UNIVERSIDADE FEDERAL FLUMINENSE

ANTONIO REGE LOPES DOS SANTOS

SOPENSA: UM FRAMEWORK PARA O ENSINO DE PROGRAMAÇÃO COM BASE NO PENSAMENTO COMPUTACIONAL E SOFT SKILLS

NITERÓI

2026

ANTONIO REGE LOPES DOS SANTOS

SOPENSA: UM FRAMEWORK PARA O ENSINO DE PROGRAMAÇÃO COM BASE NO PENSAMENTO COMPUTACIONAL E SOFT SKILLS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito para obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação.

Orientadora:
Luciana Cardoso de Castro Salgado

Coorientador:
José Viterbo Filho

NITERÓI

2026

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S237s Santos, Antonio Rege Lopes dos
SoPensa : um framework para o ensino de programação com base no Pensamento Computacional e Soft Skills / Antonio Rege Lopes dos Santos. - 2026.
233 f.

Orientador: Luciana Cardoso de Castro Salgado.
Coorientador: José Viterbo Filho.
Tese (doutorado)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2026.

1. Aplicação de programação. 2. Educação profissional. 3. Informática na educação. 4. Framework (Programa de computador). 5. Produção intelectual. I. Salgado, Luciana Cardoso de Castro, orientadora. II. Filho, José Viterbo, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

ANTONIO REGE LOPES DOS SANTOS

SOPENSA: UM FRAMEWORK PARA O ENSINO DE PROGRAMAÇÃO COM BASE NO PENSAMENTO COMPUTACIONAL E SOFT SKILLS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito para obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação.

Aprovada em Agosto de 2026.

BANCA EXAMINADORA

Profa. Dra. Luciana C. de Castro Salgado – Orientadora, UFF

Prof. Dr. José Viterbo Filho – Coorientador, UFF

Profa. Dra. Flavia Cristina Bernardini, UFF

Profa. Dra. Daniela Gorski Trevisan, UFF

Profa. Dra. Natasha Malveira Costa Valentim, UFPR

Prof. Dr. Cristiano Maciel, UFMT

Niterói

2026

Agradecimentos

Registro minha gratidão a Deus, por sua graça e misericórdia ao longo desta caminhada. Foi Ele quem me fortaleceu nos momentos de cansaço, sustentou-me nas dificuldades e me deu ânimo para seguir em frente.

Aos meus pais, Manoel Araújo dos Santos e Maria de Jesus Lopes dos Santos, agradeço pelo apoio, pelos ensinamentos e pelas orações.

À minha esposa, Maria Lucimar Araújo Monteiro, e ao meu filho, Lucas Lopes Monteiro, agradeço pelo amor, pela compreensão e pelo apoio em minhas ausências durante os estudos e na elaboração deste trabalho.

À minha orientadora, Prof.ª Dra. Luciana C. de Castro Salgado, e ao meu coorientador, Prof. Dr. José Viterbo Filho, agradeço pela orientação, pela generosidade no compartilhamento de conhecimentos e pelo acompanhamento durante a pesquisa. Suas contribuições foram fundamentais para o amadurecimento deste trabalho e para a concretização desta tese.

Aos professores que participaram do exame de qualificação e da defesa, agradeço pela disponibilidade, pela atenção e pelas contribuições valiosas. Seus apontamentos e questionamentos contribuíram para o aperfeiçoamento deste trabalho e para a construção de sua versão final.

Ao Instituto Federal do Acre, Campus Rio Branco, onde atuo como docente, agradeço pelo apoio institucional e pela colaboração dos docentes e estudantes dos cursos da área de Informática nesta pesquisa.

À Universidade Federal do Acre e à Universidade Federal Fluminense, em especial ao Programa de Pós-Graduação em Ciência da Computação, agradeço pela parceria institucional, pela oportunidade de formação e pelo aprendizado proporcionado.

Por fim, agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho.

Resumo

A educação profissional técnica de nível médio enfrenta desafios no ensino de programação, especialmente pela dificuldade dos estudantes com abstração e pela carência de abordagens que apoiem o professor. Embora a Base Nacional Comum Curricular tenha incorporado a Computação na Educação Básica, sua implementação no ensino técnico integrado ao ensino médio ainda carece de diretrizes práticas que orientem o professor no desenvolvimento do Pensamento Computacional e das *Soft Skills*. Esta tese teve como objetivo analisar como o Pensamento Computacional e as *Soft Skills* podem apoiar o ensino de programação, além de propor e analisar as contribuições do *framework* SoPensa, um modelo conceitual-operacional voltado à mediação docente, fundamentado na Aprendizagem Significativa e no Construcionismo e organizado em quatro fases. A pesquisa, aplicada e do tipo pesquisa-ação, adotou abordagem predominantemente qualitativa com dados quantitativos descritivos complementares. O *framework* foi inicialmente apreciado por dois especialistas. Em seguida, foi aplicado por dois professores em duas turmas de Lógica de Programação de um Instituto Federal. Os dados foram coletados por meio de entrevistas com os professores e questionário respondido por 64 estudantes. A análise qualitativa seguiu a Análise Temática Reflexiva. Os resultados indicaram que a integração entre Pensamento Computacional e *Soft Skills* não garantiu ganhos de aprendizagem, mas reorganizou a prática docente ao tornar explícitos aspectos antes implícitos. Os componentes do Pensamento Computacional foram mais reconhecidos pelos estudantes do que as *Soft Skills*, cuja percepção foi mais neutra, e as avaliações dos professores sobre elas divergiram conforme as condições de acompanhamento. Já a abstração foi o componente de maior dificuldade, com 48,4% de concordância, bem abaixo dos demais. Concluiu-se que a integração entre esses eixos apoia o ensino de programação primeiro pela reorganização da mediação docente e depois pelo desenvolvimento do estudante. Assim, o *framework* SoPensa contribui ao transformar práticas pouco sistematizadas em procedimentos organizados e ao mostrar a dificuldade com a abstração na passagem do raciocínio para o código, preenchendo uma lacuna de propostas que integrem esses eixos no ensino técnico.

Palavras-chave: Ensino de Programação; Pensamento Computacional; Soft Skills, Educação Profissional Técnica; Framework SoPensa; Mediação Docente.

Abstract

Technical and vocational secondary education faces challenges in programming education, particularly students' difficulty with abstraction and the scarcity of approaches that support teachers. Although Brazil's National Common Curricular Base has incorporated Computing into Basic Education, its implementation in technical education integrated with secondary education still lacks practical guidelines to help teachers develop Computational Thinking and soft skills. This thesis aimed to analyze how Computational Thinking and soft skills can support programming education at this level, as well as to propose and analyze the contributions of the SoPensa framework, a conceptual-operational model focused on teaching mediation, grounded in Meaningful Learning and Constructionism and organized into four phases. This applied study followed an action-research design and adopted a predominantly qualitative approach with complementary descriptive quantitative data. The framework was first reviewed by two experts. It was then applied by two teachers in two Programming Logic classes at a Federal Institute. Data were collected through interviews with the teachers and a questionnaire answered by 64 students. The qualitative analysis followed Reflexive Thematic Analysis. The results indicated that integrating Computational Thinking and soft skills did not guarantee learning gains but reorganized teaching practice by making explicit aspects that had previously remained implicit. The components of Computational Thinking were more readily recognized by students than the soft skills, whose perception was more neutral, and teachers' assessments of them diverged according to the monitoring conditions. Abstraction, in turn, was the component with the greatest difficulty, with 48.4% agreement, well below the others. It was concluded that integrating these two dimensions supports programming education first through the reorganization of teaching mediation and then through student development. Thus, the SoPensa framework contributes by transforming poorly systematized practices into organized procedures and by revealing the difficulty with abstraction in the transition from reasoning to code, filling a gap in proposals that integrate these two dimensions in technical education.

Keywords: Programming Education; Computational Thinking; Soft Skills; Technical and Vocational Education; SoPensa Framework; Teaching Mediation.

Lista de ilustrações

Fig. 1 Ciclo metodológico da pesquisa-ação adotado no estudo 17
Fig. 2 A assimilação ausubeliana 27
Fig. 3 Os quatro componentes do Pensamento Computacional 32
Fig. 4 Decomposição: divisão de um problema em partes menores 33
Fig. 5 Reconhecimento de padrões: identificação do padrão e de suas variações 34
Fig. 6 Abstração: redução de detalhes, mantendo características essenciais . 35
Fig. 7 Algoritmo: sequência de instruções para resolver um problema 35
Fig. 8 Visão geral das *Soft Skills* requeridas nos anúncios 43
Fig. 9 Itinerário da educação profissional técnica de nível médio 47
Fig. 10 Distribuição das publicações por ano 58
Fig. 11 Distribuição dos estudos por nível de ensino 59
Fig. 12 Frequência dos componentes de PC e SS 61
Fig. 13 Etapas metodológicas do estudo empírico 67
Fig. 14 Distribuição dos participantes por sexo e região 70
Fig. 15 Perfil acadêmico e profissional 71
Fig. 16 Infraestrutura e componentes do PC 72
Fig. 17 Competências de SS e metodologias ativas 72
Fig. 18 Engajamento e seguir carreira na área 73
Fig. 19 Condições de ensino relacionadas ao tempo e à sobrecarga dos alunos 74
Fig. 20 Fatores relacionados à matemática e à participação nas aulas 75
Fig. 21 Frequência das *Soft Skills* mais mencionadas pelos professores 78
Fig. 22 Conhecimento dos professores sobre PC 80
Fig. 23 Principais componentes do PC e de SS considerados para o *framework* 91
Fig. 24 Fluxo conceitual do *framework* SoPensa 94
Fig. 25 Fluxo operacional do *framework* SoPensa 98
Fig. 26 Síntese da linha do tempo da aplicação do SoPensa 106
Fig. 27 Distribuição dos componentes de PC e competências de SS por turma 109
Fig. 28 Mapa de calor dos componentes de PC e das competências de SS em dois times representativos 112
Fig. 29 Processo de construção de uma casa: exemplo de decomposição 114

Fig. 30 Reconhecimento de padrões a partir de um modelo base 114
Fig. 31 Abstração: simplificação progressiva 114
Fig. 32 Algoritmo em Portugol 115
Fig. 33 Esquema da atividade Robô HidroLógico 116
Fig. 34 Caixas de Missões Lógicas e percurso até o cofre lógico 118
Fig. 35 Estudantes durante a atividade Caixas de Missões Lógicas 119
Fig. 36 Projetos desenvolvidos no Scratch: cenário da Travessia do Rio Açaí e blocos de programação 121
Fig. 37 Exemplo de programa desenvolvido pelos estudantes em Python . . . 122
Fig. 38 Estudantes no laboratório de informática durante a atividade em Python 123
Fig. 39 Percurso da análise temática reflexiva adotada no estudo 131
Fig. 40 Temas e códigos da análise temática reflexiva 133
Fig. 41 Percepção dos estudantes sobre a aceitação do SoPensa 148
Fig. 42 Percepção dos estudantes sobre o desenvolvimento do PC 150
Fig. 43 Percepção dos estudantes sobre o desenvolvimento de SS 151
Fig. 44 Etapas do MSL 189
Fig. 45 Processo de seleção dos estudos 194
Fig. 46 Nível de adequação das dimensões (PC, SS e Geral) por estudante e por time, em cada curso 209
Fig. 47 Página inicial do sítio eletrônico do SoPensa 221
Fig. 48 Roteiro de entrada para aplicação do SoPensa apresentado no sítio . . 222
Fig. 49 Entrada das respostas do diagnóstico na ferramenta de formação de times 222
Fig. 50 Times gerados pela ferramenta a partir das médias de PC e de SS . . 223
Fig. 51 Capa do Projeto Pedagógico do Curso Técnico em Informática para Internet 224
Fig. 52 Ementa da disciplina de Lógica de Programação do Curso Técnico em Informática para Internet 225
Fig. 53 Capa do Projeto Pedagógico do Curso Técnico em Redes de Computadores 226
Fig. 54 Ementa da disciplina de Lógica de Programação do Curso Técnico em Redes de Computadores 227

Lista de Tabelas

Tabela1 Definições de *Soft Skills* . . . 40
Tabela2 Estudos selecionados . . . 53
Tabela3 Abordagens, ferramentas e fundamentação teórica dos estudos . . . 60
Tabela4 Estrutura do questionário aplicado aos professores . . . 67
Tabela5 Principais categorias do roteiro de entrevistas . . . 68
Tabela6 Nível de conhecimento sobre SS . . . 77
Tabela7 Categorias de uso dos componentes de PC . . . 81
Tabela8 Quadro de decisão dos componentes de PC . . . 90
Tabela9 Quadro de decisão das competências de SS . . . 91
Tabela10 Síntese da literatura e das percepções dos professores sobre PC . . . 92
Tabela11 Síntese da literatura e das percepções dos professores sobre SS . . . 92
Tabela12 Relação entre as teorias de aprendizagem e as fases do *framework* SoPensa . . . 93
Tabela13 Fase 1: Diagnosticar e Organizar . . . 94
Tabela14 Fase 2: Contextualizar e Ambientar . . . 95
Tabela15 Fase 3: Desafiar e Mediar . . . 96
Tabela16 Fase 4: Analisar e Consolidar . . . 96
Tabela17 Resultados da apreciação do *framework* SoPensa pelos especialistas . 103
Tabela18 Caracterização dos professores participantes . . . 104
Tabela19 Distribuição dos estudantes por componente do PC e competência de SS . . . 108
Tabela20 Times do curso de Informática para Internet . . . 111
Tabela21 Times do curso de Informática em Redes . . . 111
Tabela22 Papéis e responsabilidades na atividade Robô HidroLógico . . . 116
Tabela23 Etapas da atividade Robô HidroLógico . . . 117
Tabela24 Etapas da atividade Caixas de Missões Lógicas . . . 118
Tabela25 Etapas do ciclo formativo seguido nas atividades em Scratch e Python 121
Tabela26 Desafios propostos em Python na Fase 3 . . . 122
Tabela27 Autoavaliação dos times por dimensão - síntese das duas turmas . . . 125
Tabela28 Perguntas e objetivos do MSL . . . 190

Tabela29 Estratégia PICO 191
Tabela30 Estudos de referência consultados na construção da *string* 191
Tabela31 *String* de busca 192
Tabela32 Bases de dados utilizadas no mapeamento 192
Tabela33 Aplicação da *string* de busca nas bases de dados 193
Tabela34 Matriz de extração dos 18 estudos 195
Tabela35 Instrumento de apreciação do *framework* SoPensa por especialistas . 202
Tabela36 Rubrica de acompanhamento de PC e SS 206

Lista de Abreviaturas e Siglas

AS Aprendizagem Significativa

ATR Análise Temática Reflexiva

BNCC Base Nacional Comum Curricular

BPMN Business Process Model and Notation

EPTNM Educação Profissional Técnica de Nível Médio

HS *Hard Skills*

IFAC Instituto Federal do Acre

IFs Institutos Federais

MEC Ministério da Educação

MSL Mapeamento Sistemático da Literatura

OE Objetivos Específicos

PC Pensamento Computacional

SS *Soft Skills*

SUMÁRIO

1 INTRODUÇÃO 12

1.1 Problema de pesquisa . . . 14

1.2 Objetivos . . . 14

1.3 Metodologia . . . 15

1.4 Organização da tese . . . 19

2 FUNDAMENTAÇÃO TEÓRICA 21

2.1 Ensino e aprendizagem de programação . . . 21

2.1.1 Dificuldades no ensino e na aprendizagem de programação . . . 22

2.1.2 Metodologias ativas no ensino de programação . . . 24

2.2 Teorias de aprendizagem aplicadas ao ensino de programação . . . 25

2.2.1 Aprendizagem significativa . . . 26

2.2.2 Teoria construcionista . . . 28

2.2.3 Implicações das teorias no ensino de programação . . . 29

2.3 Pensamento Computacional . . . 29

2.3.1 Componentes do Pensamento Computacional . . . 31

2.3.2 Pensamento Computacional como suporte à aprendizagem de programação . . . 36

2.3.3 Computação Desplugada como estratégia para o desenvolvimento do PC . . . 37

2.4 Conceitos de *Soft Skills* . . . 38

2.4.1 *Soft Skills* na educação em computação . . . 41

2.4.2 *Soft Skills* no mercado de trabalho . . . 42

2.5 Integração entre PC e SS no ensino de programação . . . 44

2.6 Educação profissional técnica no Brasil . . . 46

2.7 Considerações finais do capítulo . . . 49

3 ESTADO DA ARTE 52

3.1 Caracterização dos estudos selecionados . . . 52

3.2 Resultados do mapeamento sistemático . . . 57

3.2.1 PS1: distribuição temporal das publicações . . . 58

3.2.2 PS2: contextos educacionais investigados . . . 59

3.2.3 PS3: abordagens, ferramentas e base teórica . . . 59
3.2.4 PS4: componentes do PC e competências de SS mais abordados . . . 60
3.2.5 PS5: desafios reportados na literatura . . . 61
3.2.6 Discussão dos resultados do MSL . . . 62
3.3 Considerações finais do capítulo . . . 63

4 ESTUDO EMPÍRICO COM PROFESSORES DOS INSTITUTOS FEDERAIS 65
4.1 Procedimentos metodológicos . . . 65
4.1.1 Contexto . . . 65
4.1.2 Participantes . . . 66
4.1.3 Critérios de inclusão e exclusão . . . 66
4.2 Método de coleta de dados . . . 66
4.2.1 Construção e aplicação do questionário . . . 67
4.2.2 Entrevistas semiestruturadas . . . 68
4.3 Análise dos dados . . . 69
4.3.1 Análise das respostas do questionário . . . 70
4.3.1.1 Perfil demográfico dos participantes . . . 70
4.3.1.2 Perfil acadêmico e profissional . . . 70
4.3.1.3 Percepções e desafios no ensino de programação . . . 71
4.3.1.4 Análise das respostas abertas . . . 75
4.3.2 Análise das entrevistas com os professores . . . 76
4.3.2.1 Percepção sobre SS no ensino de programação . . . 76
4.3.2.2 Percepção sobre componentes de PC . . . 79
4.3.2.3 Sugestões para implementação do *framework* . . . 82
4.4 Resultados e discussão . . . 84
4.5 Considerações finais do capítulo . . . 87

5 SOPENSA: *FRAMEWORK* COM BASE NO PENSAMENTO COMPUTACIONAL E *SOFT SKILLS* 89
5.1 Estrutura e componentes do *framework* SoPensa . . . 89
5.1.1 Descrição das fases e ações do professor . . . 94
5.1.2 Fluxo de aplicação do SoPensa . . . 97
5.2 Operacionalização do *framework* SoPensa . . . 99
5.2.1 Instrumentos para o acompanhamento docente . . . 99
5.3 Considerações finais do capítulo . . . 100

