridos anteriormente para construir um novo aprendizado. Analisando as metodologias de aprendizagem utilizadas durante o ensino, foi nos revelado a não utilização de metodologias passivas, e destacada a adoção das metodologias ativas PBL e da cultura maker. Na análise apresentada para responder a QP3, descobriu-se um destaque no uso de ferramentas de programação visual e computação desplugada em estudos aplicados em escolas brasileiras.

A partir dos resultados deste mapeamento, um primeiro desdobramento seria ampliar e aprofundar a análise com foco na avaliação dos efeitos do ensino de PC no desempenho dos estudantes nos diversos níveis de ensino. Espera-se também que pesquisadores possam ampliar as metodologias já adotadas para ensino de PC, identificando sua adequação às diversas realidades nacionais.

APÊNDICE D – ROTEIRO DE ENTREVISTA SEMIESTRUTURADA COM O PROFESSOR USUÁRIO DO AUTOEVAL-CT

Apresentação e Contextualização

- Agradecimento pela participação;
- Explicação breve sobre o objetivo da entrevista: compreender percepções, experiências, benefícios e desafios relacionados ao uso da ferramenta de avaliação do Pensamento Computacional;
- Garantia de sigilo e anonimato nas respostas.

1. Perfil do Professor

- Qual sua formação acadêmica?
- Há quanto tempo atua no ensino de programação ou disciplinas relacionadas ao Pensamento Computacional?
- Já havia utilizado algum tipo de ferramenta de avaliação automatizada? Qual(is)?

2. Percepções sobre Eficiência e Usabilidade

- Comparando com métodos tradicionais, a ferramenta contribuiu para otimizar o tempo dedicado à avaliação?
- Houve redução no tempo necessário para fornecer feedback aos alunos?
- Considera que a ferramenta poderia ser utilizada regularmente em sua prática docente? Por quê?

3. Qualidade da Avaliação e do Feedback Gerado

- Os feedbacks gerados foram coerentes com sua própria análise enquanto avaliador?
- A ferramenta conseguiu identificar corretamente aspectos das quatro dimensões do Pensamento Computacional (Abstração, Decomposição, Reconhecimento de Padrões e Algoritmos)?
- Houve casos em que o resultado da ferramenta discordou significativamente da sua percepção? Poderia citar exemplos?

4. Impactos no Processo de Ensino e Aprendizagem

- De que forma o uso da ferramenta impactou sua prática pedagógica?
- Percebeu mudanças na compreensão ou no desempenho dos alunos após a utilização da ferramenta?
- Os alunos receberam bem os feedbacks automáticos? Houve discussões ou reflexões decorrentes deles?

5. Barreiras e Desafios

- Quais foram as principais dificuldades encontradas no uso da ferramenta (técnicas, pedagógicas ou institucionais)?
- Quais aspectos considera passíveis de melhoria na ferramenta?
- A infraestrutura tecnológica da escola foi adequada para o uso da ferramenta?

6. Possibilidades Futuras e Sugestões

- Vê potencial na expansão desse tipo de ferramenta para outros contextos, disciplinas ou níveis de ensino?
- Que melhorias, recursos ou funcionalidades adicionais sugeriria para futuras versões da ferramenta?
- Recomendaria o uso do AutoEval-CT a outros professores? Por quê?

7. Considerações Finais

- Deseja acrescentar algum ponto que não foi abordado e que considere relevante sobre sua experiência com a ferramenta?

APÊNDICE E – ARTIGOS SELECIONADOS NA FASE DE APROFUNDAMENTO

Lista dos trabalhos selecionados no MSL referentes à Fase de Aprofundamento

1. Promoting Computational Thinking Skills in Non-Computer Science Students Gamifying Computational Notebooks to Increase Student Engagement (SANTO *et al.*, 2022);
2. Game-Based Learning for Young Children: A Case Study (POMBO; LAMAS, 2022);
3. Evaluating Usability and Educational Effectiveness of a Serious Game for Programmers Using Alternative Interfaces and Types of Activities (ZARKADOULA; XENOS, 2022);
4. How Immersion and Self-Avatars in VR Affect Learning Programming and Computational Thinking in Middle School Education (PARMAR *et al.*, 2022);
5. The Effect on Computational Thinking Using SRA-Programming: Anticipating Changes in a Dynamic Problem Environment (FANCHAMPS *et al.*, 2022);
6. The effects of constructing robot-based storytelling system on college students' computational thinking skill and technology comprehension (HU *et al.*, 2022);
7. DWES: A Dynamic Weighted Evaluation System for Scratch based on Computational Thinking (CHAI *et al.*, 2021);
8. Development and Evaluation of Quizzes Aimed at Quantifying Computational Thinking (TANIOKA; YANO, 2021);
9. Research on Teaching Effectiveness of Computational Thinking Based on Service Learning (CHEN *et al.*, 2021);
10. Implementing Blended Learning in K-12 Programming Course: Lesson Design and Student Feedback (ZHANG; CUI, 2021);
11. Investigating the Impact of Computing vs Pedagogy Experience in Novices Creation of Computing-Infused Curricula (ISVIK *et al.*, 2021);
12. Computational Thinking Test for Lower Primary Students: Design Principles, Content Validation, and Pilot Testing (ZHANG *et al.*, 2021);
13. How do Bebras Tasks Explore Algorithmic Thinking Skill in a Computational Thinking Contest? (OLIVEIRA *et al.*, 2021);
14. Mutual Improvement between Teaching Materials and Assessment Tools for K-12 Programming Education (NAGUMO *et al.*, 2021);
15. Development and Implementation of an Online Adaptive Gamification Platform for Learning Computational Thinking (NG *et al.*, 2021);
16. Making Research Practice Partnerships Work: An Assessment of The Maker Partnership