6 APLICAÇÃO DO SOPENSA 101
6.1 Apreciação do *framework* por especialistas . . . 101
6.2 Contexto e caracterização do ambiente de estudo . . . 103
6.2.1 Professores participantes . . . 104
6.2.2 Descrição das turmas . . . 105
6.2.3 Condições de aplicação . . . 105
6.3 Relato de aplicação do *framework* SoPensa . . . 105
6.3.1 Preparação da equipe de trabalho . . . 107
6.3.2 Planejamento das atividades . . . 107
6.3.3 Fase 1: Diagnosticar e Organizar . . . 107
6.3.3.1 Resultados do diagnóstico por turma . . . 108
6.3.3.2 Distribuição percentual de PC e SS por turma . . . 109
6.3.3.3 Formação dos times equilibrados . . . 109
6.3.3.4 Mapa de calor dos componentes de PC e competências de SS por time . . . 111
6.3.3.5 Acompanhamento da fase . . . 112
6.3.4 Fase 2: Contextualizar e Ambientar . . . 113
6.3.4.1 Planejamento e estratégias didáticas . . . 113
6.3.4.2 Atividade desplugada: Robô HidroLógico . . . 115
6.3.4.3 Atividade desplugada: Caixas de Missões Lógicas . . . 117
6.3.4.4 Acompanhamento da fase . . . 119
6.3.5 Fase 3: Desafiar e Mediar . . . 120
6.3.5.1 Preparação para as práticas plugadas . . . 120
6.3.5.2 Atividade plugada: A Travessia do Rio Açaí . . . 121
6.3.5.3 Atividade plugada: Desafios em Python . . . 122
6.3.5.4 Acompanhamento da fase . . . 123
6.3.6 Fase 4: Analisar e Consolidar . . . 124
6.3.6.1 Análise das produções e retomada . . . 124
6.3.6.2 Autoavaliação dos times . . . 125
6.3.6.3 Ajustes ao *framework* . . . 126
6.4 Considerações finais do capítulo . . . 126

7 PERCEPÇÕES DE PROFESSORES E ESTUDANTES SOBRE O SOPENSA 128
7.1 Percepções dos professores: análise temática reflexiva . . . 129
7.1.1 Procedimentos de análise . . . 129
7.1.2 T1. O SoPensa reorganiza a prática docente . . . 133

7.1.3 T2. O SoPensa aproxima o aluno da disciplina . . . 136
7.1.4 T3. O SoPensa e a estruturação do raciocínio antes da codificação . . . 138
7.1.5 T4. O SoPensa e a percepção divergente sobre as *soft skills* . . . 141
7.1.6 T5. O SoPensa requer condições e ajustes pedagógicos . . . 143
7.1.7 Eixo analítico complementar. Potencial pedagógico . . . 145
7.2 Percepções dos estudantes . . . 147
7.2.1 Procedimentos de análise . . . 147
7.2.2 Aceitação do SoPensa . . . 148
7.2.3 Desenvolvimento do PC . . . 150
7.2.4 Desenvolvimento de SS . . . 151
7.3 Considerações finais do capítulo . . . 152

8 DISCUSSÃO DOS RESULTADOS 154
8.1 A abstração como dificuldade persistente . . . 154
8.2 Os componentes do PC são mais reconhecidos do que as competências de SS 156
8.3 A progressão das atividades desplugadas para as plugadas . . . 158
8.4 Colaboração e comunicação como competências que demandam mediação intencional . . . 160
8.5 O *framework* na prática docente e os condicionantes da educação profissional técnica . . . 161
8.6 Considerações finais do capítulo . . . 163

9 CONSIDERAÇÕES FINAIS 164
9.1 Retomada do problema e dos objetivos . . . 164
9.2 Contribuições . . . 165
9.3 Limitações da pesquisa . . . 167
9.4 Trabalhos futuros . . . 168
9.5 Produções acadêmicas . . . 169

REFERÊNCIAS 171

Apêndice A – Protocolo do Mapeamento Sistemático da Literatura 189
A.1 Planejamento . . . 189
A.1.1 Questões de pesquisa . . . 190
A.1.2 Estratégia de busca . . . 190
A.1.3 String de busca . . . 191
A.1.4 Base de dados . . . 192

A.1.5 Critérios de inclusão e exclusão 192
A.1.6 Justificativa do recorte temporal (2020 a 2024) 193
A.2 Condução 193
A.2.1 Resultados das buscas 194
A.3 Síntese dos resultados 194

Apêndice B – Matriz de Extração dos Estudos **195**

Apêndice C – Instrumentos do Estudo Empírico com os Professores dos Institutos Federais **197**

Apêndice D – Instrumento de apreciação do SoPensa por especialistas **202**

Apêndice E – Instrumentos de Diagnóstico e Acompanhamento Docente do SoPensa **203**

Apêndice F – Mapa de calor por estudante, agrupado por time, segundo as dimensões de (PC, SS e Geral), nos cursos de Informática para Internet e Informática em Redes **209**

Apêndice G – Questionário de percepção dos estudantes sobre o *framework* SoPensa **210**

Apêndice H – Roteiro da Entrevista com os Professores após a Aplicação do Framework SoPensa **213**

Apêndice I – Termos de Consentimento e de Assentimento **215**
I.1 Termo de Consentimento Livre e Esclarecido (TCLE) – Professor 215
I.2 Termo de Consentimento Livre e Esclarecido (TCLE) – Responsáveis 217
I.3 Termo de Assentimento Livre e Esclarecido (TALE) – Estudante 219

Apêndice J – Registro do sítio eletrônico do SoPensa **221**

Anexo A – Capa e Ementa do Curso Técnico Integrado ao Ensino Médio em Informática para Internet **224**

Anexo B – Capa e Ementa do PPC Técnico em Redes de Computadores **226**

1 INTRODUÇÃO

O ensino de programação tem sido associado, na literatura recente, ao desenvolvimento de competências necessárias ao século XXI, que envolvem habilidades técnicas, cognitivas e *Soft Skills* (SS) (Ismail; Nugroho; Rohayati, 2023; Wong; Cheung, 2020; Tomić *et al.*, 2019). Essa associação reflete o reconhecimento de que a formação de profissionais da área de computação não se limita ao domínio de raciocínio lógico, linguagens e ferramentas, mas envolve também SS relacionadas a atuação em equipe, comunicação e adaptação a diferentes contextos de trabalho.

Com o objetivo de atender a essa demanda, o Brasil incorporou, por meio da Base Nacional Comum Curricular (BNCC), em 2017, o desenvolvimento de SS em suas diretrizes educacionais e, em 2022, ampliou essa iniciativa com a inclusão do ensino de Computação na Educação Básica, visando, dentre outros, ao desenvolvimento de habilidades de Pensamento Computacional (PC) (Brasil, 2017a, 2022).

No campo acadêmico, a partir do artigo publicado em 2006, por Jeannette M. Wing, o PC tem sido amplamente discutido como uma abordagem que contribui para o desenvolvimento de componentes fundamentais à programação, como decomposição, reconhecimento de padrões, abstração e construção de algoritmos (Wing, 2006; Barr; Harrison; Conery, 2011; Brennan; Resnick *et al.*, 2012; Macrides; Miliou; Angeli, 2022). De forma complementar, as SS também têm sido apontadas como relevantes no processo de aprendizagem, especialmente no desenvolvimento de competências como colaboração, comunicação, criatividade e pensamento crítico, que influenciam diretamente a forma como os estudantes enfrentam problemas em programação (Figueiredo; García-Peñalvo, 2022; Kapustina; Matyushina, 2023; Wang; Herzog, 2023). Nesse sentido, o PC e as SS podem ser compreendidos como elementos complementares no ensino de programação.

Apesar do reconhecimento dessas contribuições, o ensino de programação permanece marcado por dificuldades persistentes, especialmente em relação à compreensão de conceitos abstratos e ao desenvolvimento do raciocínio lógico para a resolução de problemas.

Essas dificuldades podem comprometer o desempenho acadêmico dos estudantes e limitar sua permanência e progressão nos cursos da área de computação (Moraes; Costa; Scholz, 2022; Morais; Mendes Neto; Osório, 2020; Ciancarini; Missiroli; Russo, 2020; Figueiredo; García-Peñalvo, 2022). Além disso, mesmo quando atividades em grupo são propostas, não há garantia de que competências como colaboração e comunicação sejam efetivamente desenvolvidas, uma vez que demandam mediação intencional por parte do professor (Younis *et al.*, 2019; Hsu *et al.*, 2022).

Percebe-se, dessa forma, que a prática docente enfrenta desafios em desenvolver, de forma integrada, habilidades técnicas e SS, especialmente em cursos de computação da educação brasileira (SBC, 2025). Entre esses desafios, destacam-se aspectos como a adaptação de metodologias tradicionais, bem como a manutenção da motivação, da persistência e da qualidade da comunicação com os alunos (SBC, 2025; Medeiros; Ramalho; Falcão, 2018; Medeiros; Falcão; Ramalho, 2020, 2021). Embora se reconheça a importância de PC e SS no ensino de programação, na literatura, seu desenvolvimento ainda ocorre de maneira pouco sistematizada (Rege; Salgado; Viterbo, 2025). Em muitos casos, esses elementos são trabalhados de forma implícita, sem orientação pedagógica que apoie o professor em sua condução em sala de aula (Ciancarini; Missiroli; Russo, 2020; Nouri *et al.*, 2020; SBC, 2025; Maitz; Paleczek; Danielowitz, 2022).

Para enfrentar essas dificuldades, diversas estratégias têm sido propostas na educação infantil e no ensino fundamental, incluindo o uso de programação em blocos, robótica educacional, gamificação, computação desplugada e ambientes interativos (Yang, 2024; Wong; Cheung, 2020; Hsu *et al.*, 2022; Maitz; Paleczek; Danielowitz, 2022; Madariaga *et al.*, 2023; Valls Pou; Canaleta; Fonseca, 2022; Kukul; Çakır, 2020). Tais abordagens têm demonstrado potencial para favorecer o engajamento e a aprendizagem inicial dos estudantes, mas, em sua maioria, não oferecem orientação direta para a atuação do professor (Ciancarini; Missiroli; Russo, 2020; Maitz; Paleczek; Danielowitz, 2022; Yang, 2024).

Diante desse cenário, identifica-se uma lacuna na literatura quanto a abordagens pedagógicas estruturadas que apoiem o professor na integração entre PC e SS no ensino de programação, sobretudo na Educação Profissional Técnica de Nível Médio (EPTNM). Nessa modalidade, o ensino de programação integra a formação para o trabalho em um currículo orientado pela articulação entre formação geral e formação técnica. Essa formação requer dos estudantes tanto conhecimentos técnicos e habilidades cognitivas quanto competências de SS para a adaptação a situações profissionais.

1.1 Problema de pesquisa

No ensino de programação, ainda se verificam limitações no desenvolvimento dos componentes do PC e das SS. Na literatura, os estudos analisados focam em sua maioria no ensino fundamental, com presença reduzida nos demais níveis e ausência de investigações específicas sobre o ensino médio técnico. Além disso, a prática docente ocorre, em muitos casos, sem uma abordagem sistematizada para o desenvolvimento dessas dimensões, indicando uma lacuna na atuação do professor na EPTNM.

Diante dessas limitações, este estudo investiga de que forma os componentes do PC e das SS podem apoiar o ensino de programação na EPTNM. Assim, formulou-se a seguinte questão de pesquisa: **Como integrar os componentes do Pensamento Computacional e as *soft skills* no ensino de programação na educação profissional técnica de nível médio?**

1.2 Objetivos

O objetivo geral deste estudo é analisar a integração entre os componentes do PC e as competências de SS no ensino de programação na educação técnica de nível médio, a fim de propor um *framework* conceitual-operacional[1] com foco na mediação docente, voltado à resolução de problemas, e examinar suas contribuições. Para alcançar esse objetivo, foram definidos os seguintes Objetivos Específicos (OE):

- OE1: identificar na literatura científica as estratégias, lacunas e conceitos referentes à aplicação integrada do PC e das SS no ensino de programação.
- OE2: investigar as percepções e práticas de professores de programação dos Institutos Federais quanto ao uso do PC e das SS em suas disciplinas.
- OE3: propor e estruturar o *framework* com base no referencial teórico, no mapeamento da literatura e nas percepções de professores de programação.
- OE4: implementar o *framework* em um ambiente real de sala de aula de Lógica de Programação, junto aos docentes de um Instituto Federal.

[1]Nesta tese, *framework* é compreendido como uma estrutura conceitual-operacional que organiza fundamentos teóricos e diretrizes práticas para apoiar a atuação do professor, caracterizado em detalhe na Seção 1.3.

- OE5: analisar as percepções de professores e estudantes sobre as contribuições do *framework* SoPensa para o ensino e a aprendizagem de programação.

1.3 Metodologia

Este estudo caracteriza-se como uma pesquisa aplicada, de natureza descritiva, com características exploratórias (Mattar; Ramos, 2021), com o objetivo de propor um *framework* conceitual-operacional para o desenvolvimento de componentes do PC e das competências de SS no ensino de programação. A investigação foi conduzida no contexto da EPTNM em computação, adotando-se uma abordagem predominantemente qualitativa, com uso complementar de dados quantitativos descritivos, o que possibilitou ampliar a compreensão do objeto investigado.

Quanto aos procedimentos técnicos, trata-se de uma pesquisa-ação caracterizada pela participação ativa dos envolvidos, pela flexibilidade das etapas e pelo seu caráter participativo e reflexivo (Thiollent, 2022; Tripp, 2005). Nessa perspectiva, a pesquisa-ação é compreendida como um processo que integra conhecimento e ação, no qual a intervenção no contexto investigado contribui para a produção de novos conhecimentos e, ao mesmo tempo, permite compreender os problemas observados (Thiollent, 2022).

A pesquisa-ação mostrou-se adequada por permitir analisar o fenômeno a partir das práticas docentes e das percepções dos participantes. Esse tipo de abordagem caracteriza-se por um movimento cíclico, no qual cada etapa fornece subsídios para ajustes nas fases subsequentes. O pesquisador atuou de forma participante, em colaboração com os professores responsáveis pela regência das aulas. Sua atuação concentrou-se no apoio técnico e operacional e no registro do processo, sem assumir a condução das atividades em sala, em consonância com o caráter colaborativo da pesquisa-ação (Thiollent, 2022; Tripp, 2005).

A pesquisa foi submetida à apreciação do Comitê de Ética em Pesquisa do Instituto Federal do Acre, sob o CAAE 85642524.5.0000.0233, em conformidade com os procedimentos éticos aplicáveis à pesquisa envolvendo seres humanos. Os termos de consentimento e de assentimento utilizados constam do Apêndice I.

A coleta de dados foi realizada por meio de questionários e entrevistas, elaborados conforme a etapa da pesquisa e o perfil dos participantes, e voltados ao diagnóstico, ao acompanhamento da aplicação e à validação do *framework*. A descrição dos instrumentos e os respectivos protocolos encontram-se nos Apêndices C, E, G e H.

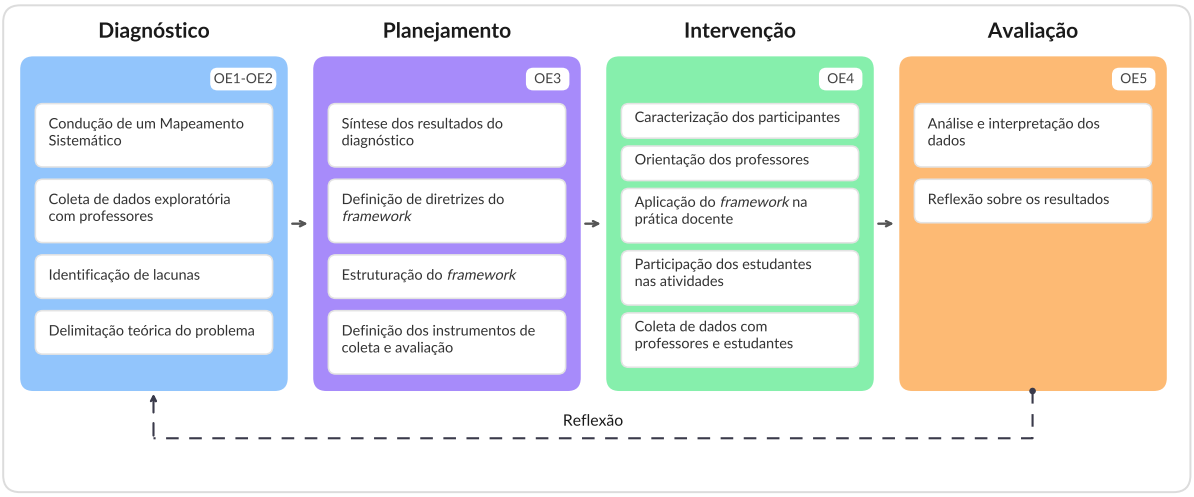
Durante o desenvolvimento desta pesquisa, ferramentas de inteligência artificial generativa foram empregadas como recursos auxiliares na revisão e no aprimoramento textual, na organização de conteúdos e na elaboração e aperfeiçoamento de tabelas, gráficos e figuras. Seu uso teve caráter exclusivamente auxiliar, permanecendo sob responsabilidade do pesquisador a avaliação, seleção e validação dos conteúdos incorporados à tese.

A estratégia metodológica adotada manteve alinhamento com a pergunta de pesquisa, uma vez que o estudo propôs uma estrutura conceitual-operacional para o ensino de programação e analisou sua aplicação em contexto real. A abordagem qualitativa predominante foi adotada para compreender, de forma interpretativa, essas percepções, conforme a natureza desse tipo de investigação (Mattar; Ramos, 2021). Os dados quantitativos foram analisados por meio de estatística descritiva, com uso de frequências absolutas e relativas, sendo utilizados como complemento à análise qualitativa (Creswell; Creswell, 2017).

Cabe destacar que, embora a pesquisa-ação seja caracterizada por ciclos iterativos de ação e reflexão, este estudo realizou um único ciclo. Assim, não foi conduzida uma nova intervenção a partir dos resultados da avaliação. Os achados obtidos foram utilizados para analisar a aplicabilidade do *framework*, identificar limitações e subsidiar possibilidades de melhorias em estudos posteriores.

O desenho metodológico foi organizado nas fases de Diagnóstico, Planejamento, Intervenção e Avaliação. A reflexão ocorreu ao longo de todo o processo, visto que não se configura como uma etapa isolada, mas sim como um componente contínuo do ciclo investigativo (Tripp, 2005). A Figura 1 apresenta a organização dessas fases e os principais procedimentos realizados no ciclo da pesquisa, em alinhamento com os objetivos propostos.

Figura 1: Ciclo metodológico da pesquisa-ação adotado no estudo



Fonte: Autoria própria.

Inicialmente, na fase de **Diagnóstico**, investigou-se o problema de pesquisa e identificaram-se lacunas relacionadas ao desenvolvimento dos componentes do PC e das SS no ensino de programação. Para isso, foram consideradas evidências provenientes do referencial teórico e da literatura científica, incluindo a realização de um Mapeamento Sistemático da Literatura (MSL) (Petersen *et al.*, 2008). Além disso, foram coletados dados junto aos professores de computação dos Institutos Federais (IFs) do Brasil, por meio de um questionário, seguindo as diretrizes de Kitchenham e Pfleeger (2008), e entrevistas com docentes, conforme orienta Mattar e Ramos (2021). Os dados do diagnóstico foram tratados de forma descritiva: o questionário por estatística descritiva e as respostas abertas e entrevistas por categorização direta com categorias definidas *a priori.*

Na fase de **Planejamento**, os resultados obtidos na etapa de diagnóstico foram sintetizados para orientar as decisões que estruturaram a intervenção. Nesse processo, foram considerados os dados provenientes do referencial teórico, do MSL e da coleta exploratória com professores, permitindo identificar as necessidades relacionadas ao desenvolvimento do PC e das SS no ensino de programação.

A partir dessa compreensão, foram definidas as diretrizes do *framework* denominado SoPensa, considerando seus aspectos conceituais e operacionais. No nível conceitual, entende-se que um *framework* estrutura conceitos e suas relações a partir do problema investigado, combinando elementos teóricos e empíricos (Rocco; Plakhotnik, 2009; Brennan; Resnick *et al.*, 2012).

Em nível operacional, tomou-se como referência a aplicação em contextos educacionais, com base na estrutura de conceitos e práticas do PC (Santana; Chavez; Bittencourt,

2021; Brennan; Resnick *et al.*, 2012). O *framework* conceitual-operacional é compreendido, neste estudo, como uma estrutura que integra fundamentos teóricos e evidências empíricas à prática de desenvolvimento dos componentes do PC e das SS, orientando sua aplicação no ensino de programação. Ao final dessa etapa, definiram-se os instrumentos de coleta e avaliação destinados a acompanhar a aplicação do *framework*. Tais recursos permitiram analisar as percepções de docentes e discentes, além dos registros produzidos durante o processo.

Na fase de **Intervenção**, o *framework* foi utilizado em duas turmas de Lógica de Programação, com o objetivo de examinar suas contribuições para o ensino dos componentes do PC e das SS. A etapa iniciou-se com a seleção e a orientação dos professores participantes, seguida de sua caracterização e da condução das atividades em sala. Os professores conduziram o trabalho em suas turmas, enquanto o pesquisador acompanhou o processo de forma observacional, registrando evidências e oferecendo suporte pontual quando necessário. Os estudantes participaram das atividades propostas, integrando o contexto investigado e a produção dos dados da pesquisa. Concluída a intervenção, foram realizadas entrevistas semiestruturadas com os professores, a fim de explorar suas percepções sobre o uso do *framework* e sobre o desenvolvimento dos componentes do PC e das SS pelos estudantes. Aos alunos, aplicou-se um questionário sobre as atividades realizadas com o SoPensa.

Por fim, na fase de **Avaliação**, as evidências produzidas durante a intervenção foram analisadas e interpretadas, considerando as percepções dos professores e estudantes e os dados qualitativos e quantitativos descritivos obtidos por meio dos instrumentos utilizados. Essa etapa permitiu refletir sobre os resultados da aplicação e identificar aspectos que poderiam ser mantidos ou aprimorados.

Para o tratamento dos dados qualitativos dessa fase de avaliação, adotou-se a Análise Temática Reflexiva (ATR) sugerida por Braun e Clarke (2022). A sistematização dos dados foi apoiada pelo uso do *software* Atlas.ti, seguindo as orientações de Friese (2019), e organizada com base em quadros analíticos inspirados em Huberman *et al.* (2014). Na análise e interpretação dos dados, foram considerados conjuntamente os diferentes registros produzidos durante a aplicação, incluindo as entrevistas com os professores, o questionário dos estudantes e os instrumentos de acompanhamento, conforme a perspectiva de triangulação discutida por Flick (2013).

1.4 Organização da tese

Além deste capítulo introdutório, a tese está organizada em oito capítulos, descritos a seguir.

- **Capítulo 2:** apresenta os fundamentos teóricos que sustentam a pesquisa, abordando o ensino e a aprendizagem de programação, os desafios enfrentados por professores e estudantes, as metodologias ativas empregadas nesse contexto e a relevância do desenvolvimento das habilidades técnicas e das SS na área. São discutidos os conceitos de PC e SS, sua integração no ensino de programação e as teorias de aprendizagem que dão suporte à proposta, como o Construcionismo e a Aprendizagem Significativa. O capítulo apresenta ainda o contexto da EPTNM e a organização da Rede Federal, no qual a pesquisa se insere.

- **Capítulo 3:** apresenta o estado da arte por meio do MSL, detalhando suas etapas e critérios. São analisadas as abordagens identificadas na literatura relacionadas ao desenvolvimento do PC e das SS no ensino de programação, com destaque para lacunas que evidenciam a necessidade de propostas mais estruturadas para esse contexto.

- **Capítulo 4:** apresenta o estudo empírico realizado com professores dos Institutos Federais, com o objetivo de investigar suas percepções sobre o desenvolvimento dos componentes do PC e das SS no ensino de programação. São descritos os participantes, os instrumentos de coleta de dados, compostos por questionários e entrevistas, e os procedimentos de análise, cujos resultados contribuíram para a fundamentação da construção do *framework* SoPensa.

- **Capítulo 5:** apresenta o *framework* SoPensa, desenvolvido com base nas evidências teóricas e empíricas obtidas nas etapas anteriores. São descritas sua estrutura conceitual-operacional, suas fases e componentes, bem como as diretrizes que orientam sua aplicação no ensino de programação no contexto da EPTNM.

- **Capítulo 6:** apresenta a aplicação do *framework* SoPensa em contexto real de ensino. Inicia pela apreciação por especialistas e pela caracterização do ambiente, dos professores e das turmas, e segue com o relato da aplicação nas quatro fases, do diagnóstico do perfil da turma à consolidação das produções, incluindo a formação dos times, as atividades desplugadas e plugadas e o acompanhamento de cada fase.

- **Capítulo 7:** apresenta as percepções de professores e estudantes sobre o *framework* SoPensa. As percepções dos professores são analisadas por meio da Análise Temática Reflexiva, organizada em cinco temas e um eixo complementar sobre o potencial pedagógico, e as dos estudantes tratam da aceitação da proposta.

- **Capítulo 8:** discute os resultados em diálogo com a literatura sobre ensino de programação, PC e SS. São tratados os achados que atravessam o estudo: a abstração como dificuldade persistente, o reconhecimento maior dos componentes do PC do que das competências de SS, a progressão das atividades desplugadas para as plugadas, a colaboração e a comunicação como competências que demandam mediação intencional e os condicionantes da educação profissional técnica sobre a prática docente.

- **Capítulo 9:** apresenta as considerações finais, com a retomada do problema e dos objetivos, as contribuições teóricas e práticas do estudo, suas limitações, as possibilidades de trabalhos futuros e as publicações derivadas da tese.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta o referencial teórico que fundamenta a tese, abordando o ensino e a aprendizagem de programação e as dificuldades enfrentadas por professores e estudantes nesse contexto. O ensino de programação é compreendido como um processo que envolve o desenvolvimento de habilidades técnicas e SS, o que reforça a complexidade dessa aprendizagem.

O capítulo está estruturado em seis partes. Inicialmente, são discutidos o ensino e a aprendizagem de programação, suas dificuldades e as metodologias ativas adotadas nesse contexto. Em seguida, são apresentadas as teorias de aprendizagem que sustentam esta pesquisa, com destaque para a Aprendizagem Significativa e o Construcionismo. A terceira parte aborda o PC, considerando seus componentes e sua relação com a aprendizagem e a computação desplugada como estratégia para seu desenvolvimento. A quarta parte discute as SS e sua relevância para a formação dos estudantes. A quinta parte trata da integração entre o PC e as SS no ensino de programação, eixo conceitual desta tese. Por fim, é apresentado o contexto da EPTNM, no qual a investigação está inserida.

2.1 Ensino e aprendizagem de programação

De 2020 a 2024, as matrículas em cursos técnicos de Informática no Brasil passaram de 80.613 para 89.168, um crescimento de aproximadamente 10,6%, abrangendo variadas formações em programação (Brasil, 2024b). Esse aumento reflete a crescente demanda por profissionais de tecnologia, inclusive em áreas que antes não exigiam tais conhecimentos, uma vez que habilidades como a programação deixaram de ser restritas a um campo específico, tornando-se necessárias em diversas áreas profissionais (Ribeiro *et al.*, 2020). Estudos sobre o ensino e a aprendizagem de programação têm destacado a relação desse processo com o desenvolvimento de capacidades voltadas à compreensão e à resolução de problemas (Medeiros; Falcão; Ramalho, 2021; Hudin, 2023; Figueiredo; García-Peñalvo, 2022; Sim; Lau, 2022).

Compreende-se que aprender programação vai além do domínio de comandos ou sintaxes específicas. Ribeiro *et al.* (2020) destacam que esse aprendizado ultrapassa a manipulação de estruturas de dados, exigindo a capacidade de abstrair soluções e interpretar problemas de forma lógica. Esse processo envolve tanto o embasamento teórico quanto a aquisição de habilidades práticas, como destacam (López-Pernas; Saqr; Viberg, 2021). A associação entre tais elementos contribui para a construção de conhecimento aplicado e amplia as possibilidades de utilização da programação em diferentes contextos.

Para Apeanti e Essel (2021), a programação é uma atividade prática orientada ao desenvolvimento de habilidades cognitivas, como a resolução de problemas. O ensino de programação tem, então, como objetivo desenvolver conhecimentos e habilidades que possibilitem ao estudante projetar programas capazes de resolver problemas reais (Figueiredo; García-Peñalvo, 2022). Nesse sentido, a resolução de problemas se configura como um elemento estruturante da aprendizagem, sendo tratada como uma das bases fundamentais nesse processo (Medeiros; Falcão; Ramalho, 2021).

Na aprendizagem de programação, programar e resolver problemas exige a mobilização de diversas habilidades cognitivas. Ribeiro *et al.* (2020) destacam que essa prática envolve diferentes operações mentais, que demandam organização, interpretação e elaboração de soluções. À medida que o estudante analisa enunciados, estrutura etapas e constrói respostas para os problemas propostos, desenvolve formas de pensamento que sustentam a aprendizagem da programação. Essas dinâmicas se ampliam quando são mobilizados componentes do PC, como decomposição e abstração, ao lado de operações como análise e avaliação de problemas (Macrides; Miliou; Angeli, 2022). Nesse contexto, a prática da programação contribui para o desenvolvimento de formas de pensar que apoiam a resolução estruturada de problemas.

No percurso formativo, Hudin (2023) defende que a programação deve ser iniciada desde os primeiros anos de educação, favorecendo o desenvolvimento de habilidades como pensamento crítico, imaginação criativa e resolução de problemas. Ribeiro, Martins e Berssanette (2022) indicam ainda que aprender a programar pode acarretar benefícios em diferentes áreas, considerando a presença da computação em diversas profissões, o que reforça sua relevância na formação contemporânea.

2.1.1 Dificuldades no ensino e na aprendizagem de programação

Apesar dos benefícios, a aprendizagem de programação envolve dificuldades na compreensão dos conteúdos e na autorregulação da aprendizagem, especialmente em aspectos

cognitivos e metacognitivos, o que torna o processo mais desafiador (Hudin, 2023; Ribeiro; Martins; Berssanette, 2022). Em disciplinas introdutórias, essa área costuma ser percebida por estudantes e professores como um campo complexo (Sim; Lau, 2022; Figueiredo; García-Peñalvo, 2024). Essa percepção está relacionada ao fato de que a programação exige esforço contínuo, persistência e o desenvolvimento integrado de múltiplas habilidades (Cheah, 2020; Figueiredo; García-Peñalvo, 2022).

Entre as dificuldades mais citadas, destacam-se limitações relacionadas à abstração e ao raciocínio lógico, que impactam diretamente a capacidade de compreender problemas e estruturar soluções de forma coerente (Silva; Moreira, 2021; Ribeiro *et al.*, 2020; Medeiros; Falcão; Ramalho, 2021). Somam-se a isso dificuldades para decompor problemas em partes menores e para organizar funcionalidades em procedimentos, o que compromete a elaboração de arquiteturas de *software* mais robustas (Cheah, 2020; Yong; Tiong, 2022).

A literatura aponta, ainda, por parte dos estudantes, barreiras na identificação dos elementos importantes dos enunciados e limitações na aplicação de conceitos básicos na construção de soluções (Medeiros; Falcão; Ramalho, 2021; Ribeiro *et al.*, 2020). Para Medeiros, Falcão e Ramalho (2021) e Qian e Lehman (2017), essas dificuldades estão associadas à ausência de conhecimentos prévios, como lógica e fundamentos matemáticos, além de limitações na interpretação dos problemas (Cheah, 2020; Morais; Mendes Neto; Osório, 2020). Como consequência, a aprendizagem tende a ocorrer de forma fragmentada, dificultando a assimilação do conhecimento ao longo do tempo.

Além dos aspectos cognitivos, fatores comportamentais influenciam diretamente o aprendizado. Observam-se manifestações de desmotivação, frustração e até aversão à programação ao longo da aprendizagem (Morais; Mendes Neto; Osório, 2020; Ribeiro *et al.*, 2020; Queiroz; Rodrigues; Coutinho, 2018). Erros contínuos e a sensação de não conseguir acompanhar o conteúdo contribuem para a perda de interesse e, em alguns casos, para a desistência (Ribeiro *et al.*, 2020; Cheah, 2020). Esse quadro também pode estar associado à ansiedade e a percepções negativas em relação aos conteúdos de programação (Chang; Lin; Lin, 2024).

Por outro lado, professores também enfrentam dificuldades relacionadas às suas competências pedagógicas, incluindo o conhecimento pedagógico, o desenvolvimento de habilidades e o uso de estratégias adequadas para o ensino de programação (Hudin, 2023). A escolha de métodos e ferramentas apropriadas nem sempre é simples, especialmente em níveis introdutórios, onde abordagens tradicionais mostram limitações (Medeiros; Falcão; Ramalho, 2021; Cheah, 2020).

Figueiredo e García-Peñalvo (2024) destacam que outro ponto relevante é a dificuldade em acompanhar o progresso dos estudantes, principalmente quando há barreiras na comunicação ou quando os alunos têm dificuldade em expressar suas dúvidas. A heterogeneidade das turmas e as condições institucionais ampliam esses desafios (Medeiros; Falcão; Ramalho, 2020; Hudin, 2023). Diferenças no conhecimento prévio, turmas numerosas e limitações de infraestrutura comprometem o acompanhamento individualizado dos estudantes e a realização de atividades práticas (Morais; Mendes Neto; Osório, 2020; Hudin, 2023).

Mesmo com a disponibilidade de ferramentas tecnológicas, muitos problemas persistem, o que sugere que a presença de recursos, por si só, não resolve as dificuldades no ensino de programação (Cheah, 2020). Diante dessas dificuldades, podem ocorrer desmotivação, reprovação e evasão em disciplinas de programação (Santos; Santos, 2017; Morais; Mendes Neto; Osório, 2020; Ribeiro *et al.*, 2020).

2.1.2 Metodologias ativas no ensino de programação

As metodologias ativas têm sido adotadas no ensino de programação como forma de deslocar parte da centralidade da exposição docente para a participação do estudante. Calderon, Silva e Feitosa (2024) identificam 37 tipos de metodologias empregados em cursos de graduação e, no cenário brasileiro, os jogos educacionais e a gamificação aparecem como as opções mais recorrentes (Calderon; Silva; Feitosa, 2021). A maior parte dos docentes universitários brasileiros consultados declara utilizar ou já ter utilizado alguma dessas metodologias, e a gamificação reaparece entre as três mais adotadas (Calderon *et al.*, 2024).

Os arranjos identificados nem sempre se restringem a uma metodologia. Parte dos trabalhos analisados combina duas ou mais, em composições que reúnem Sala de Aula Invertida, Aprendizagem Baseada em Projetos, Aprendizagem Baseada em Problemas, Aprendizagem Baseada em Times, gamificação, programação em pares e estratégias investigativas (Calderon; Silva; Feitosa, 2024). O proveito desses arranjos está na complementaridade, uma vez que a Sala de Aula Invertida favorece a preparação prévia e a interação com o professor, enquanto a Aprendizagem Baseada em Problemas privilegia a centralidade do estudante e a aplicação do conhecimento.

As contribuições relatadas na literatura apresentam pontos de convergência. Berssanette e Francisco (2021) registram maior aceitação e retorno positivo dos estudantes, com efeitos sobre satisfação e motivação, além de melhorias na experiência de aprendizagem

e no desempenho. Percepção semelhante foi identificada entre os professores consultados por Calderon *et al.* (2024), que atribuem a essas metodologias ganhos de engajamento em sala. Os dois estudos apontam ainda o desenvolvimento de competências ligadas ao trabalho colaborativo e à comunicação, acrescentando-se o pensamento crítico, apontado no levantamento com docentes brasileiros.

Os benefícios dependem de condições de implementação. A dificuldade mais frequente entre os professores consultados é a ausência de formação docente específica, seguida da carência de suporte tecnológico e da restrição de tempo para o planejamento das aulas (Calderon *et al.*, 2024). As dificuldades relatadas indicam que a adoção dessas metodologias não elimina a necessidade de mediação docente nem dispensa as condições institucionais que a viabilizam.

As evidências reunidas por Berssanette e Francisco (2021) concentram-se no contexto universitário, e os autores recomendam verificar se as contribuições se reproduzem em outras etapas da formação. Na educação profissional, contexto ainda pouco explorado nos estudos sobre metodologias ativas no ensino de programação, Alves (2023) relata o caso de uma disciplina de Lógica de Programação de curso técnico em Informática, na qual reprovações e evasões atingiam a maior parte dos matriculados sob o modelo expositivo. A adoção da Aprendizagem Baseada em Problemas com Scratch e, depois, da Aprendizagem Baseada em Times com Sala de Aula Invertida reduziu as medianas de reprovação e ampliou a integração entre os estudantes, com aumento da carga de trabalho de docentes e estudantes como contrapartida.

Azevedo Dias e Souza Araújo (2024) relatam experiência em contexto próximo, em oficina sobre PC Paralelo com estudantes de cursos técnicos em Informática, na qual a Aprendizagem Baseada em Projetos e a gamificação foram associadas a indicadores de aceitabilidade, motivação e envolvimento durante as atividades. Os dois relatos indicam a viabilidade dessas abordagens em contextos de formação técnica, sobretudo quando as atividades envolvem projetos e situações práticas relacionadas à Computação.

2.2 Teorias de aprendizagem aplicadas ao ensino de programação

Considerando as demandas do ensino de programação, especialmente aquelas relacionadas a compreensão de conceitos, resolução de problemas e organização do conhecimento, torna-se necessário recorrer a fundamentos teóricos que orientem a prática pedagógica.

Nesse sentido, destacam-se a Aprendizagem Significativa (AS), de David Ausubel, e o Construcionismo, de Seymour Papert. A AS oferece uma base cognitiva para compreender como o conhecimento é estruturado e assimilado, enquanto o Construcionismo enfatiza a construção ativa do conhecimento por meio da criação e da experimentação.