(FANCSALI *et al.*, 2021);

17. A Game Development to Promote Computational Thinking (PROMPOLMAUENG *et al.*, 2021);
18. Improving Students' Mathematical Computational Thinking Using Scratch Program th-rough Project Based Learning: A Development Research during Pandemic Covid-19 (HADI *et al.*, 2021)
19. Active Learning Strategies: A Computing Course for Undergraduates (DUAN *et al.*, 2021)
20. Diversified Teaching Evaluation of Python Programming Based on OBE Concept (GOU *et al.*, 2021)
21. Collaborative Game-Based Environment and Assessment Tool for Learning Computational Thinking in Primary School: A Case Study (ZAPATA-CÁCERES *et al.*, 2021)
22. Social Robot to teach coding in primary school (LANZILOTTI *et al.*, 2021)
23. Exploring The Psychometric Properties of Computational Thinking Assessment in Intro-ductory Programming (ANISTYASARI *et al.*, 2021)
24. TechCheck-K: A Measure of Computational Thinking for Kindergarten Children (REL-KIN; BERS, 2021)
25. Evaluating a Computational Thinking and Computing Attitudes Instrument for Educational Purposes (OLIVEIRA *et al.*, 2021)
26. Designing "Bebras" serious games interaction for indonesian upper elementary school students (ASLINA *et al.*, 2020)
27. Designing Interaction and User Interface of Computational Thinking Digital Game for Children using User-Centered Design Approach (RAMADHANI *et al.*, 2020)
28. Low-Cost Educational Robotics Car Promotes STEM Learning and 21st Century Skills (ONG; LING, 2020)
29. Exploring the Relationship Between Collaborative Discourse, Programming Actions, and Cybersecurity and Computational Thinking Knowledge (YETT *et al.*, 2020)
30. Stem education activities development to promote computational thinking's students (SRISANGNGAM; DECHSURA, 2020)
31. The Research of Programming Teaching in Primary School on the Cultivation of Compu-tational Thinking (WANG *et al.*, 2020)
32. Computational Thinking Growth During a First-Year Engineering Course (DIAZ *et al.*, 2020)

33. An Effective Approach to Teach an Introductory Computer Science Course with Computational Thinking and Flow-Chart Based Visual Programming (RAHMAN *et al.*, 2020)
34. Developing Computational Thinking and Reading and Writing Skills through an Approach for Creating Games (FERNANDES *et al.*, 2020)
35. Teaching CT through Internet of Things in High School: Possibilities and Reflections (SCHNEIDER *et al.*, 2020)
36. Comparing computational thinking skills of engineering students in urban and rural areas of Peru (NUNEZ *et al.*, 2020)
37. Comprehension analysis considering programming thinking ability using code puzzle (ITO *et al.*, 2020)
38. When Positive Perception of the Robot Has No Effect on Learning (NASIR *et al.*, 2020)
39. Automatic Formative Assessment in Computer Science: Guidance to Model-Driven Design (MARCHISIO *et al.*, 2020)
40. The Teaching Innovation and Research of Python Programming in Financial and Economic Universities (FANG, 2020)
41. Teaching reform of programming basic course based on SPOC blended teaching method (LI; GU, 2020)
42. Comparing the effects of robots and IoT objects on STEM learning outcomes and computational thinking skills between programming-experienced learners and programming-novice learners (HU *et al.*, 2020)
43. An Automatic Analysis Tool Based on Computational Thinking for BlockPy Programs (XU *et al.*, 2020)
44. Usability evaluation of block programming tools in IoT contexts for initial engineering courses (SOBREIRA *et al.*, 2020)
45. Pic2Program - an Educational Android Application Teaching Computational Thinking (UTESCH *et al.*, 2020)
46. Computational Thinking Test for Beginners: Design and Content Validation (ZAPATA-CÁCERES *et al.*, 2020)
47. Emídio Garcia School Pilot description: A Robosteam Erasmus+ Project Activity based on a Challenge based Learning Approach (CAMARGO *et al.*, 2020)
48. Impact of Programming Exposure on the Development of Computational Thinking Capabilities: An Empirical Study (CACHERO *et al.*, 2020)

49. DeepStealth: Game-Based Learning Stealth Assessment With Deep Neural Networks (MIN *et al.*, 2019)
50. A Formative Assessment Tool to Support Computational Thinking in the Classroom (OROZCO-GARCIA *et al.*, 2019)
51. Computational Thinking Education: Who Let the Dog Out? (SWAID; SUID, 2019)
52. Computer Science Unplugged for Developing Computational Thinking and Mathematical Thinking (NAKAMURA; KAWASAKI, 2019)
53. Evaluation of the ROOT Robot System and Curriculum to Improve Computational Thinking in Chinese Children (LIU; ROJAS, 2019)
54. Designing hybrid physics labs: combining simulation and experiment for teaching computational thinking in first-year engineering (FENNELL *et al.*, 2019)
55. An investigation of undergraduates' computational thinking in a sophomore-level biomedical engineering course (SHOAIB *et al.*, 2019)
56. Algorithmic Expressions for Assessing Algorithmic Thinking Ability of Elementary School Children (OOMORI *et al.*, 2019)
57. Exploring the Effectiveness of Learning Scratch Programming with Code.org (YANG; LIN, 2019)
58. Pilot Experience: Play and Program with Bee-Bot to Foster Computational Thinking Learning in Young Children (CABALLERO-GONZÁLEZ *et al.*, 2019)
59. Use of Augmented Reality for Computational Thinking Stimulation through Virtual (ESTEVES *et al.*, 2019)
60. Introducing Computational Thinking Concept Learning in Building Cognitive Capacity and Character for Elementary Student (YULIANA *et al.*, 2019)
61. Assessment of Computational Thinking in regular basic education: case IETP "José Obrero" (CURASMA *et al.*, 2019)
62. A New Teaching Pattern Based on PBL and Visual Programming in Computational Thinking Course (GAO *et al.*, 2019)
63. Gamification and Engagement: Development of Computational Thinking and the Implications in Mathematical Learning (PIRES *et al.*, 2019)
64. Using topic modeling to extract pre-service teachers' understandings of computational thinking from their coding reflections (CUTUMISU; GUO, 2019)
65. A Feasibility Study of Arducation Bot: An Educational Robotics and Mobile Application

Kit for Computational Thinking Skills (PHETSRIKRAN *et al.*, 2018)

66. Learning to think like a trainer: bringing Scratch for Educational Sciences professional's formation (ALMEIDA *et al.*, 2018)
67. Effects of Professional Development on Programming Knowledge and Self-Efficacy (REIMER *et al.*, 2018)
68. Computational Thinking and Coding Subject in Primary Schools: Methodological Approach Based on Alternative Cooperative and Individual Learning Cycles (VLAHU-GJORGIEVSKA *et al.*, 2018)
69. Computational Thinking Education for Children: Algorithmic Thinking and Debugging (WONG; JIANG, 2018)
70. Exploring the Role of Visual Programming Activities in Computational Thinking (LIN *et al.*, 2018)
71. Developing Computational Thinking Skills in Adolescents With Autism Spectrum Disorder Through Digital Game Programming (MUNOZ *et al.*, 2018)
72. Alg-Design: Facilitates to Learn Algorithmic Thinking for Beginners (VINAYAKUMAR *et al.*, 2018)
73. Development and assessment of computational thinking: A methodological proposal and a support tool (PÉREZ; VALLADARES, 2018)
74. Towards playful learning and computational thinking â €" Developing the educational robot BRICKO (PEDERSEN *et al.*, 2018)
75. A multimodal LEGO®-based learning activity mixing musical notation and computer programming (LUDOVICO *et al.*, 2017);
76. Active Learning Environments with Robotic Tangibles: Children's Physical and Virtual Spatial Programming Experiences (BURLESON *et al.*, 2017)
77. iProg: Getting started with Programming: Pilot experiment in two elementary schools (ALMEIDA *et al.*, 2017)
78. Integrating computational thinking into english dialogue learning through graphical programming tool (WENG; WONG, 2017)
79. Exploring computational thinking assessment in introductory programming courses (ARAUJO *et al.*, 2017)
80. Teaching computational thinking to entry-level undergraduate engineering students at Amrita University (SHYAMALA *et al.*, 2017)