2.2.1 Aprendizagem significativa

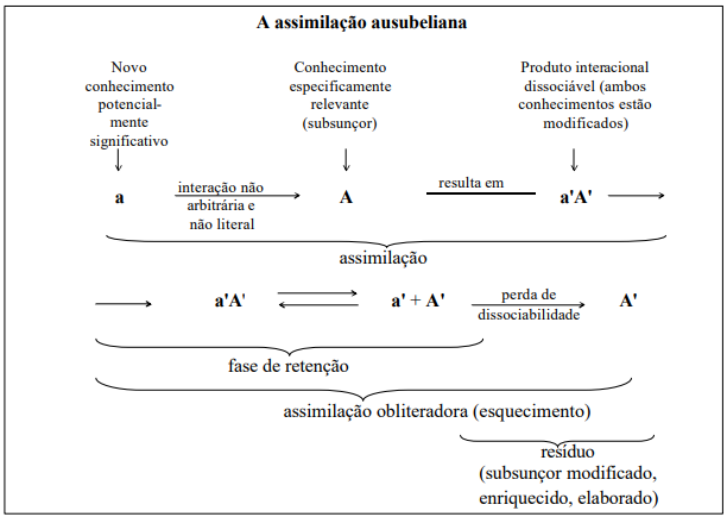
A AS, proposta por David Ausubel na década de 1960, destaca a importância do conhecimento prévio do aprendiz na aquisição de novos conhecimentos (Ausubel, 2003). Segundo Moreira (2012), novas informações são relacionadas ao conhecimento já existente, permitindo maior integração dos conceitos.

O processo ocorre por meio da interação entre novas informações e estruturas cognitivas pré-existentes, denominadas subsunçores, de forma não arbitrária e não literal (Moreira, 2012; Pelizzari *et al.*, 2002). Nessa perspectiva, o conhecimento é organizado em estruturas cognitivas nas quais os conceitos se relacionam de forma hierárquica, influenciando a aprendizagem de novos conteúdos (Moreira, 2012).

Para que a AS ocorra, duas condições devem ser atendidas: a utilização de materiais potencialmente significativos e a predisposição do estudante para aprender (Ausubel; Novak; Hanesian, 1980). Diferentemente da aprendizagem mecânica, baseada na memorização, a AS favorece maior retenção, pois os conceitos passam a integrar a estrutura cognitiva do indivíduo (Moreira, 2012; Pelizzari *et al.*, 2002). Além disso, trata-se de um processo ativo, no qual o estudante relaciona, reorganiza e integra o novo conhecimento à estrutura cognitiva existente (Moreira, 2012).

A Figura 2 apresenta a organização do processo de AS, destacando a relação entre o conhecimento prévio e a incorporação de novas informações. Os novos conceitos são integrados à estrutura cognitiva existente, favorecendo a construção de significados.

Figura 2: A assimilação ausubeliana



Fonte: Moreira (2006b).

A predisposição para a aprendizagem está relacionada ao interesse e ao envolvimento do estudante com o conteúdo (Novak; Rabaça; Valadares, 2000; Moreira, 2012). Para Pelizzari *et al.* (2002), a AS exige participação ativa, evitando a simples reprodução de conceitos. Moreira (2006b) destaca que o material educativo deve ser relevante e organizado de modo a permitir sua integração ao conhecimento prévio. Além dos aspectos cognitivos, esse processo também envolve dimensões afetivas (Novak; Gowin, 1996).

A diferenciação progressiva refere-se ao aprofundamento gradual dos conceitos, enquanto a reconciliação integrativa relaciona novos e antigos conhecimentos (Moreira, 2006b), processo que ocorre de forma contínua ao longo da aprendizagem (Moreira, 2012). Assim, a AS não se estabelece de forma imediata, mas por meio de reorganizações sucessivas da estrutura cognitiva.

No ensino de programação, a AS contribui para a organização do conteúdo de forma progressiva, favorecendo a compreensão de conceitos abstratos (Zanetti; Borges; Ricarte, 2022). Um dos desafios está na ausência de conhecimentos prévios adequados, o que dificulta a construção de novas relações cognitivas (Berssanette; Francisco, 2018). Esse cenário é recorrente em disciplinas introdutórias, nas quais muitos estudantes apresentam dificuldades para relacionar conhecimentos anteriores com novos conceitos (Zanetti; Borges; Ricarte, 2022). Para minimizar tais dificuldades, torna-se necessário considerar o repertório dos estudantes e propor atividades que favoreçam a conexão entre conceitos (Astolfi; Junior, 2016).

2.2.2 Teoria construcionista

O Construcionismo, proposto por Seymour Papert, enfatiza a aprendizagem por meio da criação. Influenciado pelo construtivismo de Piaget, essa abordagem considera que o conhecimento é construído a partir da interação com o meio (Papert, 1980). E a aprendizagem ocorre de forma mais efetiva quando o estudante está envolvido na construção de artefatos significativos, favorecendo a organização do pensamento e a construção do conhecimento (Papert, 1980; Resnick *et al.*, 2009). No contexto educacional, essa abordagem se traduz em práticas baseadas na criação, nas quais o estudante assume papel ativo no processo de aprendizagem (Eggert; Asquino; Cruz, 2023). Assim, ao desenvolver projetos, há mobilização de competências relacionadas ao contexto educacional e profissional (Avila; Costa Cavalheiro, 2022), sendo a criação de artefatos com significado pessoal um dos princípios dessa abordagem (Resnick *et al.*, 2009).

A tecnologia também ocupa papel relevante nesse processo, pois o computador atua como suporte para a construção do conhecimento, permitindo explorar conceitos de forma concreta (Papert, 1980). Nesse sentido, ferramentas como o Scratch exemplificam esse potencial ao possibilitar a criação de programas de forma visual e interativa (Resnick *et al.*, 2009).

No construcionismo, o erro não é tratado como falha, mas como parte do processo de construção do conhecimento. Segundo Papert (1980), ao testar diferentes soluções e revisar suas produções, o estudante compreende melhor os conceitos envolvidos e ajusta seu raciocínio. A experimentação de alternativas favorece uma aprendizagem exploratória, na qual diferentes possibilidades são analisadas, contribuindo para a consolidação do conhecimento por meio da prática (Valente, 1993).

Outro aspecto relevante está relacionado à externalização do pensamento durante a construção de artefatos. Ao desenvolver programas ou objetos, o estudante torna visíveis suas ideias, permitindo analisar, testar e modificar suas próprias estratégias. Como discutido por Resnick *et al.* (2009), esse movimento contribui para a organização do pensamento e para a compreensão de conceitos, uma vez que o conhecimento passa a ser representado de forma concreta.

Além disso, o construcionismo destaca a importância de ambientes de aprendizagem que favoreçam a criação, a experimentação e a interação com diferentes recursos. Nesses ambientes, o estudante tem maior liberdade para explorar soluções e desenvolver projetos, o que contribui para uma aprendizagem mais ativa e contextualizada. O uso de tecnolo-

gias digitais amplia essas possibilidades ao oferecer ferramentas que permitem criar, testar e compartilhar produções, fortalecendo a construção do conhecimento (Papert, 1980; Resnick *et al.*, 2009).

No ensino de programação, o Construcionismo vai além do uso de sintaxe e comandos, pois favorece a construção de soluções por meio da criação (Valente, 1993; Papert, 1980). As tentativas e revisões assumem papel formativo, sendo parte da aprendizagem (Papert, 1980). A exploração de ferramentas como o Scratch permite iniciar o contato com a programação de forma mais intuitiva e apoiar a construção de programas em contextos de criação (Resnick *et al.*, 2009). Nesse processo, o professor atua como mediador, orientando a aprendizagem e apoiando o desenvolvimento dos estudantes (Eggert; Asquino; Cruz, 2023). Atuação que favorece a construção do conhecimento de forma mais autônoma.

2.2.3 Implicações das teorias no ensino de programação

A AS e o Construcionismo podem ser compreendidos como abordagens complementares no ensino de programação. A AS enfatiza a relação entre novos conhecimentos e o repertório prévio (Zanetti; Borges; Ricarte, 2022; Astolfi; Junior, 2016), enquanto o Construcionismo destaca a aplicação prática por meio da criação e da resolução de problemas (Resnick *et al.*, 2009). Assim, a aprendizagem em programação envolve tanto a construção de significados quanto sua aplicação em contextos práticos. Sob esse ponto de vista, o conhecimento não é apenas assimilado, mas também utilizado na elaboração de soluções, aproximando teoria e prática no processo de aprendizagem.

A construção de soluções por meio da criação e da experimentação contribui para a compreensão de conceitos e para a aplicação do conhecimento em diferentes contextos (Resnick *et al.*, 2009; Papert, 1980; Zanetti; Borges; Ricarte, 2022). Esse movimento reforça a importância de integrar abordagens que considerem, ao mesmo tempo, a organização cognitiva do conhecimento e sua aplicação prática no ensino de programação.

2.3 Pensamento Computacional

Em 2006, o conceito de PC ganhou destaque com a publicação do artigo *Computational Thinking*, de Jeannette Wing, que o apresentou como uma habilidade fundamental para a resolução de problemas com base em conceitos da Ciência da Computação (Wing, 2006). Contudo, por associar o PC à resolução de problemas de forma ampla, essa formulação não permitia distingui-lo de outras formas de raciocínio. A própria autora refinou o conceito

ao descrever o PC como a formulação de problemas e a construção de soluções executáveis por humanos ou máquinas (Wing, 2011), destacando o papel central da abstração (Wing, 2014).

A literatura sobre PC adota terminologias variadas para descrever sua estrutura interna. Wing (2006) e Wing (2011) refere-se ao PC como um modo de pensar e a uma habilidade fundamental. Brennan, Resnick *et al.* (2012) o organizam em conceitos, práticas e perspectivas computacionais. Barr e Stephenson (2011) o descrevem por conceitos centrais e capacidades.

Diante dessa diversidade conceitual, esta tese adota a seguinte terminologia: o PC é compreendido como uma capacidade cognitiva ampla voltada à formulação e à resolução estruturada de problemas, mobilizada por meio de quatro componentes, a saber, decomposição, reconhecimento de padrões, abstração e algoritmos. O termo "componentes" é adotado por sua presença consistente na literatura científica internacional (Grover; Pea, 2013; Shute; Sun; Asbell-Clarke, 2017). E, embora parte da literatura empregue "pilares" como sinônimo (Wu *et al.*, 2024; BBC, 2024), neste trabalho reserva-se "pilar" para as bases do *framework* SoPensa, tratando-se os elementos internos do PC como componentes e os das SS como competências.

Tal processo inclui a organização de dados, a representação de informações por meio de abstrações e a estruturação de soluções com base em sequências lógicas de passos (Barr; Harrison; Conery, 2011). Nessa perspectiva, o PC pode ser compreendido como uma capacidade cognitiva voltada à organização e à resolução estruturada de problemas.

Nesta tese, a definição operacional adotada apoia-se na caracterização do PC por meio de seus componentes (decomposição, reconhecimento de padrões, abstração e algoritmos), que permitem identificar práticas específicas mobilizadas na análise e na construção de soluções computacionais (Wu *et al.*, 2024; Grover; Pea, 2013; Román-González; Pérez-González; Jiménez-Fernández, 2017), o que encontra respaldo em Wing (2006), ao defender que essa habilidade deve ocupar um papel semelhante ao da leitura, escrita e aritmética. Na mesma direção, Chang, Lin e Lin (2024) descrevem o PC como uma forma analítica de pensamento, próxima do raciocínio utilizado na engenharia, especialmente ao lidar com sistemas complexos e restrições do mundo real. Essa aproximação com outras áreas do conhecimento tem levado pesquisadores a discutir o potencial do PC para além do campo específico da computação.

Montuori *et al.* (2024), em revisão sistemática e meta-analítica de estudos experimentais publicados entre 2006 e 2022, investigaram os efeitos de intervenções baseadas

em programação e codificação sobre funções executivas de crianças e adolescentes. Os resultados apontam efeitos positivos em habilidades relacionadas à resolução de problemas, ao planejamento, à inibição de respostas e à memória de trabalho. E os achados oferecem evidências empíricas de que intervenções envolvendo PC podem favorecer o desenvolvimento de habilidades cognitivas associadas ao raciocínio estruturado. Os autores ressaltam, contudo, que os efeitos variam conforme o grau de estruturação da intervenção e a faixa etária dos participantes, indicando cautela na generalização dos resultados para contextos distintos dos investigados.

Nessa perspectiva, o PC pode ser compreendido como uma abordagem que organiza o raciocínio por meio de etapas estruturadas, envolvendo a definição de problemas, a elaboração de soluções e a verificação de resultados (Chang; Lin; Lin, 2024; Barr; Harrison; Conery, 2011). Embora a literatura aponte benefícios cognitivos associados ao desenvolvimento do PC, a discussão sobre sua transferência para domínios distintos da computação permanece em aberto, com resultados que variam conforme o tipo de intervenção e as funções cognitivas avaliadas (Montuori *et al.*, 2024).

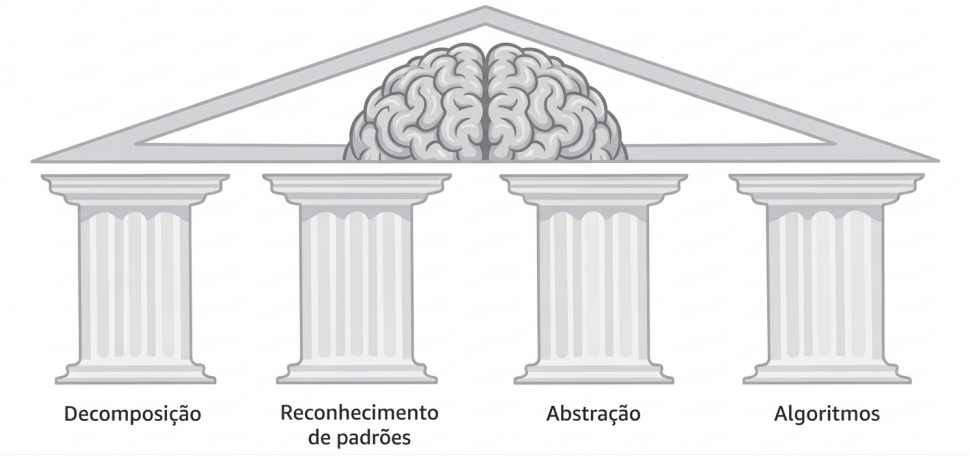
Nesta tese, o PC é compreendido como uma capacidade cognitiva que pode apoiar o processo de ensino e aprendizagem de programação, contexto em que seus componentes são mobilizados diretamente na construção de soluções computacionais.

2.3.1 Componentes do Pensamento Computacional

Embora não haja consenso sobre os fundamentos que sustentam o PC, autores como Wu *et al.* (2024), BBC (2024) e Farias e Barone (2023) apontam que o PC pode ser organizado em quatro componentes: decomposição, reconhecimento de padrões, abstração e algoritmos. Outros autores incluem componentes adicionais, como *debugging* e automação, o que indica variações na delimitação do conceito (Grover; Pea, 2013; Barr; Stephenson, 2011). Apesar das variações conceituais entre os autores, observa-se convergência na adoção desses quatro componentes como base estruturante do PC.

A Figura 3 apresenta os quatro principais componentes do PC discutidos pelos autores mencionados. Na sequência, são apresentadas as definições detalhadas de cada um deles.

Figura 3: Os quatro componentes do Pensamento Computacional



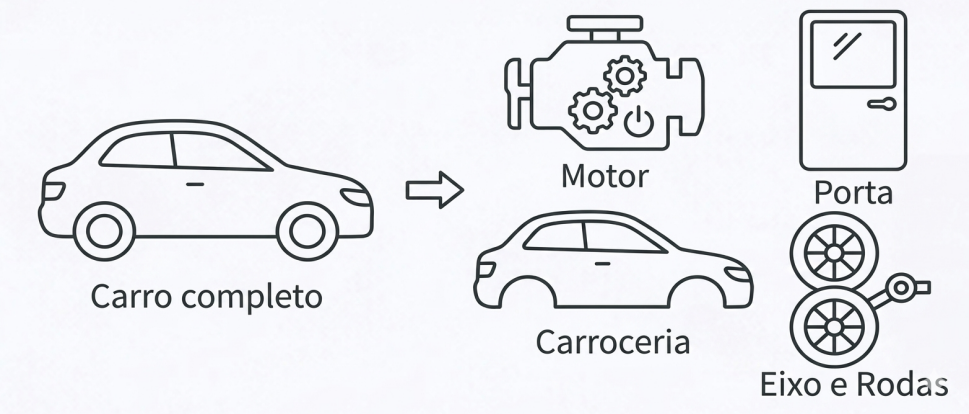
Fonte: Autoria própria

A **Decomposição** consiste em dividir um problema complexo em partes menores, tornando-o mais compreensível e viável de ser resolvido (Barr; Harrison; Conery, 2011; Fang *et al.*, 2022). Essa divisão reduz a complexidade inicial e permite que a solução seja construída de forma gradual (Wu *et al.*, 2024).

Wing (2006) associa diretamente o uso da decomposição à resolução de problemas complexos, ao destacar sua aplicação na análise de tarefas extensas e desafiadoras. Complementando essa perspectiva, Farias e Barone (2023) argumentam que, após a formulação de um problema, a divisão em pequenos problemas torna a resolução mais acessível e prática. Para os autores, a decomposição atua como um mecanismo inicial de organização cognitiva, permitindo estruturar problemas de forma progressiva.

A Figura 4 apresenta um exemplo de decomposição a partir da divisão de um sistema em partes menores. Observa-se um carro completo, que posteriormente é organizado em componentes como motor, porta, carroceria e eixo com rodas. Essa representação permite compreender como um problema pode ser analisado por partes, favorecendo o entendimento da solução como um todo ao evidenciar a relação entre as partes e o sistema completo.

Figura 4: Decomposição: divisão de um problema em partes menores



Fonte: Autoria própria.

O **Reconhecimento de Padrões** consiste em identificar similaridades entre diferentes dados ou situações, permitindo a aplicação de soluções previamente conhecidas a novos problemas. Para Farias e Barone (2023), esse processo é importante para o desenvolvimento de soluções inovadoras, pois possibilita a identificação de padrões comuns em diferentes contextos. Wu *et al.* (2024) destacam que o reconhecimento de padrões envolve identificar semelhanças, repetições ou características específicas em um problema, o que contribui para a construção de algoritmos eficientes (Kim; Leftwich; Castner, 2024).

O reconhecimento de padrões pode ocorrer tanto no problema principal quanto em seus subproblemas, permitindo reconhecer regularidades que podem ser reutilizadas em diferentes situações (Wu *et al.*, 2024). Além disso, está associado à possibilidade de reaproveitar estratégias previamente utilizadas em outros problemas (Barr; Harrison; Conery, 2011), visto que, ao identificar padrões, os estudantes desenvolvem a capacidade de resolver problemas de forma mais eficaz, pois esse processo envolve a generalização de soluções a partir de casos específicos (Wing, 2014). Além disso, organizar objetos e informações com base em características comuns contribui para uma compreensão mais estruturada (Kim; Leftwich; Castner, 2024).