81. Automated Personalized Feedback in Introductory Java Programming MOOCs (MARIN *et al.*, 2017)

APÊNDICE F – HABILIDADES, EVIDÊNCIAS E INDÍCIOS

F.1 Habilidade 1: Abstração

- **Evidência 1**: A solução apresenta a contextualização do desafio.
  - **Indício 1.1**: Introdução clara e completa ao problema no código ou comentários.
  - **Indício 1.2**: Relação da solução com o contexto real ou cenários práticos.
- **Evidência 2**: Foram identificados os requisitos essenciais para a solução.
  - **Indício 2.1**: Mencionados requisitos essenciais no código ou comentários.
  - **Indício 2.2**: Implementação direta dos requisitos essenciais mencionados.
  - **Indício 2.3**: Priorização correta dos requisitos mais importantes.
- **Evidência 3**: Foi atingido o resultado esperado pelo desafio.
  - **Indício 3.1**: Saída correta para os casos de teste fornecidos.
  - **Indício 3.2**: Tratamento adequado de cenários de borda ou casos excepcionais.
  - **Indício 3.3**: Otimização da solução para desempenho e eficiência.

F.2 Habilidade 2: Decomposição

- **Evidência 4**: O desafio foi dividido em sub-desafios.
  - **Indício 4.1**: Identificação de funções ou métodos separados.
  - **Indício 4.2**: Comentários que descrevem a lógica de divisão.
  - **Indício 4.3**: Estrutura hierárquica do código (uso de classes, módulos).
- **Evidência 5**: Utiliza-se da resolução dos sub-desafios para compor a solução final.
  - **Indício 5.1**: Chamadas a funções/métodos em sequência lógica.
  - **Indício 5.2**: Integração de resultados dos sub-desafios.
  - **Indício 5.3**: Uso de estruturas de controle para coordenar sub-desafios (ex.: loops, condicionais).

F.3 Habilidade 3: Reconhecimento de Padrões

- **Evidência 6**: Os padrões do desafio foram identificados.
  - **Indício 6.1**: Reutilização de soluções pré-existentes.
  - **Indício 6.2**: Agrupamento de similaridades no código.
  - **Indício 6.3**: Uso de padrões de design (ex.: MVC, Singleton, Factory).

- **Evidência 7**: Houve reaproveitamento de soluções.
  - **Indício 7.1**: Reaproveitamento de blocos de código já desenvolvidos.
  - **Indício 7.2**: Reutilização de funções em múltiplas situações.
- **Evidência 8**: Reaproveitamento de instruções por meio de repetição e condição.
  - **Indício 8.1**: Uso de loops para ações repetitivas.
  - **Indício 8.2**: Uso de condicionais para evitar duplicação de lógica.

F4 Habilidade 4: Algoritmos

- **Evidência 9**: A solução fornece uma sequência de instruções para resolver o desafio.
  - **Indício 9.1**: Sequência lógica de instruções.
  - **Indício 9.2**: Cobertura de todas as etapas do problema.
- **Evidência 10**: As instruções estão detalhadas.
  - **Indício 10.1**: Detalhamento das instruções por meio de comentários.
  - **Indício 10.2**: Instruções claras e detalhadas no código.
- **Evidência 11**: A solução está completa.
  - **Indício 11.1**: Código resolve todos os requisitos do desafio.
  - **Indício 11.2**: Tratamento de todas as possíveis entradas e cenários previstos.
- **Evidência 12**: Reconhece a diferença entre as estruturas de condição e repetição.
  - **Indício 12.1**: Uso correto de condicionais e loops.
  - **Indício 12.2**: Diferenciação clara entre uso de condicionais e loops.
- **Evidência 13**: Controla as repetições e condições.
  - **Indício 13.1**: Controle adequado das repetições com condições de saída.
  - **Indício 13.2**: Uso eficiente de condicionais para controlar o fluxo.