A Figura 5 apresenta o reconhecimento de padrões por meio da identificação de características comuns entre diferentes elementos. Observa-se um objeto base, o veículo, cuja estrutura serve como referência. Esse padrão permite reconhecer variações que mantêm a mesma forma, mesmo com diferenças em atributos secundários. Nesse caso, os padrões são utilizados na interpretação de novos exemplos, demonstrando a generalização a partir de casos específicos.

Figura 5: Reconhecimento de padrões: identificação do padrão e de suas variações

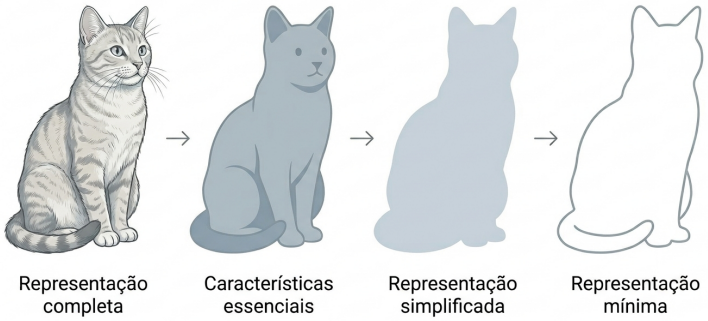
Fonte: Autoria própria.

A **Abstração** é um dos componentes principais do PC. Essa habilidade possibilita a criação de representações simplificadas, mantendo apenas os elementos relevantes para a resolução do problema(Wing, 2014; Barr; Harrison; Conery, 2011). Ao ignorar características menos importantes e focar nos elementos centrais, a abstração possibilita a criação de representações simplificadas de problemas (BBC, 2024).

Wing (2006) destaca que pensar como cientista da computação exige operar em diferentes níveis de abstração. Em trabalhos posteriores, a autora enfatiza que a abstração ocupa um papel central no PC, pois permite identificar propriedades relevantes ao mesmo tempo em que oculta elementos irrelevantes (Wing, 2014). Barr e Stephenson (2011) seguem essa ideia ao definir abstração como a simplificação do concreto para o geral.

A Figura 6 apresenta o conceito de abstração mediante a redução progressiva de um gato, partindo de sua representação detalhada até uma forma minimalista que preserva apenas os elementos fundamentais do objeto. Desse modo, a abstração viabiliza a eliminação de detalhes secundários em favor dos aspectos centrais, o que favorece a construção de representações simplificadas e eficazes para a resolução de problemas, alinhadas ao nível de detalhamento necessário para cada contexto.

Figura 6: Abstração: redução de detalhes, mantendo características essenciais



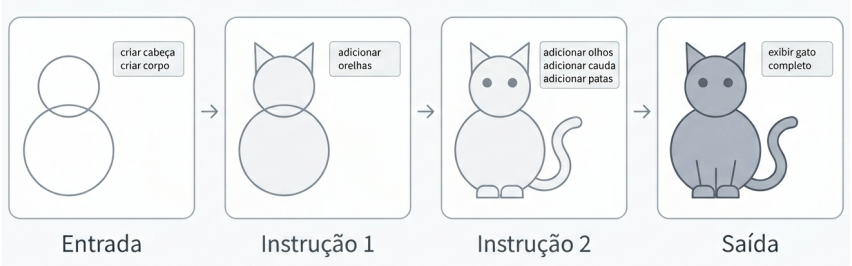
Fonte: Autoria própria.

O **Algoritmo** corresponde à definição de sequências ordenadas de passos para resolver um problema (Wing, 2011; Barr; Stephenson, 2011). A solução é organizada de forma lógica e estruturada, permitindo sua execução e replicação. Sim e Lau (2022) e Yong e Tiong (2022) associam o PC à automação de soluções por meio de sequências de passos, enquanto Wing (2011) descreve o algoritmo como uma abstração que expressa uma sequência de instruções para execução de tarefas. Além disso, os algoritmos permitem estruturar o raciocínio de forma sequencial e lógica, contribuindo para a organização da solução e para a resolução sistemática de problemas (Barr; Stephenson, 2011).

Algoritmos são utilizados em diferentes situações, incluindo cálculos, processamento de dados, automação e inteligência artificial (BBC, 2024). A ideia básica por trás do PC reside no pensamento algorítmico, que envolve a criação de soluções estruturadas para problemas complexos (Farias; Barone, 2023). A definição reforça a importância dos algoritmos como ferramentas que formalizam o processo de resolução de problemas, permitindo sua replicação e aprimoramento (Grover; Pea, 2013).

A Figura 7, apresenta um algoritmo por meio da execução de uma sequência ordenada de instruções. O fluxo começa com uma estrutura básica, seguida da adição progressiva de novos elementos até a obtenção do resultado final. A imagem contribui para representar visualmente a organização sequencial que caracteriza o pensamento algorítmico.

Figura 7: Algoritmo: sequência de instruções para resolver um problema



Fonte: Autoria própria

2.3.2 Pensamento Computacional como suporte à aprendizagem de programação

O PC também tem sido relacionado à aprendizagem de programação. Cheah (2020) aponta que, durante o processo de aprender a programar, os estudantes mobilizam componentes do PC como abstração e pensamento algorítmico, ao lado de capacidades mais amplas como a resolução de problemas. Wong e Cheung (2020) reforçam essa perspectiva ao registrar que os estudantes demonstraram consciência da relação entre PC, programação e competências do século XXI.

No contexto pedagógico, o uso do PC pode contribuir para a organização do raciocínio dos estudantes. A definição de objetivos claros, associada ao uso de estratégias baseadas no PC, pode orientar a construção de código, reduzir dificuldades no processo de aprendizagem e melhorar os resultados (Chang; Lin; Lin, 2024). Mais do que reiterar que a programação envolve a resolução de problemas, o PC oferece uma forma de organizar esse processo por meio de componentes como decomposição, reconhecimento de padrões, abstração e elaboração de algoritmos. Assim, sua contribuição está na organização das etapas envolvidas na construção de soluções, permitindo que os estudantes avancem de representações mais simples para estruturas mais elaboradas. A integração do PC em modelos de ensino tem sido apontada como uma possibilidade de ampliar a proficiência dos estudantes (Jalinus; Putra *et al.*, 2024).

No ensino de programação, essa organização do raciocínio também se relaciona a processos metacognitivos envolvidos no acompanhamento da própria aprendizagem. Esses processos permitem ao estudante monitorar o próprio raciocínio e avaliar a eficácia das estratégias utilizadas. Loksa *et al.* (2022) distinguem metacognição, entendida como o conhecimento que o aprendiz possui sobre seu controle cognitivo, de autorregulação, relacionada às decisões tomadas a partir desse monitoramento.

Durante atividades de programação, processos metacognitivos podem ser observados quando o estudante identifica erros no código e reconsidera a estratégia utilizada antes de tentar corrigi-los, comportamento percebido apenas em parte dos participantes (Loksa *et al.*, 2022). Os autores indicam ainda que a instrução explícita sobre etapas e estratégias do processo de programação pode favorecer a consciência metacognitiva, permitindo que os estudantes acompanhem o próprio progresso e reflitam sobre as decisões tomadas durante a resolução dos problemas.

Hu (2024), ao revisar estudos sobre o impacto da programação nas competências do

século XXI, registra a ausência de investigações específicas sobre a relação entre programação e metacognição, apontando uma lacuna no campo. Essa discussão reforça a relevância de considerar, no ensino de programação, não apenas os aspectos técnicos da construção de soluções, mas também os processos pelos quais o estudante percebe e acompanha o próprio aprendizado.

2.3.3 Computação Desplugada como estratégia para o desenvolvimento do PC

A Computação Desplugada corresponde a atividades de ensino de Computação realizadas sem computador ou em formato não digital (Rehder *et al.*, 2024), caracterizadas por desafios lúdicos, discussões mediadas e resolução de problemas em grupo (Huang; Looi, 2021). Huang e Looi (2021) observam que a justificativa mais recorrente para seu emprego em comunidades desassistidas é o acesso insuficiente ou instável a computadores, condição também apontada por Hyury *et al.* (2024) entre os motivadores da escolha dessas atividades no contexto brasileiro.

Foi com base na situação socioeconômica brasileira que Brackmann (2018) justificou a opção pelo formato desplugado, apresentado como alternativa para que todo o país, independentemente da situação da escola, possa usufruir dos benefícios do PC. Em estudo com estudantes dos anos finais do Ensino Fundamental da rede pública de Madri, sem experiência formal de programação, o autor registrou ganho de desempenho em teste de PC após um conjunto de atividades desplugadas, ganho não observado no grupo que seguiu o ensino habitual (Brackmann, 2018).

Li *et al.* (2022), ao compararem atividades desplugadas e exercícios de programação, verificaram que ambas as abordagens contribuem para o desenvolvimento do PC, com resultados mais expressivos para os exercícios de programação e sem variação consistente conforme o nível de ensino. Montuori *et al.* (2024) chegam a leitura distinta, ao atribuírem as diferenças de eficácia ao grau de estruturação da intervenção e à idade dos participantes, e não à modalidade adotada. A literatura, portanto, não é conclusiva quanto ao peso do formato desplugado, questão retomada na discussão dos resultados.

No cenário brasileiro, Hyury *et al.* (2024) identificam o ensino da lógica de programação e as habilidades do PC como conteúdos predominantes nas atividades desplugadas voltadas ao Ensino Fundamental II, abordados em grande parte por meio de conceitos da disciplina de Matemática. Os autores observam que a produção voltada a esse nível de ensino permanece limitada.

Huang e Looi (2021) ponderam que os achados sobre como essas atividades desenvolvem o PC ainda são limitados, uma vez que a presença da programação nas intervenções dificulta isolar seus efeitos, e acrescentam a necessidade de melhor fundamentação em teorias de aprendizagem. Rehder *et al.* (2024) verificaram que o êxito dessas atividades depende do conhecimento prévio dos estudantes, da complexidade da tarefa e do desenho pedagógico adotado, e que o uso ou não do computador não se mostrou fator decisivo em si mesmo.

Ao remover o computador, o projeto do algoritmo separa-se do ato de programar, ainda que algoritmos formulados em linguagem humana sejam mais difíceis de testar (Huang; Looi, 2021). Sim, Teow e Lau (2021) observam que atividades desplugadas podem contribuir para a capacidade de programação mesmo entre estudantes sem experiência prévia com linguagens, por permitirem concentrar esforços na resolução de problemas antes do contato formal com a sintaxe. No contexto da educação profissional, Geraldes e Afonseca (2024) relatam forte engajamento em atividades desplugadas realizadas em uma disciplina de algoritmos de curso técnico integrado, embora alguns estudantes tenham apresentado dificuldades no trabalho em grupo.

2.4 Conceitos de *Soft Skills*

As SS são habilidades não técnicas, cada vez mais reconhecidas como fundamentais para o desenvolvimento profissional e acadêmico, especialmente diante das exigências do mercado de trabalho atual (Schwartz, 2016; Burbekova, 2021; Desidério *et al.*, 2023; Fojcik *et al.*, 2023). Esse reconhecimento está associado à necessidade de profissionais capazes de atuar em contextos dinâmicos, nos quais a interação, a adaptação e a colaboração [1] são constantemente exigidas. Nesse cenário, essa exigência se reflete na demanda por indivíduos que consigam integrar habilidades técnicas com SS, como comunicação, trabalho em equipe e pensamento crítico (SBC, 2025).

Embora sejam distintas das *Hard Skills* (HS), compreendidas aqui como habilidades técnicas, as SS podem ser desenvolvidas de forma integrada a essas habilidades, ampliando a capacidade dos estudantes de aprender, pensar e atuar em cenários complexos (Yurchenko *et al.*, 2024). Essa relação não se limita a um complemento funcional, mas amplia as possibilidades de atuação dos indivíduos em situações que exigem tomada de

[1] Neste estudo, os termos Colaboração e Trabalho em Equipe são tratados como equivalentes, em conformidade com a literatura sobre *Soft Skills*, que frequentemente os utiliza de forma intercambiável (Matturro; Raschetti; Fontán, 2019).

decisão, interação e adaptação. As SS influenciam a forma como os indivíduos aprendem, pensam e aplicam conhecimentos, atuando como suporte para o uso adequado das habilidades técnicas (Kapustina; Matyushina, 2023).

Nos últimos anos, pesquisadores discutem o tema, porém ainda não há uma classificação padronizada, o que indica a ausência de consenso conceitual e a diversidade de interpretações presentes na literatura (Villegas, 2024; Yurchenko *et al.*, 2024; Espina-Romero *et al.*, 2023). Essa diversidade pode ser observada na variedade de termos utilizados para descrever essas habilidades, conforme o contexto ou tipo de atividade, como no estudo de Culcasi e Venegas (2023), em que foram identificados 17 termos diferentes associados a essas competências, como *"soft skills", "generic skills", "transversal competencies", "twenty-first century skills", "non-cognitive behavioral skills", "professional skills", "interpersonal skills", "sustainability competencies", "vocational skills", "employability skills", "practical skills", "social skills", "skills for sustainable development", "skills", "competencies", "capabilities"* e *"abilities"*.

De modo geral, as SS referem-se a um conjunto de atributos pessoais e competências comportamentais que influenciam a interação do indivíduo com seu ambiente social e profissional (Yurchenko *et al.*, 2024; Espina-Romero *et al.*, 2023; Kapustina; Matyushina, 2023). Essas habilidades incluem comunicação, pensamento crítico, criatividade e capacidade de adaptação (Shekh-Abed; Barakat, 2022; Zamora-Hernandez; Rodriguez-Paz; Gonzalez-Mendivil, 2023). Assim, as SS podem ser compreendidas como competências que orientam a forma como o indivíduo se posiciona, interage e responde a diferentes situações. A Tabela 1 reúne definições de SS presentes na literatura, o que mostra a falta de uma definição única para o termo.

Tabela 1: Definições de *Soft Skills*

Definição	Autor/Ano
Soft skills englobam traços de personalidade, habilidades não cognitivas e socioemocionais, que podem variar quanto ao grau de permanência e desenvolvimento ao longo da vida.	(Heckman; Kautz, 2012)
São compreendidas como habilidades interpessoais e pessoais que complementam as *hard skills*.	(Chaibate *et al.*, 2020)
Soft skills correspondem a habilidades sociais classificadas como uma combinação de traços de personalidade, comportamentos e atitudes sociais, que permitem a comunicação eficaz, a colaboração e a gestão de conflitos.	(Burbekova, 2021)
Soft skills consistem em uma combinação de traços de personalidade e habilidades interpessoais desenvolvidas por meio de interações cotidianas.	(Wang; Herzog, 2023)
Soft skills abrangem habilidades que vão além do contexto profissional, como comunicação, pensamento e gestão de emoções, sendo desenvolvidas ao longo da vida pessoal.	(Kapustina; Matyushina, 2023)
Soft skills correspondem ao conjunto de competências sociais e de comunicação que moldam a capacidade de interação dos indivíduos, sendo valorizadas pelas organizações.	(Espina-Romero *et al.*, 2023)
Soft skills são entendidas como qualidades pessoais e competências sociais não vinculadas diretamente a atividades profissionais específicas.	(Yurchenko *et al.*, 2024)
Soft skills envolvem competências como comunicação, pensamento crítico, flexibilidade e inteligência emocional, contribuindo para a preparação profissional.	(Villegas, 2024)
Soft skills incluem liderança, resolução de problemas, habilidades socioemocionais, comunicação interpessoal e autogestão.	(Mohammed; Ozdamli, 2024)
Soft skills referem-se a um conjunto de habilidades e competências necessárias em diferentes áreas profissionais, relacionadas ao desempenho e à adaptação a contextos dinâmicos.	(Dadelo, 2025)

Fonte: Autoria própria.

A análise das definições apresentadas na Tabela 1 reforça que as SS envolvem aspectos relacionados à comunicação, interação social, pensamento crítico e adaptação a diferentes contextos. Embora apresentem variações conceituais, as SS são frequentemente associadas à capacidade de colaborar, resolver problemas e atuar em cenários dinâmicos. Além disso, observa-se que não devem ser tratadas de forma isolada em relação às HS, mas como elementos complementares que contribuem para a aplicação do conhecimento técnico em situações práticas.

Com base nas definições apresentadas, adota-se neste estudo a seguinte definição: *"as SS são competências interpessoais e intrapessoais que, associadas a habilidades técnicas, sustentam a comunicação, a colaboração e a resolução estruturada de problemas em contextos formativos e profissionais."*

A definição adotada situa a resolução de problemas como aquilo que as SS ajudam a

sustentar, e não como uma competência ao lado das demais. Essa escolha responde ao fato de ela aparecer na literatura ora como competência de SS (Mohammed; Ozdamli, 2024; Matturro; Raschetti; Fontán, 2019), ora como resultado associado ao PC (Wing, 2006; Montuori *et al.*, 2024), ora como finalidade da própria atividade de programação (Medeiros; Falcão; Ramalho, 2021; Apeanti; Essel, 2021). Por atravessar esses três domínios, ela é tratada nesta tese como objetivo transversal do *framework*, e não como componente do PC nem como competência de SS.

2.4.1 *Soft Skills* na educação em computação

No contexto educacional brasileiro, o desenvolvimento das SS tem despertado interesse crescente, especialmente a partir de diretrizes voltadas à formação integral dos estudantes (Brasil, 2017b, 2022). A incorporação das SS no currículo busca ampliar o processo formativo para além do domínio técnico, considerando aspectos sociais e emocionais no desenvolvimento dos estudantes. A promoção das SS por meio de práticas estruturadas reforça seu caráter ensinável, sendo a escola um ambiente adequado para esse desenvolvimento (Brasil, 2021). Nesse sentido, sua presença amplia as possibilidades de aprendizagem ao favorecer a comunicação, a colaboração e a participação ativa dos estudantes.

Apesar desse reconhecimento, existem limitações evidentes na forma como as SS são abordadas pelas instituições de ensino. O distanciamento entre as demandas do mundo contemporâneo e as práticas educacionais dificulta a formação de estudantes preparados para cenários em constante evolução (Villegas, 2024). Além disso, métodos tradicionais baseados em aulas expositivas tendem a limitar oportunidades de interação e colaboração, reduzindo o desenvolvimento dessas habilidades (Cheng; Lee; Chan, 2018; SBC, 2025).

Porém, a simples organização de estudantes em grupos não garante o desenvolvimento dessas habilidades, sendo necessário o ensino intencional de competências relacionadas ao trabalho em equipe. Por essa razão, ambientes educacionais que incentivam práticas colaborativas são considerados mais adequados para o desenvolvimento das SS (Silva Devincenzi *et al.*, 2023).