APÊNDICE G – TRECHOS DO CÓDIGO-FONTE UTILIZADO

```python
import json
from typing import List
from fastapi import HTTPException, Form

from src.core import (
    Engine,
    OpenaiModel,
    OllamaModel,
    GeminiModel,
    DeepSeekModel
)

from ..utils import save_md_file, get_absolute_path
from ...dimensions import dimensions as base_dimensions
from src.api.models.challenge_models import Dimension

def parse_dimensions(dimensions: str = Form(...)) -> List[Dimension]:
    try:
        dimensions_list = json.loads(dimensions)
        return [Dimension(**dimension) for dimension in
            dimensions_list]
    except json.JSONDecodeError:
        raise HTTPException(status_code=400, detail="Invalid JSON
            format in dimensions")
    except ValueError as e:
        raise HTTPException(status_code=422, detail=str(e))

def get_llm_model(llm: str):
    llm = llm.lower()
    models = {
        "gpt": OpenaiModel,
        "ollama": OllamaModel,
        "gemini": GeminiModel,
        "deepseek":  DeepSeekModel,
    }
    if llm not in models:
```

```
35          raise HTTPException(status_code=400, detail=f"LLM '{llm}' nao
                e suportado.")
36      return models[llm]()
37
38  def process_challenge_submission(
39      general_context: str,
40      code_text: str,
41      dimensions: List[Dimension],
42      ai_model,
43  ) -> List[dict]:
44      """
45      Processa a submissao do desafio:
46        1. Gera a documentacao.
47        2. Salva o documento.
48        3. Alimenta a engine com codigo e documentacao.
49        4. Para cada dimensao, evidencia e indicio, executa a engine.
50      """
51      engine = Engine(ai_model)
52
53      documentation = engine.make_documentation(general_context)
54      docs_path = get_absolute_path("./../../../md/docs/")
55      save_md_file(content=documentation, path=docs_path)
56
57      engine.inputs(code=code_text, documentation=documentation)
58
59      response = []
60      for dimension in dimensions:
61          dimension_config = base_dimensions[str(dimension.key)]
62          engine.set_dimension(dimension_config)
63
64          dimension_item = engine.get_dimension()
65          dimension_item['evidences'] = []
66
67          for evidence in dimension.evidences:
68              engine.set_evidence(evidence.key)
69              evidence_item = engine.get_evidence(evidence.key)
70              evidence_item['clues'] = []
71
72              for clue in evidence.clues:
```

```
                    engine.set_clue(clue.key)
                    output = engine.output()
                    clue_item = engine.get_clue(clue.key)
                    clue_item['output'] = output
                    evidence_item['clues'].append(clue_item)

                dimension_item['evidences'].append(evidence_item)

            response.append(dimension_item)

    return response
```

Código-fonte 4 – Service

```
from .base_model import BaseModel
from ..clients.open_ai import OpenaiClient

class OpenaiModel(BaseModel):
    def __init__(self):
        super().__init__()
        self.client = OpenaiClient

    def chat(self):
        self.completion = self.client.chat.completions.create(
            model="gpt-4o",
            messages=self.messages,
        )
        output = self.completion.choices[0].message.content
        return output

    def show(self):
        for choice in self.completion.choices:
            print(choice.message.content)
```

Código-fonte 5 – Model OpenIA

```
import os
from openai import OpenAI
```

```
3
4  API_KEY   =   os.getenv('OPENAI_API_KEY')
5
6  OpenaiClient = OpenAI(
7      api_key=API_KEY
8  )
9
10 __all__ = ['OpenaiClient']
```

Código-fonte 6 – API OpenIA

APÊNDICE H – TRANSCRIÇÃO DA ENTREVISTA SEMIESTRUTURADA DO EXPERIMENTO CONTROLADO

Entrevista — Professor do Ensino Médio

Pesquisador/Entrevistador: Boa tarde, professor. Primeiro, gostaria de agradecer sua participação na nossa pesquisa. Esta entrevista semi-estruturada tem o objetivo de compreender suas percepções e experiências com a utilização da nossa ferramenta para o ensino e avaliação do pensamento computacional.

Apesar de estar sendo gravada, garantimos o anonimato das respostas e o sigilo das informações. Todas as respostas serão vinculadas apenas ao perfil de um professor do ensino médio, a menos que o senhor nos autorize a utilizar seu nome. Tudo bem?

Professor: Tudo bem.

Pesquisador/Entrevistador: Professor, vamos começar com uma pergunta de cunho mais pessoal. Qual é sua formação acadêmica e há quanto tempo atua no ensino de disciplinas relacionadas à programação?

Professor: Muito bem. Sou formado em Engenharia de Computação pela UFC, tenho pós-graduação em Engenharia de Software e mestrado em Engenharia Elétrica com foco em Computação. Estou nesta escola desde 2022.

Pesquisador/Entrevistador: Você já utilizou alguma outra ferramenta que realizasse avaliação automatizada de código? Se sim, quais?

Professor: Já utilizei ferramentas de análise de código, mas não com meus alunos, apenas em projetos de pesquisa na universidade.

Pesquisador/Entrevistador: Comparando com métodos tradicionais, você acha que a ferramenta que propusemos contribuiu para otimizar o tempo dedicado à avaliação dos códigos? Houve redução do tempo necessário para fornecer feedback aos alunos?

Professor: Sim. Percebi que os relatórios ajudam muito, pois permitem visualizar, de forma mais técnica, os conceitos que os alunos utilizaram nas respostas. Gostei bastante dessa parte.

Pesquisador/Entrevistador: Você consideraria utilizar essa ferramenta de maneira mais regular na sua prática docente?