Outro aspecto importante está relacionado à formação docente. Na área de Computação, muitos professores não receberam, em sua formação inicial, suporte suficiente para trabalhar aspectos relacionados às SS ou empregar metodologias que favoreçam seu desenvolvimento (Prabawati; Agustika, 2020; SBC, 2025). Sem essa formação, sua aplicação tende a se tornar limitada no contexto educacional (Prabawati; Agustika, 2020). Nessa direção apontam as *Diretrizes para o Ensino de Computação na Educação Básica* e os

Grandes Desafios da Educação em Computação 2025–2035, que reforçam a necessidade de preparar professores e estudantes para competências que ultrapassam o domínio técnico (SBC, 2025).

No ensino de programação, essa relação entre habilidades técnicas e SS fica mais clara. O desenvolvimento de algoritmos e soluções computacionais envolve não apenas conhecimento técnico, mas também pensamento crítico, organização do raciocínio e colaboração (Zamora-Hernandez; Rodriguez-Paz; Gonzalez-Mendivil, 2023; Mohammed; Ozdamli, 2024). Nesse contexto, as SS influenciam a forma como o estudante participa do processo de aprendizagem e interage na construção de soluções, especialmente em atividades que exigem cooperação (Younis *et al.*, 2019).

Estudos indicam que, apesar da formação técnica, muitos estudantes apresentam dificuldades relacionadas à comunicação, trabalho em equipe e resolução de problemas, o que amplia a distância entre a formação acadêmica e as demandas do mercado (Mohammed; Ozdamli, 2024). Esse cenário está associado ao foco predominante em conteúdos técnicos, limitando o desenvolvimento das SS (SBC, 2025).

Também existem desafios na forma como as SS são avaliadas. A literatura aponta falhas em sua integração ao currículo e na ausência de métodos padronizados de avaliação (Dadelo, 2025; Yurchenko *et al.*, 2024). Tal limitação dificulta acompanhar o desenvolvimento dessas habilidades ao longo da formação, comprometendo sua integração ao processo educativo.

2.4.2 *Soft Skills* no mercado de trabalho

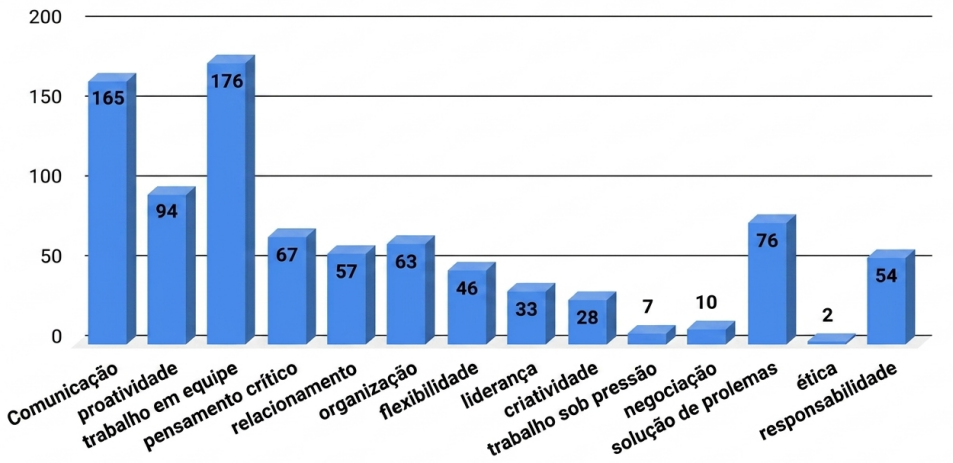
Com as transformações tecnológicas e a expansão da Inteligência Artificial, o mercado de trabalho tem passado por mudanças contínuas, demandando dos profissionais habilidades que vão além do domínio técnico. Nesse contexto, conhecimentos técnicos isolados deixam de ser suficientes para atuação em ambientes dinâmicos (Wang; Herzog, 2023; Desidério *et al.*, 2023; Silva; Albuquerque, 2023).

Esse reconhecimento também aparece em estudos empíricos sobre as demandas da indústria. Galster *et al.* (2022), ao analisarem 530 anúncios de vagas em posições relacionadas à computação na Nova Zelândia, identificaram referências explícitas a SS em 82% dos anúncios e mapearam 17 tipos de competências. Os autores observam que companhias do setor solicitam SS em suas vagas, e que as habilidades comunicacionais dominam o conjunto de competências demandadas, independentemente do tipo de posição. Nudin,

Warsito e Wibowo (2022), em estudo aplicado a uma base de 4.549 egressos da educação superior, identificam que as SS exercem impacto relevante sobre a absorção dos egressos pela indústria, com maiores níveis de SS associados a maior probabilidade de obtenção de emprego em até seis meses.

Na área de computação, essa exigência assume características próprias. A atuação profissional envolve interação constante entre diferentes perfis, desenvolvimento de soluções em conjunto e comunicação de resultados técnicos. No estudo de Matturro, Raschetti e Fontán (2019), ao analisarem as SS mais requisitadas entre engenheiros de *software*, destacam-se comunicação (91%), trabalho em equipe (68%) e análise crítica (55%). Também foram identificadas habilidades como liderança (48%), resolução de problemas (48%) e autonomia (43%), relacionadas à atuação em equipe e à resolução conjunta de problemas. Por sua vez, Silva e Albuquerque (2023) analisaram anúncios de vagas na área de TI e constataram que competências como trabalho em equipe, comunicação, proatividade e resolução de problemas são requisitos frequentes no mercado. A Figura 8 apresenta esse panorama ao indicar a frequência dessas habilidades nos anúncios de vagas na área de TI.

Figura 8: Visão geral das *Soft Skills* requeridas nos anúncios



Fonte: Silva e Albuquerque (2023)

Wilson e Marnewick (2018), em estudo qualitativo com participantes da indústria de engenharia, registram que a indústria descreve sua atuação como dependente de comunicação e trabalho em equipe, com relacionamentos interpessoais ocupando papel principal no sucesso de projetos, o que, segundo os autores, sugere que habilidades técnicas e SS não podem ser tratadas de forma desconectada no exercício profissional. Os mesmos autores identificam, na comparação entre o currículo acreditado e a demanda do setor, que habilidades interpessoais, flexibilidade, liderança e habilidades organizacionais constituem lacunas do currículo em relação às expectativas da indústria.

Essa tendência também aparece em relatórios institucionais. O relatório *The Future of Jobs* do Fórum Econômico Mundial destaca competências como pensamento analítico, pensamento criativo, resiliência, aprendizagem contínua e empatia entre as mais demandadas no horizonte de 2027 (WEF, 2023).

Os resultados discutidos indicam que o ensino de programação não pode se restringir ao desenvolvimento de habilidades técnicas. Ao reconhecer que o exercício profissional na área exige, simultaneamente, domínio técnico e SS, a formação no contexto da EPTNM é provocada a integrar essas duas dimensões. Essa integração, no entanto, não se confunde com submissão ao mercado, visto que ela responde a um princípio formativo mais amplo, discutido na Seção 2.6, segundo o qual a educação profissional integra formação geral e formação técnica em um mesmo percurso formativo.

2.5 Integração entre PC e SS no ensino de programação

A integração entre PC e SS no ensino de programação tem sido tema de estudos recentes, ainda que de forma incipiente. Tomić *et al.* (2019) argumentam que um bom programador depende tanto de habilidades técnicas, relacionadas a métodos, técnicas e tecnologias, quanto de SS, em especial colaboração, comunicação, resolução de problemas e pensamento crítico. Nesse contexto, os componentes do PC contribuem para a organização do raciocínio e para o desenvolvimento de soluções em programação. Para os autores, programadores precisam de ambos para atuar no mercado de trabalho, mas os cursos de programação concentram-se principalmente em habilidades técnicas, dedicando atenção limitada às SS. Esse desequilíbrio também é registrado por Tsortanidou, Daradoumis e Barberá (2019), segundo os quais a literatura ainda carece de estudos que relacionem PC, colaboração, criatividade e habilidades de letramento midiático.

Na prática da programação, PC e SS também aparecem de forma integrada. Wong e Cheung (2020) relatam que estudantes envolvidos com programação demonstraram percepção da relação entre PC, programação e competências do século XXI, indicando que PC e SS emergem em conjunto na experiência de aprender a programar. Hu (2024), em revisão sistemática sobre o impacto da programação nas competências do século XXI, identifica efeitos significativos em sub-habilidades associadas aos modos de pensar, como criatividade, resolução de problemas e pensamento crítico, além da capacidade de aprender a aprender e aos modos de trabalhar, como comunicação, colaboração e trabalho em equipe. De forma semelhante, Jalinus, Putra *et al.* (2024) propõem a integração de

componentes do PC ao modelo de aprendizagem baseado em projetos para favorecer o desenvolvimento das competências 4C [2], pensamento crítico, criatividade, comunicação e colaboração, na educação vocacional, especificamente em aulas de programação móvel.

A coexistência de PC e SS em atividades de programação, no entanto, não garante seu desenvolvimento. Hsu *et al.* (2022), ao discutirem intervenções interdisciplinares que reúnem PC e outras competências, observam que a simples presença desses componentes em uma mesma atividade não assegura o desenvolvimento das habilidades esperadas, sendo necessário planejar estratégias que considerem cada uma delas de forma intencional. Maitz, Paleczek e Danielowitz (2022) acrescentam que, apesar de habilidades socioemocionais e PC estarem relacionados à resolução de problemas, intervenções pedagógicas que atuem sobre esses aspectos simultaneamente ainda são raras na literatura. Nouri *et al.* (2020), por sua vez, registram que as estratégias didáticas de professores frequentemente carecem de foco explícito no desenvolvimento de habilidades de PC e, por consequência, também de habilidades colaborativas e de resolução criativa de problemas.

Ciancarini, Missiroli e Russo (2020) ampliam essa perspectiva ao argumentar que o PC tem sido tradicionalmente considerado uma atividade individual, com trabalho em equipe e outras SS frequentemente desconsiderados, e por vezes até desencorajados, em cursos introdutórios de programação. Para os autores, educadores não devem se limitar a promover práticas de engenharia de *software*, devendo também fomentar a colaboração e capacitar equipes de estudantes a cooperar em problemas complexos. Essa observação também é apontada por SBC (2025), que registra a persistência de uma lacuna na formação dos estudantes de computação no Brasil, na medida em que as metodologias de ensino tradicionalmente adotadas não favorecem o desenvolvimento das SS.

A integração entre PC e SS no ensino de programação, portanto, não se sustenta apenas na coexistência desses componentes em atividades de programação, mas no reconhecimento de que sua integração responde a uma necessidade formativa: preparar estudantes capazes de compreender, comunicar e resolver problemas computacionais em contextos que exigem cooperação. Essa perspectiva orienta o desenvolvimento conjunto dos componentes do PC e das SS no ensino de programação, abordagem adotada nesta tese, com atenção particular ao recorte da EPTNM, segmento em que estudos voltados especificamente a essa proposta permanecem escassos (Rege; Salgado; Viterbo, 2025).

[2] Framework de Habilidades de Aprendizagem e Inovação, composto pelas competências de criatividade, pensamento crítico, comunicação e colaboração (Partnership for 21st Century Learning, 2019).

2.6 Educação profissional técnica no Brasil

Esta seção apresenta o contexto da EPTNM, no qual se insere a aplicação do *framework* SoPensa. A discussão se concentra na organização institucional dessa modalidade e nos princípios pedagógicos que orientam a formação técnica integrada ao ensino médio, oferecendo base para situar o ensino de programação no contexto investigado.

A EPTNM no Brasil é definida como uma modalidade integrada aos diferentes níveis de ensino e às esferas do trabalho, da ciência e da tecnologia. A definição aparece tanto no material institucional do Ministério da Educação (MEC) quanto na discussão sobre a redefinição da Lei nº 9394/96 - LDB promovida pela Lei nº 11.741/2008, que integrou as ações da EPTNM aos diferentes níveis e modalidades da educação nacional (Vieira; Souza Júnior, 2016; Pacheco, 2012). A inclusão da educação profissional na educação básica posiciona a formação técnica no mesmo sistema educacional, incorporando-a ao campo da educação básica e aproximando-a das demandas ligadas à formação para atividades científicas e tecnológicas (Brasil, 2008).

A estrutura institucional se consolida com a Lei nº 11.892/2008, responsável pela criação da Rede Federal de Educação Profissional, Científica e Tecnológica. Os Institutos Federais passam a ser definidos como instituições pluricurriculares e multicampi, voltadas à oferta de educação básica, técnica e superior com atuação em pesquisa aplicada e desenvolvimento de soluções tecnológicas. A legislação estabelece a prioridade da oferta de cursos técnicos integrados e reforça a centralidade dessa modalidade na organização da Rede Federal (Brasil, 2008).

No ensino médio, a relação entre formação geral e formação profissional orienta a organização curricular. O Documento Base da EPTNM Integrada ao Ensino Médio associa essa forma de oferta a melhores resultados pedagógicos no processo de integração entre formação geral e técnica, ao indicar a necessidade de desenvolver conteúdos dessas duas dimensões no mesmo percurso formativo (Brasil, 2007). A proposta considera trabalho, ciência, cultura e tecnologia como elementos inseparáveis da formação, indicando a superação da separação entre essas áreas, além de deslocar o foco da formação para o desenvolvimento integral do estudante (Brasil, 2007).

A discussão apresentada por Pacheco (2012) amplia essa interpretação ao questionar a redução da educação profissional a competências voltadas exclusivamente ao mercado. O autor defende a formação integrada como princípio educacional, ao criticar abordagens que reduzem a educação à sua funcionalidade e ao propor a incorporação de dimensões

como ética, metodologia científica, trabalho em equipe e desenvolvimento de projetos no processo formativo (Pacheco, 2012).

A relação entre formação para o trabalho e formação ampla do sujeito acompanha a trajetória da educação profissional brasileira. A esse respeito, Moura (2008) destaca que a educação no país historicamente se organizou de forma separada, com uma formação acadêmica voltada às elites e outra, de caráter instrumental, destinada à classe trabalhadora. Na educação técnica integrada ao ensino médio, essa separação se reflete na coexistência entre conteúdos operacionais e o desenvolvimento de competências que ultrapassam a esfera técnica.

Nesse sentido, Ramos (2008) argumenta que a formação profissional não se limita ao exercício imediato de uma profissão, mas envolve também a compreensão das dinâmicas produtivas e a atuação autônoma e crítica diante da atividade profissional. Para a autora, conceitos ensinados de forma isolada, sem relação com os fundamentos científicos que os sustentam, tendem a permanecer restritos ao contexto em que foram aprendidos (Ramos, 2008). No ensino de programação na educação profissional técnica, essa discussão ajuda a compreender que o desenvolvimento dos componentes do PC e das SS não depende apenas do domínio de comandos e sintaxes, mas também da capacidade de mobilizar raciocínios em diferentes situações de resolução de problemas.

A Figura 9 apresenta a organização da EPTNM, destacando a qualificação profissional técnica, os cursos técnicos em suas formas de oferta (integrada, concomitante, concomitante intercomplementar, articulada e subsequente) e a especialização técnica de nível médio. Essa organização contribui para visualizar o percurso formativo previsto nessa modalidade.

Figura 9: Itinerário da educação profissional técnica de nível médio

Fonte: Adaptado Brasil (2024a)

O Catálogo Nacional de Cursos Técnicos orienta o planejamento curricular ao definir perfil profissional, carga horária mínima e campos de atuação, além das possibilidades de continuidade formativa. No eixo de Informação e Comunicação, são definidos cursos como Técnico em Desenvolvimento de Sistemas, Técnico em Informática, Técnico em Informática para Internet, Técnico em Programação de Jogos Digitais, Técnico em Manutenção e Suporte em Informática, Técnico em Redes de Computadores e Técnico em Telecomunicações (Brasil, 2020).

Os cursos definidos no Catálogo caracterizam a computação como eixo tecnológico na educação profissional técnica, com perfis profissionais e campos de atuação definidos em âmbito nacional, além de contemplar, na formação dos estudantes, aspectos relacionados à liderança de equipe e à ética profissional, ampliando a formação para além das competências técnicas.

A educação profissional técnica, especialmente na forma integrada, organiza a formação no ensino médio ao reunir formação geral e profissional em um mesmo percurso formativo. Nos Institutos Federais, essa oferta articula ensino, pesquisa aplicada e desenvolvimento tecnológico. Essa estrutura segue a integração entre educação básica e educação profissional prevista nos documentos oficiais (Brasil, 2007; Pacheco, 2012).

Nesse contexto, o ensino requer alinhamento entre conteúdos da formação geral, formação técnica e práticas profissionais, conforme as diretrizes da educação profissional técnica, considerando as características do contexto da educação profissional técnica em que essa formação ocorre. A compreensão desse cenário permite situar o ensino de programação no âmbito da EPTNM, que constitui o espaço de inserção desta pesquisa.

Um aspecto que condiciona o ensino de programação nesse contexto é a defasagem educacional com que parte dos estudantes ingressa nos cursos técnicos integrados. A literatura aponta que dificuldades relacionadas à matemática, ao raciocínio lógico e à interpretação de texto comprometem a aprendizagem desde as disciplinas introdutórias (Silva, 2018; Ribeiro *et al.*, 2020; Silva; Moreira, 2021). Assim, ao comparar o panorama internacional e nacional sobre o ensino de programação, Medeiros, Falcão e Ramalho (2021) observaram que publicações brasileiras atribuem maior ênfase do que as internacionais às dificuldades relacionadas ao raciocínio lógico e à abstração, o que pode indicar particularidades associadas à trajetória escolar dos estudantes brasileiros antes do ingresso na educação técnica. Os autores ainda registram como a ausência de conhecimentos de base influencia aspectos comportamentais, como confiança e motivação dos estudantes.

No ensino técnico integrado, essa discussão ganha relevância particular, pois os alunos

ingressam simultaneamente no ensino médio e na formação técnica, enfrentando uma estrutura curricular extensa que pressupõe conhecimentos nem sempre consolidados na etapa anterior. Assim, a defasagem formativa passa a ser compreendida como um elemento que não se restringe a uma disciplina específica, mas se relaciona às condições educacionais que antecedem o ingresso na formação técnica.

A discussão apresentada nesta seção mostra que a EPTNM se organiza a partir da integração entre formação geral e formação técnica, sustentada pela noção de trabalho como princípio educativo (Ramos, 2008; Pacheco, 2012), concepção que dialoga com a fundamentação desenvolvida nas seções anteriores. Ao envolver simultaneamente o desenvolvimento de componentes do PC e de SS, o ensino de programação na EPTNM encontra base conceitual na proposta de integração que orienta essa modalidade educacional. Dessa forma, o recorte adotado nesta tese decorre do referencial teórico discutido, e não apenas de uma escolha metodológica.

2.7 Considerações finais do capítulo

Este capítulo apresentou os conceitos teóricos que fundamentam esta pesquisa, organizados em seis eixos: o ensino e a aprendizagem de programação, as teorias de aprendizagem, o PC, as SS, a integração entre PC e SS no ensino de programação e o contexto da EPTNM.