Professor: Com certeza.

Pesquisador/Entrevistador: Você percebeu se os feedbacks gerados foram coe-

rentes com a sua análise enquanto avaliador? Havia coerência nos feedbacks fornecidos pela ferramenta em cada dimensão analisada?

Professor: Sim, percebi coerência. Em algumas atividades, por exemplo, na parte de conceituação de estruturas de repetição, havia feedbacks corretos e outros nem tanto. Mas, de forma geral, ajudou a identificar quando o aluno não utilizou corretamente determinada estrutura ou, em outros casos, aspectos como decomposição funcional ou tratamento de erros.

Pesquisador/Entrevistador: Você acredita que a ferramenta conseguiu identificar corretamente aspectos das quatro dimensões do pensamento computacional — abstração, decomposição, reconhecimento de padrões e algoritmos?

Professor: Não consegui perceber isso em todas as dimensões, mas em algumas ficou bem evidente.

Pesquisador/Entrevistador: Houve algum caso em que você discordou significativamente do feedback fornecido pela ferramenta? Algum exercício ou resposta de estudante em que tenha tido opinião diferente?

Professor: Não. Achei que a ferramenta relatou perfeitamente o que o aluno havia feito ou deixado de fazer.

Pesquisador/Entrevistador: De que forma você acha que essa ferramenta impactaria sua prática pedagógica no dia a dia? Onde acha que ela poderia trazer impacto positivo?

Professor: Primeiramente, na discussão em sala de aula. Ganharíamos muito tempo, pois os relatórios já trazem os erros mais comuns. A aula ficaria mais objetiva e interativa, permitindo abordar diretamente as dificuldades específicas de cada aluno.

Além disso, ajudaria na minha tomada de decisão como professor. Se percebo que muitos alunos não conseguiram atingir um objetivo em determinada questão, já sei que preciso revisar o assunto ou explicar de outra forma. Isso torna meu trabalho mais eficiente.

Também vejo utilidade na criação de novos exercícios baseados nos pontos em que os alunos mais erraram.

Pesquisador/Entrevistador: Em relação à ferramenta e aos feedbacks fornecidos, quais aspectos você considera que poderiam ser melhorados? Onde acha que ela poderia oferecer algo a mais ou onde está oferecendo informação em excesso?

Professor: Os relatórios são bem completos e mostram claramente o que o aluno acertou ou precisa melhorar. No entanto, poderiam ter uma apresentação mais visual, pois atualmente são muito literais. Um gráfico ou outra forma visual facilitaria a leitura e a análise.

Além disso, penso que seria interessante ter uma comparação entre a solução do aluno e uma questão-modelo, mostrando quais pontos eram esperados e quais não foram atendidos. Isso ajudaria tanto o aluno quanto o professor a identificar as diferenças e promoveria um feedback ainda mais direcionado.

Pesquisador/Entrevistador: Apesar de estarmos falando de programação e pensamento computacional, você vê potencial para expansão desse tipo de ferramenta para outros contextos, disciplinas ou níveis de ensino?

Professor: Sim. Vejo aplicação possível em disciplinas como cálculo ou física, onde também existem diferentes métodos para resolver problemas. Professores das áreas de exatas poderiam utilizar relatórios semelhantes para tomar decisões pedagógicas com seus alunos.

Pesquisador/Entrevistador: Gostaria de deixar algum comentário ou consideração final sobre o uso da ferramenta, ou abordar algum ponto que não foi mencionado nesta conversa?