A revisão do ensino de programação indicou que a aprendizagem nessa área envolve dificuldades de diferentes naturezas. De um lado, limitações cognitivas relacionadas à abstração, ao raciocínio lógico e à formulação de problemas comprometem a construção de soluções. De outro, fatores comportamentais como desmotivação, frustração e perda de confiança afetam a permanência e o engajamento dos estudantes. A coexistência dessas dificuldades sugere que abordagens centradas exclusivamente em aspectos técnicos podem ser insuficientes, demandando perspectivas que considerem também dimensões comportamentais e formativas presentes na aprendizagem de programação. As metodologias ativas discutidas neste capítulo respondem parcialmente a esse cenário, ainda que sua adoção dependa de formação docente e de condições institucionais.

Diante desse cenário, as teorias de aprendizagem apresentadas oferecem perspectivas distintas para compreender essas dificuldades. A AS foi adotada por contribuir para a compreensão de como o conhecimento é organizado e integrado ao repertório prévio do estudante, aspecto relevante diante das defasagens educacionais identificadas na literatura. O Construcionismo, por sua vez, fundamenta a aprendizagem por meio da criação e

da experimentação, o que se alinha à natureza prática do ensino de programação, em que o estudante constrói soluções, testa possibilidades e revisa suas produções. As duas abordagens foram adotadas de forma complementar, associando a organização cognitiva do conhecimento à elaboração prática de soluções e artefatos.

Em complemento às teorias de aprendizagem, o PC foi discutido a partir de seus componentes, ou seja, decomposição, reconhecimento de padrões, abstração e algoritmos, os quais oferecem uma forma de organizar o raciocínio voltado à resolução de problemas. Essa organização do raciocínio proposta pelo PC é relevante para o ensino de programação, considerando que orienta as etapas envolvidas na resolução de problemas e permite que o estudante avance de problemas mais simples para estruturas mais complexas.

A computação desplugada foi apresentada como estratégia para introduzir esses componentes sem a barreira da sintaxe, com resultados que a literatura ainda não trata de forma conclusiva. Além disso, a dimensão metacognitiva discutida ao longo deste capítulo aponta que a instrução explícita sobre etapas e estratégias do processo de programação pode favorecer o acompanhamento do próprio aprendizado.

Em relação às SS, a discussão voltou-se às competências que influenciam a forma como o estudante participa do processo de aprendizagem, especialmente em atividades que envolvem colaboração, comunicação e tomada de decisão. Conforme discutido nas seções anteriores, a literatura aponta desafios na integração das SS ao ensino de programação, tanto na ausência de métodos padronizados de avaliação quanto na dificuldade de definir quais competências priorizar em cada etapa formativa.

A aproximação entre PC e SS deu origem à seção sobre a integração entre esses eixos no ensino de programação, apresentada como eixo conceitual desta tese. A literatura indica que a integração se manifesta empiricamente na prática de programar, ainda que os referenciais discutidos não estabeleçam como esse desenvolvimento conjunto deve ser conduzido pedagogicamente.

O contexto da EPTNM foi apresentado como o espaço em que esta pesquisa se desenvolve. A organização dessa modalidade, que reúne formação geral e formação técnica no mesmo percurso formativo, impõe condições específicas ao ensino de programação: os estudantes ingressam simultaneamente no ensino médio e na formação técnica, enfrentando uma estrutura curricular extensa que pressupõe conhecimentos nem sempre consolidados na etapa anterior. Essa configuração torna o contexto do ensino técnico integrado distinto do ensino superior e do ensino fundamental, níveis em que se concentra a maioria dos estudos sobre PC e SS, conforme será detalhado no capítulo seguinte.

O conjunto dos conceitos discutidos neste capítulo oferece fundamento para a questão de pesquisa apresentada no Capítulo 1, ao reunir, em uma mesma discussão teórica, os componentes do PC, as SS e o contexto da EPTNM. Esses conceitos orientam tanto a construção do *framework* proposto nesta tese quanto a interpretação dos dados obtidos nos capítulos de análise e discussão, principalmente no que se refere à organização do raciocínio, às dinâmicas de aprendizagem e às SS observadas durante a intervenção. O próximo capítulo apresenta o mapeamento sistemático da literatura recente, no qual se examina como esses elementos têm sido tratados em estudos empíricos sobre o ensino de programação.

3 ESTADO DA ARTE

Este capítulo apresenta um estado da arte sobre o desenvolvimento do PC e das SS no ensino de programação. Para sua elaboração, foi realizado um MSL, conduzido a partir das diretrizes propostas por (Petersen *et al.*, 2008; Kitchenham; Charters, 2007). O protocolo detalhado do MSL encontra-se no Apêndice A. O capítulo está organizado em três seções: a caracterização dos 18 estudos selecionados; os resultados do mapeamento, que reúnem a análise das cinco questões secundárias e a discussão dos achados; e as considerações finais.

3.1 Caracterização dos estudos selecionados

O processo de mapeamento descrito no Apêndice A resultou na seleção de 18 estudos primários publicados entre 2020 e 2024. A Tabela 2 apresenta os estudos selecionados, com autor, ano, título, local de publicação e tipo (periódico ou conferência). A matriz de extração, com os dados levantados de cada estudo, encontra-se no Apêndice B.

Tabela 2: Estudos selecionados

ID	Autor(es) e Ano	Título do Estudo	Local de Publicação	Tipo
E01	Kukul e Çakır (2020)	Exploring the Development of Primary School Students' Computational Thinking and 21st Century Skills through Scaffolding	International Journal of Computer Science Education in Schools	Periódico
E02	Nouri et al. (2020)	Development of Computational Thinking, Digital Competence and 21st Century Skills When Learning Programming in K-9	Education Inquiry	Periódico
E03	Wong e Cheung (2020)	Exploring Children's Perceptions of Developing Twenty-First Century Skills through Computational Thinking and Programming	Interactive Learning Environments	Periódico
E04	Ciancarini, Missiroli e Russo (2020)	Exploiting Agile Practices to Teach Computational Thinking	International Workshop on DEVOPS, Springer	Conferência
E05	Manikutty (2021)	My Robot can tell stories: Introducing robotics and physical computing to children using dynamic dioramas	IEEE Frontiers in Education Conference	Conferência
E06	Valls Pou, Canaleta e Fonseca (2022)	Computational Thinking and Educational Robotics Integrated into Project-Based Learning	Sensors	Periódico
E07	Hsu et al. (2022)	Effects of a Pair Programming Educational Robot-Based Approach on Students' Interdisciplinary Learning of Computational Thinking and Language Learning	Frontiers in Psychology	Periódico
E08	Fernández e Roa Martín (2022)	Educational Policy Framework to promote Computational Thinking towards STEAM in Public Schools in Boyacá - Colombia	Conference for Engineering, Education and Technology	Conferência
E09	Maitz et al. (2022)	Simultaneously Fostering Computational Thinking and Social-Emotional Competencies in 4th Graders Using Scratch: A Feasibility Study	Proceedings of Mensch und Computer, ACM	Conferência
E10	Delzanno et al. (2022)	Experience-Based Training in Computer Science Education via Online Multiplayer Games on Computational Thinking	HELMeTO, Springer	Conferência
E11	Wang (2023)	The role of computer supported project-based learning in students' computational thinking and engagement in robotics courses	Thinking Skills and Creativity	Periódico
E12	Rocha et al. (2023)	Coding Together: On Co-located and Remote Collaboration between Children with Mixed-Visual Abilities	Conference on Human Factors in Computing Systems	Conferência
E13	Bodaker e Rosenberg-Kima (2023)	Online Pair-Programming: Elementary School Children Learning Scratch Together Online	Journal of Research on Technology in Education	Periódico
E14	Schwartz et al. (2023)	Teaching Computational Thinking with a Tangible Development Platform: An Exploratory Field Study at School with Kniwwelino	Education and Information Technologies	Periódico
E15	Vasconcelos et al. (2023)	Scratch4All Project - Educate for an All-inclusive Digital Society	EAEEIE Conference, IEEE	Conferência
E16	Yang (2024)	Coding With Robots or Tablets? Effects of Technology-Enhanced Embodied Learning on Preschoolers' Computational Thinking and Social-Emotional Competence	Journal of Educational Computing Research	Periódico
E17	Jalinus e Putra et al. (2024)	Implementation of Project-Based Learning Computational Thinking Models in Mobile Programming Courses	International Journal of Interactive Mobile Technologies	Periódico
E18	Kalluri et al. (2024)	Developing Future Computational Thinking in Foundational CS Education: A Case Study From a Liberal Education University in India	IEEE Transactions on Education	Periódico

Fonte: Autoria própria.

A distribuição dos 18 estudos mostra que 11 destes foram publicados em periódicos científicos (E01, E02, E03, E06, E07, E11, E13, E14, E16, E17 e E18) e sete em anais de conferências internacionais (E04, E05, E08, E09, E10, E12 e E15), com concentração nas

áreas de engenharia, educação em computação e interação humano-computador. Quanto às editoras e sociedades responsáveis, há diversidade de procedência, com presença de IEEE (E05, E15 e E18), ACM (E09 e E12), Springer (E04 e E10), Elsevier (E11), MDPI (E06), Frontiers (E07), Taylor & Francis (E02, E03 e E13) e demais editoras especializadas em educação e tecnologia. A diversidade de veículos mostra que as publicações sobre PC e SS distribuem-se tanto em periódicos e conferências da educação quanto da computação, refletindo a natureza interdisciplinar do tema.

Os 18 estudos selecionados abrangem diferentes níveis de ensino, abordagens pedagógicas e ferramentas tecnológicas voltadas ao desenvolvimento conjunto do PC e das SS no ensino de programação. A caracterização a seguir organiza os trabalhos pelos níveis educacionais investigados, descrevendo a contribuição de cada estudo.

Na educação infantil, verifica-se que apenas um estudo investiga o desenvolvimento de PC e SS. Trata-se do estudo de Yang (2024) [E16] que adota desenho experimental, no qual 73 pré-escolares de Hong Kong participam de dois programas de nove semanas, um com robôs de programação e outro com aplicativos para *tablet*; as crianças do grupo dos robôs registram desempenho superior em PC, enquanto os ganhos em competência socioemocional variam conforme o desempenho inicial dos participantes. A escassez de estudos nesse nível pode ser interpretada como sinal de que a integração entre PC e SS em idades muito tenras ainda é um terreno pouco explorado pela literatura recente.

No ensino fundamental, encontra-se a maior parte dos estudos analisados, com treze trabalhos voltados a esse nível de ensino, listados a seguir.

Em Hong Kong, Wong e Cheung (2020) [E03] acompanharam, durante um ano, atividades de programação com Scratch e KoduGameLab em uma escola primária com alunos de *Primary* 4, 5 e 6, identificando ganhos perceptíveis em criatividade, comunicação e colaboração a partir da percepção das próprias crianças.

O uso de *scaffolding* em atividades de programação de jogos com o KoduGameLab é investigado por Kukul e Çakır (2020) [E01], em estudo qualitativo realizado na Turquia, que identifica efeitos positivos sobre o PC, habilidades do século XXI e aspectos como autoconfiança e motivação. Na Suécia, Nouri *et al.* (2020) [E02] analisaram o desenvolvimento conjunto de PC, competência digital e habilidades do século XXI em estudantes do K-9, utilizando Scratch e robótica educacional, e relataram que essas competências tendem a se desenvolver de forma combinada quando integradas ao currículo.

Em comunidades rurais da Índia, Manikutty (2021) [E05] desenvolve uma proposta de

robótica e computação física baseada em *storytelling*, na qual as crianças constroem dioramas interativos com base nos fundamentos do Construcionismo de Papert, combinando criatividade, comunicação e pensamento crítico ao aprendizado de PC.

No contexto colombiano, Fernández e Roa Martín (2022) [E08] propõem um *framework* de política educacional voltado à promoção do PC na perspectiva STEAM em escolas públicas do departamento de Boyacá. A proposta combina aprendizagem baseada em problemas, projetos e jogos com o uso de Scratch e Python, e sinaliza que o desenvolvimento conjunto de PC e SS depende não apenas das estratégias didáticas, mas também de condições estruturais relacionadas à infraestrutura, formação docente e políticas públicas.

Já o estudo de Maitz, Paleczek e Danielowitz (2022) [E09], conduzido na Áustria com 18 estudantes do 4º ano, testa a viabilidade de um *workshop* de quatro dias com Scratch projetado para fomentar PC e competência socioemocional. Os resultados indicam que a atividade em pares foi particularmente benéfica e que os estudantes apreciaram tanto o trabalho com Scratch quanto os desafios de SS propostos pelo estudo.

Já em Portugal, Rocha *et al.* (2023) [E12] examinam a colaboração entre crianças com habilidades visuais distintas na aprendizagem de programação, comparando ambientes presenciais e remotos. Situada na interseção entre robótica, jogos e acessibilidade, a pesquisa contribui para a discussão sobre como o desenvolvimento de PC pode dialogar com competências de colaboração e empatia em contextos inclusivos.

Em Israel, Bodaker e Rosenberg-Kima (2023) [E13] analisam a programação em pares conduzida de forma remota com crianças do ensino fundamental aprendendo Scratch colaborativamente e registram que o formato *on-line* não compromete o desenvolvimento das competências comunicativas e de cooperação tradicionalmente associadas à programação em pares presencial.

Em uma escola de Barcelona, com estudantes de 12 a 14 anos, Valls Pou, Canaleta e Fonseca (2022) [E06] integram robótica educacional ao Scratch em proposta baseada em aprendizagem por projetos e indicam resultados mais favoráveis dessa abordagem no desenvolvimento de componentes do PC e de habilidades do século XXI quando comparada a metodologias tradicionais.

Com estudantes do 6º ano em Taiwan e Singapura, Hsu *et al.* (2022) [E07] investigam a programação em pares aliada à robótica educacional em atividades interdisciplinares que integram o aprendizado de PC ao desenvolvimento de competências em língua estrangeira, com ganhos significativos tanto em PC quanto na redução da ansiedade no aprendizado

da língua-alvo.

Na Itália, Delzanno *et al.* (2022) [E10] descrevem a Italian Coding League, competição online de programação que envolveu 609 estudantes de 29 turmas de nove regiões italianas. Fundamentada em aprendizagem experiencial e em jogos, a iniciativa indica que ambientes lúdicos e competitivos podem favorecer o desenvolvimento conjunto de PC e de competências socioemocionais, como cooperação, comunicação e resolução de conflitos.

á em Portugal, Vasconcelos *et al.* (2023) [E15] apresentam o projeto Scratch4All, que integra Scratch e robótica educacional em ações voltadas a uma sociedade digital mais inclusiva. Conduzido pelo consórcio CASPAE, em parceria com o Instituto Politécnico de Coimbra, o projeto abrange múltiplos níveis de ensino e reforça a dimensão social da aprendizagem.

Por fim, o Kniwwelino Classroom Kit, descrito por Schwartz *et al.* (2024) [E14], é uma plataforma tangível adotada em escolas de Luxemburgo, cuja aplicação abrange alunos de 8 a 16 anos e, segundo os autores, a abordagem favorece tanto o aprendizado de PC quanto o desenvolvimento de SS como comunicação e colaboração.

O ensino fundamental aparece como o nível mais investigado e também o mais heterogêneo em estratégias: *scaffolding*, robótica, jogos, programação em pares e plataformas tangíveis convivem em propostas que, na maior parte das vezes, tratam o PC como objetivo principal, enquanto as SS surgem como efeitos secundários observados pelos pesquisadores. Poucos estudos partem do princípio de planejar PC e SS conjuntamente desde o desenho da intervenção, o que sugere que a integração entre esses dois eixos ainda ocorre mais por consequência do que por intenção pedagógica explícita.

No ensino médio, dois estudos investigam o desenvolvimento conjunto de PC e SS em contextos de programação. Em pesquisa conduzida na Itália, Ciancarini, Missiroli e Russo (2020) [E04] exploram o uso de práticas ágeis para o ensino de PC, comparando quatro abordagens de desenvolvimento de *software*: programador solo, programação em pares, equipes auto-organizadas e equipes dirigidas. Com base em experimentos didáticos realizados ao longo de anos, os autores propõem o modelo de *Cooperative Thinking*, no qual o PC é potencializado pela combinação com habilidades sociais e práticas de trabalho em equipe. O estudo de Schwartz *et al.* (2024) [E14], já descrito no bloco anterior, também alcança o ensino médio em sua aplicação da plataforma Kniwwelino, o que reforça a versatilidade das abordagens tangíveis em diferentes níveis educacionais.

A presença reduzida de estudos no ensino médio, com apenas dois trabalhos entre os 18

analisados, e a ausência de investigações voltadas especificamente ao ensino médio técnico sinalizam que esta etapa permanece pouco explorada pela literatura recente, lacuna que será aprofundada na análise dos resultados do mapeamento.

No ensino superior, quatro estudos exploram o desenvolvimento conjunto de PC e SS no contexto da formação inicial em computação e áreas correlatas. Três deles são descritos a seguir, aos quais se soma o estudo de Ciancarini, Missiroli e Russo (2020) [E04], apresentado no bloco anterior, cujos experimentos didáticos abrangem também o ensino superior.

Com 79 estudantes universitários na China, Wang e Herzog (2023) [E11] investigam os efeitos da aprendizagem baseada em projetos com suporte computacional em cursos de robótica, observando que essa abordagem favoreceu tanto o desenvolvimento de PC quanto o engajamento dos estudantes, particularmente entre aqueles com mais de um ano de experiência prévia em programação. Na Indonésia, Jalinus, Putra *et al.* (2024) [E17] propõem um modelo de aprendizagem baseada em projetos voltado ao ensino de PC em cursos de programação móvel, com ganhos significativos em aspectos cognitivos e nas quatro competências do 4C (comunicação, colaboração, criatividade e pensamento crítico) em comparação com os grupos de controle.

Por último, em uma universidade indiana de educação liberal, Kalluri *et al.* (2024) [E18] propõem um modelo teórico que reúne seis tipos de pensamento (lógico, sistêmico, sustentável, estratégico, criativo e responsável) em uma pedagogia híbrida baseada na aprendizagem experiencial, autorreflexão e aprendizagem entre pares, voltada ao ensino introdutório de programação no primeiro ano de graduação.

O ensino superior aparece como espaço em que PC e SS podem ser tratados de forma integrada, possivelmente porque os contextos universitários permitem trabalhar projetos complexos e modelos pedagógicos mais sofisticados, como mostram E04, E11, E17 e E18. Entretanto, mesmo em países com experiências de integração entre formação acadêmica e preparação profissional, como Itália (E04) e Áustria (E09), nenhum dos 18 estudos investiga o ensino médio técnico, etapa que combina formação geral e profissional e poderia se beneficiar de estratégias utilizadas no ensino superior.