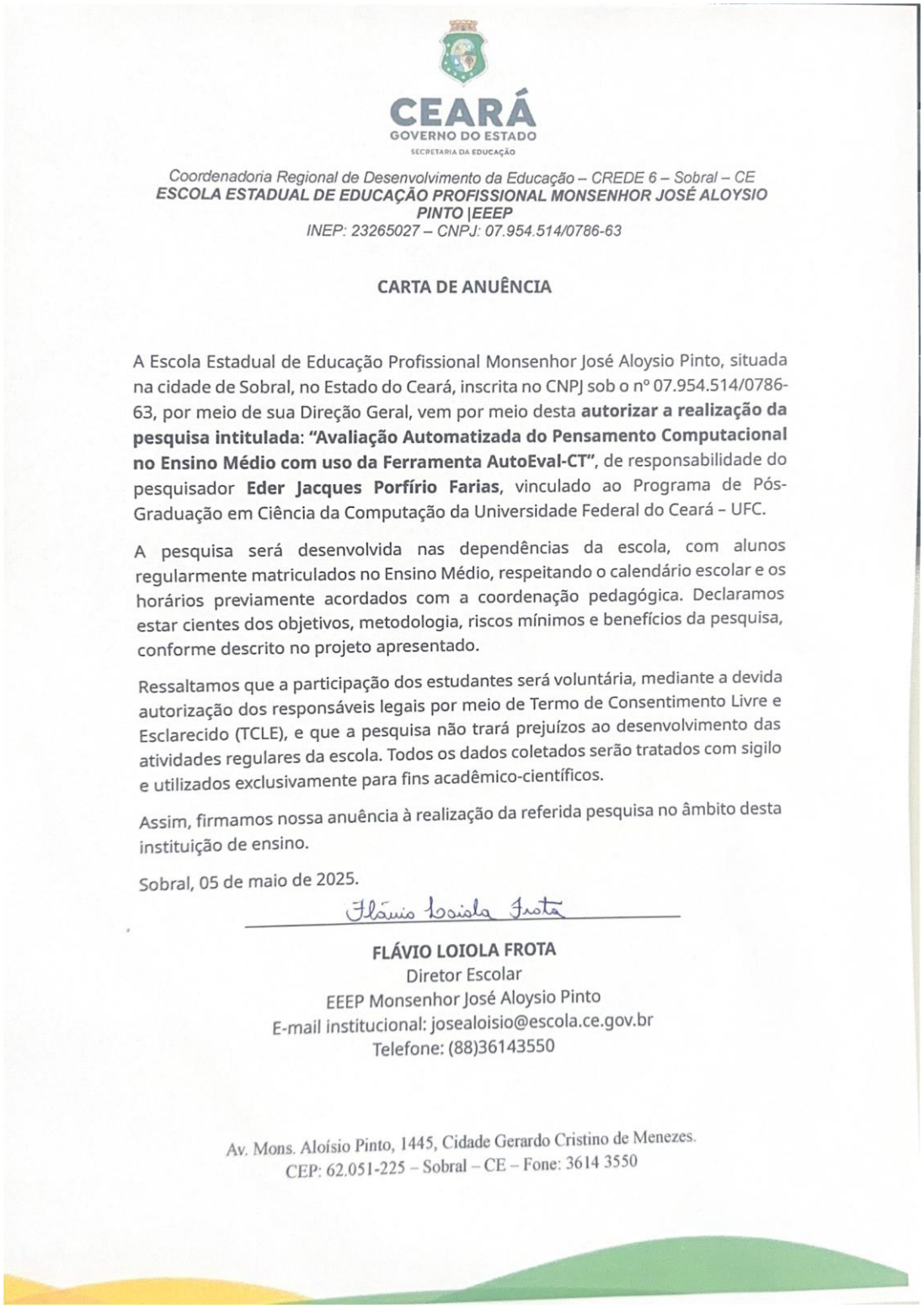
Professor: Sim. Esses relatórios proporcionam uma visão técnica que permite ao professor filtrar o nível de cada aluno pelas questões que eles resolveram. Eles ampliam o olhar do professor para aspectos que, na correria do dia a dia, podem passar despercebidos.

Além disso, ajudam muito na mediação pedagógica. Costumo fazer, após provas práticas, uma aula para comentar os erros e acertos mais frequentes. Com os relatórios, essa mediação ficaria muito mais rica, permitindo ir direto ao ponto do que o aluno errou ou acertou. Isso torna o aprendizado mais eficiente, evitando que o aluno fique protelando dificuldades por vários bimestres ou semestres.

Pesquisador/Entrevistador: Muito obrigado, professor. Agradeço novamente sua participação, que é muito importante para nossa pesquisa e para a validação do artefato que estamos desenvolvendo. Vou desligar a gravação agora.

ANEXO A – DOCUMENTOS DO EXPERIMENTO CONTROLADO

A.1 Carta de Anuência



CEARÁ
GOVERNO DO ESTADO
SECRETARIA DA EDUCAÇÃO

Coordenadoria Regional de Desenvolvimento da Educação – CREDE 6 – Sobral – CE
ESCOLA ESTADUAL DE EDUCAÇÃO PROFISSIONAL MONSENHOR JOSÉ ALOYSIO PINTO \|EEEP
INEP: 23265027 – CNPJ: 07.954.514/0786-63

CARTA DE ANUÊNCIA

A Escola Estadual de Educação Profissional Monsenhor José Aloysio Pinto, situada na cidade de Sobral, no Estado do Ceará, inscrita no CNPJ sob o nº 07.954.514/0786-63, por meio de sua Direção Geral, vem por meio desta **autorizar a realização da pesquisa intitulada**: **"Avaliação Automatizada do Pensamento Computacional no Ensino Médio com uso da Ferramenta AutoEval-CT"**, de responsabilidade do pesquisador **Eder Jacques Porfírio Farias**, vinculado ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará – UFC.

A pesquisa será desenvolvida nas dependências da escola, com alunos regularmente matriculados no Ensino Médio, respeitando o calendário escolar e os horários previamente acordados com a coordenação pedagógica. Declaramos estar cientes dos objetivos, metodologia, riscos mínimos e benefícios da pesquisa, conforme descrito no projeto apresentado.

Ressaltamos que a participação dos estudantes será voluntária, mediante a devida autorização dos responsáveis legais por meio de Termo de Consentimento Livre e Esclarecido (TCLE), e que a pesquisa não trará prejuízos ao desenvolvimento das atividades regulares da escola. Todos os dados coletados serão tratados com sigilo e utilizados exclusivamente para fins acadêmico-científicos.

Assim, firmamos nossa anuência à realização da referida pesquisa no âmbito desta instituição de ensino.

Sobral, 05 de maio de 2025.

Flávio Loiola Frota

FLÁVIO LOIOLA FROTA
Diretor Escolar
EEEP Monsenhor José Aloysio Pinto
E-mail institucional: josealoisio@escola.ce.gov.br
Telefone: (88)36143550

Av. Mons. Aloísio Pinto, 1445, Cidade Gerardo Cristino de Menezes.
CEP: 62.051-225 – Sobral – CE – Fone: 3614 3550

A.2 Parecer Comitê de Ética em Pesquisa - Versão 1

UNIVERSIDADE ESTADUAL
VALE DO ACARAÚ - UVA/CE

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Automatic Evaluation for Computational Thinking}): Um Artefato de Avaliação Automatizada do Pensamento Computacional

Pesquisador: EDER JACQUES PORFIRIO FARIAS

Área Temática:

Versão: 1

CAAE: 88541225.6.0000.5053

Instituição Proponente: Universidade Estadual Vale do Acaraú - UVA

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 7.620.580

Apresentação do Projeto:

Trata-se de um estudo de caso exploratório, de natureza quali-quantitativa, que visa validar empiricamente um artefato computacional desenvolvido para avaliação automatizada do Pensamento Computacional (PC) em alunos do Ensino Médio.

O estudo será realizado com 23 alunos da Escola Estadual de Educação Profissional Monsenhor José Aloysio Pinto, situada no estado do Ceará, como parte da etapa de experimentação de uma tese de doutorado em Ciência da Computação da Universidade Federal do Ceará.

O artefato, denominado AutoEval-CT (Automatic Evaluation for Computational Thinking), foi desenvolvido com base nos princípios do Design Science Research (DSR) e utiliza modelos de linguagem natural (Large Language Models LLMs) para analisar códigos em linguagem Python
gerados por estudantes. O sistema identifica, de forma automatizada, evidências de competências relacionadas ao PC como abstração, decomposição, reconhecimento de padrões e pensamento algorítmico fornecendo relatórios com feedback estruturado ao docente.

Metodologia de Análise de Dados:

A análise dos dados será realizada por meio de uma abordagem quali-quantitativa, contemplando tanto aspectos estatísticos quanto descritivo-interpretativos.

Endereço: Av Comandante Maurocélio Rocha Ponte, 150
Bairro: Derby **CEP:** 62.041-040
UF: CE **Município:** SOBRAL
Telefone: (88)3677-4255 **Fax:** (88)3677-4242 **E-mail:** cep_uva@uvanet.br

A.3 Parecer Comitê de Ética em Pesquisa - Versão 2

UNIVERSIDADE ESTADUAL
VALE DO ACARAÚ - UVA/CE

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Automatic Evaluation for Computational Thinking}): Um Artefato de Avaliação Automatizada do Pensamento Computacional

Pesquisador: EDER JACQUES PORFIRIO FARIAS

Área Temática:

Versão: 2

CAAE: 88541225.6.0000.5053

Instituição Proponente: Universidade Estadual Vale do Acaraú - UVA

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 7.722.336

Apresentação do Projeto:

Vide parecer anterior.

Objetivo da Pesquisa:

Vide parecer anterior.

Avaliação dos Riscos e Benefícios:

Vide parecer anterior.

Comentários e Considerações sobre a Pesquisa:

Vide parecer anterior.

Considerações sobre os Termos de apresentação obrigatória:

Vide parecer anterior.

Recomendações:

Vide parecer anterior.

Conclusões ou Pendências e Lista de Inadequações:

As pendências observadas anterioremente foram resolvidas

Considerações Finais a critério do CEP:

As pendências observadas anterioremente foram resolvidas

Endereço: Av Comandante Maurocélio Rocha Ponte, 150
Bairro: Derby **CEP:** 62.041-040
UF: CE **Município:** SOBRAL
Telefone: (88)3677-4255 **Fax:** (88)3677-4242 **E-mail:** cep_uva@uvanet.br