3.2 Resultados do mapeamento sistemático

Após a caracterização dos 18 estudos selecionados, esta seção apresenta os resultados estruturados a partir das questões que orientaram o mapeamento. O protocolo completo,

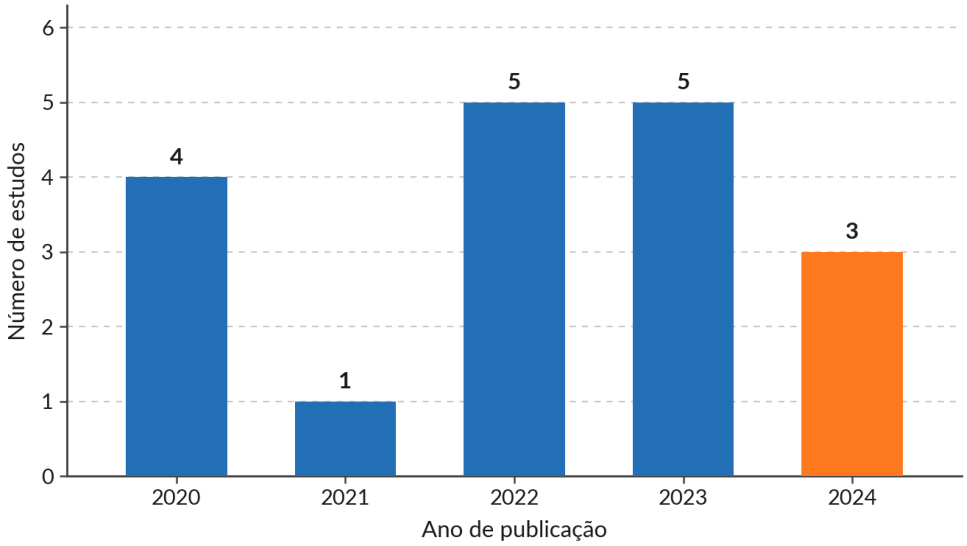
incluindo a *string* de busca, as bases consultadas, os critérios de inclusão e exclusão e o processo de seleção dos estudos, está descrito no Apêndice A. A construção deste estudo foi orientada por uma Pergunta Principal (QP) e cinco Perguntas Secundárias (PS1 a PS5). A questão principal busca compreender, de forma ampla, como a literatura aborda o tema da pesquisa: QP: **De que maneira a literatura aborda o desenvolvimento do PC e das SS no ensino de programação?**

As cinco perguntas secundárias fragmentam essa questão em eixos específicos: PS1: Como se distribuem temporalmente as publicações sobre PC e SS no ensino de programação? PS2: Quais contextos educacionais (níveis de ensino) têm sido investigados? PS3: Quais abordagens pedagógicas, ferramentas e fundamentos teóricos têm sido empregados? PS4: Quais competências de PC e SS são mais frequentemente abordadas? PS5: Quais desafios professores e estudantes enfrentam no desenvolvimento de PC e SS?

3.2.1 PS1: distribuição temporal das publicações

A Figura 10 apresenta a distribuição das publicações por ano entre 2020 e 2024. Os anos de 2022 e 2023 concentram o maior número de estudos, com cinco trabalhos cada, o que pode estar relacionado ao aumento do interesse por abordagens digitais e colaborativas observado após o período mais crítico da pandemia.

Figura 10: Distribuição das publicações por ano



Nota: as buscas foram encerradas entre 12 e 14 de outubro de 2024; publicações de novembro e dezembro de 2024 não foram contabilizadas.

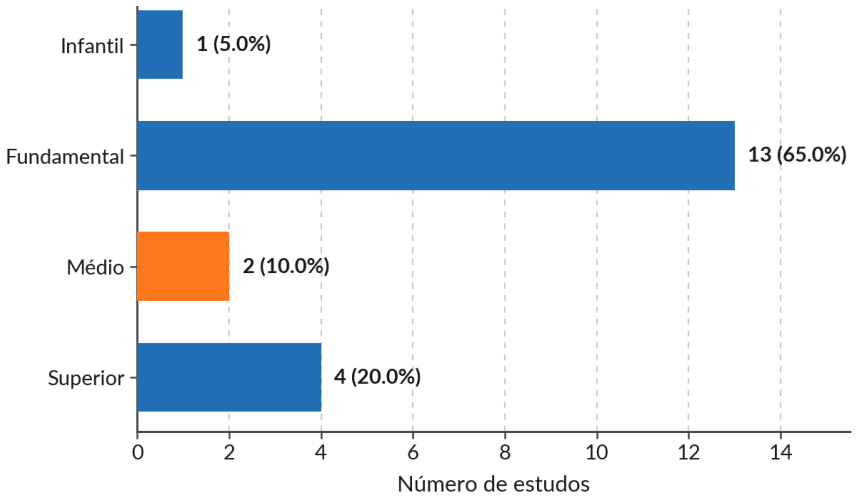
Fonte: Autoria própria.

No ano de 2024, foram identificados três estudos. Esse número, inferior ao registrado em 2022 e 2023, deve ser interpretado à luz da janela temporal das buscas, conduzidas entre 12 e 14 de outubro de 2024, conforme descrito no Apêndice A. Publicações disponibilizadas nos meses de novembro e dezembro de 2024 não foram contabilizadas neste mapeamento, o que limita a comparação direta com os anos anteriores. Ainda assim, a concentração de publicações entre 2022 e 2023 sugere uma intensificação recente das pesquisas sobre o tema.

3.2.2 PS2: contextos educacionais investigados

A Figura 11 apresenta a distribuição dos estudos por nível de ensino. O ensino fundamental concentra a maior parte dos trabalhos (65,0%), seguido pelo ensino superior (20,0%), pelo ensino médio (10,0%) e pela educação infantil (5,0%). Nenhum dos 18 estudos selecionados investigou especificamente o ensino médio técnico, lacuna que se relaciona diretamente com o problema de pesquisa apresentado no Capítulo 1.

Figura 11: Distribuição dos estudos por nível de ensino



Nota: A soma das ocorrências (20) supera o total de estudos (18) porque E04 e E14 investigam mais de um nível de ensino, sendo contabilizados em cada um deles.

Fonte: Autoria própria.

3.2.3 PS3: abordagens, ferramentas e base teórica

Esta questão busca identificar as abordagens utilizadas para promover o desenvolvimento do PC e das SS no ensino de programação. A análise mostrou que os elementos descritos pelos autores abrangem estratégias pedagógicas, ferramentas tecnológicas e fundamentos teóricos. Desse modo, a Tabela 3 organiza esses elementos em três blocos: (i) Abordagens

Pedagógicas, com estratégias didáticas e metodológicas; (ii) Ferramentas, Plataformas e Tecnologias, com os recursos digitais e físicos utilizados; e (iii) Base Teórica, com os referenciais adotados nos estudos. Em cada bloco, os estudos foram organizados conforme os níveis de ensino investigados.

Tabela 3: Abordagens, ferramentas e fundamentação teórica dos estudos

Categoria	**Infantil**	**Fundamental**	**Médio**	**Superior**
Bloco 1 – Abordagens Pedagógicas				
Aprendizagem Autodirigida			E04	E04
Aprendizagem Baseada em Problemas		E08		
Aprendizagem Baseada em Projetos		E02, E06, E08		E11, E17
Aprendizagem Experiencial		E10		E18
Jogos	–	E03, E06, E07, E08, E10, E12		
Metodologias Ágeis			E04	E04
Programação em Pares		E07, E13	E04	E04, E18
Scaffolding		E01		
Storytelling		E05		
Bloco 2 – Ferramentas, Plataformas e Tecnologias				
Kniwwelino		E14	E14	
KoduGameLab		E01, E03		
Robótica Educacional	E16	E02, E05, E06, E07, E12, E15		E11
Scratch/ ScratchJr	E16	E02, E03, E05, E06, E08, E09, E13, E15		
Bloco 3 – Base Teórica				
Construcionismo (Papert)		E03, E05		

Fonte: Autoria própria.

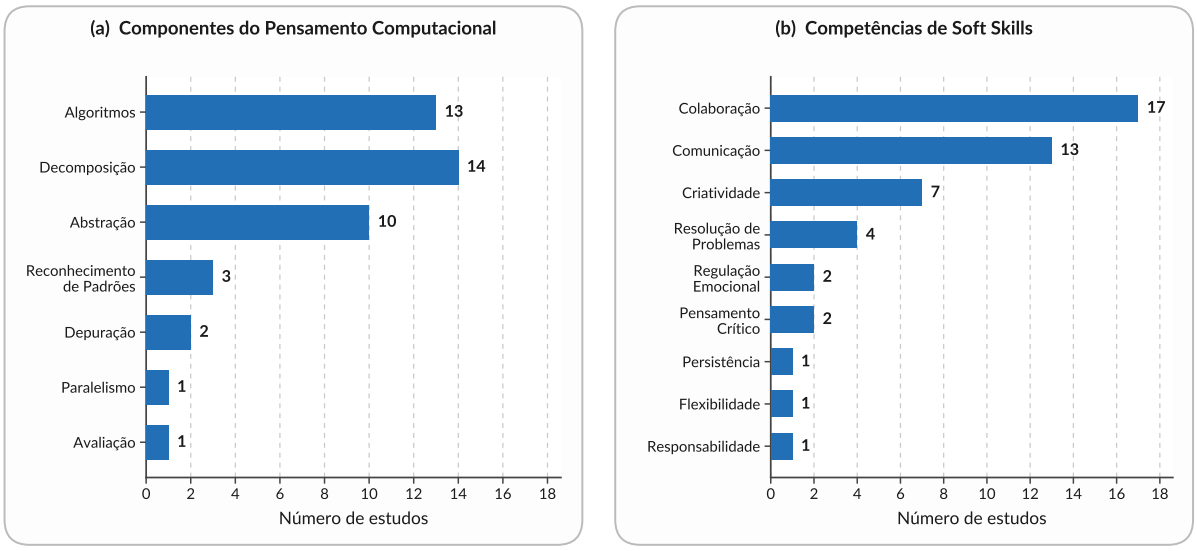
No ensino fundamental, observa-se diversidade de abordagens e ferramentas, com Aprendizagem Baseada em Projetos (E02, E06, E08), Jogos (E03, E06, E07, E08, E10, E12), Programação em Pares (E07, E13) e Robótica Educacional (E02, E05, E06, E07, E12, E15) entre as mais recorrentes. No ensino superior, destacam-se Programação em Pares (E04, E18), Aprendizagem Baseada em Projetos (E11, E17), Aprendizagem Experiencial (E18) e Robótica Educacional (E11), abordagens voltadas à autonomia e à aplicação prática de conceitos teóricos (Wang; Herzog, 2023; Jalinus; Putra *et al.*, 2024; Kalluri *et al.*, 2024). O Construcionismo de Papert aparece como fundamentação teórica em apenas dois estudos (E03 e E05), embora ferramentas de natureza construcionista, como Scratch e robótica educacional, sejam recorrentes entre os 18 trabalhos.

3.2.4 PS4: componentes do PC e competências de SS mais abordados

A Figura 12 apresenta a distribuição dos componentes do PC e das SS mais citados nos estudos. No campo do PC, os componentes mais citados são decomposição (14 estudos),

algoritmos (13) e abstração (10), seguidos de reconhecimento de padrões (3), que completa o conjunto dos quatro componentes, e de componentes menos recorrentes, como depuração (2), paralelismo (1) e avaliação (1). No campo das SS, as mais frequentes são colaboração (17 estudos, considerando trabalho em equipe como equivalente), comunicação (13) e criatividade (7), seguidas de resolução de problemas (4), regulação emocional (2) e pensamento crítico (2), entre outras. Essa distribuição converge para os componentes do PC tratados na Seção 2.3 e para o modelo 4C discutido na Seção 2.4, ainda que reconhecimento de padrões e pensamento crítico apareçam com frequência menor nos estudos empíricos do que no referencial teórico.

Figura 12: Frequência dos componentes de PC e SS



Nota: Colaboração inclui Trabalho em Equipe, tratados como equivalentes neste estudo, conforme o Capítulo 2.

Fonte: Autoria própria.

3.2.5 PS5: desafios reportados na literatura

Embora os estudos apresentem recorrência em determinados componentes do PC e em determinadas competências de SS, a implementação dessas propostas também enfrenta obstáculos pedagógicos e estruturais presentes em diferentes contextos.

Os desafios identificados podem ser agrupados em quatro categorias principais. A primeira envolve a falta de formação e preparo dos professores (E02, E08, E17), com destaque para a dificuldade de integrar PC e SS de forma conjunta nas práticas docentes. A segunda está relacionada à falta de infraestrutura tecnológica adequada e de recursos materiais (E05, E08, E12, E15), particularmente em contextos de escolas públicas ou regiões periféricas.

A terceira abrange dificuldades cognitivas dos estudantes com abstração e programação (E01, E06, E11), o que exige estratégias específicas de *scaffolding* e acompanhamento. A quarta categoria trata da integração entre componentes do PC e as SS (E04, E13, E16), em que se observa uma divisão entre o foco em habilidades técnicas e o desenvolvimento de SS. É relevante notar que três dessas quatro categorias dizem respeito a condições do professor e da escola, enquanto apenas uma se relaciona diretamente ao estudante, o que sugere um deslocamento do problema: a literatura tem reconhecido que a principal dificuldade não está nos alunos, mas nas condições oferecidas para que professores conduzam essa integração.

3.2.6 Discussão dos resultados do MSL

Os resultados das cinco questões secundárias permitem responder à questão principal que orientou esta revisão. A literatura recente aborda PC e SS no ensino de programação por meio de estratégias colaborativas, como programação em pares, robótica educacional e aprendizagem baseada em projetos, cuja natureza favorece, ainda que sem intencionalidade pedagógica explícita, o desenvolvimento das SS. A integração entre PC e SS aparece, assim, mais como consequência das escolhas didáticas do que como objetivo pedagógico estruturado. Três pontos sustentam essa leitura.

O primeiro ponto trata do modo como PC e SS são abordados nos objetivos dos estudos. Na maioria dos trabalhos analisados, o PC aparece como objetivo principal, enquanto as SS surgem como ganho secundário observado pelos pesquisadores durante a aplicação das atividades. Estudos como E04, que propõe o modelo de *Cooperative Thinking*, e E09, que projeta um *workshop* voltado simultaneamente ao PC e à competência interpessoal, são exceções em que a integração entre PC e SS está presente desde o planejamento da intervenção.

O segundo ponto refere-se à base teórica. Ferramentas e estratégias de inspiração construcionista, como Scratch, robótica educacional e dispositivos tangíveis, são recorrentes nos estudos, mas apenas E03 e E05 ancoram explicitamente suas propostas no Construcionismo de Papert. A recorrência das ferramentas sem o respectivo amparo teórico sugere que o construcionismo opera nos estudos como repertório técnico, não como fundamento pedagógico, o que reduz a possibilidade de relações mais profundas entre as habilidades técnicas e as SS que o referencial originalmente propõe.

O terceiro ponto diz respeito ao deslocamento da dificuldade. Três das quatro categorias de desafios reportadas pelos estudos (PS5) remetem a condições alheias ao aprendiz,

à formação docente, à infraestrutura tecnológica e à própria dificuldade de integrar PC e SS nas práticas pedagógicas, enquanto apenas uma se refere diretamente aos estudantes. A literatura analisada reconhece, portanto, que o gargalo da integração entre PC e SS está menos no aprendiz e mais nas condições oferecidas ao professor para conduzir essa integração de forma planejada.

Consideradas em conjunto, essas três observações apontam para uma mesma direção. A aproximação entre PC e SS existe na prática das salas de aula investigadas, mas carece de intencionalidade pedagógica, de ancoragem teórica explícita e de apoio pedagógico sistematizado ao professor. Esse diagnóstico orienta tanto o recorte empírico quanto a proposta conceitual desenvolvidos nos capítulos seguintes.

3.3 Considerações finais do capítulo

Este capítulo apresentou um mapeamento da literatura sobre o desenvolvimento integrado de PC e SS no ensino de programação, conduzido por meio de um MSL que selecionou 18 estudos publicados entre 2020 e 2024. A análise das cinco questões secundárias permitiu caracterizar a distribuição temporal e contextual das pesquisas, as abordagens e ferramentas mais empregadas, as competências mais recorrentes e os desafios reportados.

Os resultados deste mapeamento dialogam com o problema de pesquisa apresentado no Capítulo 1, no qual a EPTNM foi identificada como contexto em que a integração entre formação geral e formação profissional impõe exigências específicas ao ensino de programação. A ausência de estudos voltados a essa etapa, revelada pelo MSL, confirma a pertinência do recorte adotado e reforça a relevância da investigação proposta.

Os achados também se relacionam com os fundamentos teóricos discutidos no Capítulo 2. Os componentes do PC tratados na Seção 2.3, ou seja, decomposição, reconhecimento de padrões, abstração e algoritmos, aparecem como componentes centrais nos estudos analisados, com destaque para decomposição e algoritmos e com reconhecimento de padrões em menor recorrência, ainda que consolidado entre os componentes. No campo das SS, as competências mais frequentes como colaboração, comunicação e criatividade, convergem com o modelo 4C discutido na Seção 2.4. O pensamento crítico, quarta competência do 4C, aparece com menor frequência nos estudos empíricos, o que reforça o peso da base teórica na sua incorporação ao *framework*. Já o Construcionismo de Papert, adotado nesta tese como uma das bases da proposta, aparece de forma pouco explícita nos estudos selecionados, o que indica que a tradição construcionista permanece presente nas

ferramentas, mas ausente como referencial declarado, abrindo espaço para uma retomada teoricamente fundamentada.

O MSL revelou, ainda, uma lacuna principal e três lacunas complementares. A principal refere-se à ausência de estudos voltados ao ensino médio técnico. Essa etapa, em que se combinam formação geral e formação profissional, impõe condições específicas ao ensino de programação que não são contempladas pelas pesquisas analisadas, concentradas no ensino fundamental.

Mesmo com a inclusão de cinco bases internacionais (ACM Digital Library, IEEE Digital Library, ERIC, ISI Web of Science e Scopus) e da Biblioteca Digital da SBC (SOL-SBC), nenhum dos 18 trabalhos selecionados aborda especificamente o ensino médio técnico. Esse cenário reforça o recorte empírico desta tese e justifica a investigação conduzida nos capítulos seguintes.

As lacunas complementares permitem compreender limitações ainda presentes na literatura e ampliam o escopo da contribuição pretendida:

(i) escassez de investigações na educação infantil, etapa em que o desenvolvimento conjunto de PC e SS permanece pouco explorado; (ii) integração incipiente entre PC e SS, com ênfase nos aspectos técnicos do PC e tratamento das SS como efeito secundário, mesmo quando ambas figuram nos objetivos dos estudos; (iii) carência de diretrizes sistematizadas para a atuação docente, aspecto particularmente relevante, porque os próprios estudos reconhecem que parte significativa das dificuldades reside nas condições oferecidas ao professor.

É a partir dessas lacunas que se organiza o espaço em que esta tese se insere e que se orientam os capítulos seguintes. O Capítulo 4 responde à lacuna principal ao investigar as percepções de professores da área de computação dos IFs brasileiros, contexto representativo da EPTNM. O Capítulo 5 enfrenta as lacunas (ii) e (iii) ao apresentar o *framework* SoPensa, uma proposta conceitual e operacional voltada ao desenvolvimento integrado do PC e das SS, com apoio ao planejamento docente. O Capítulo 6, por sua vez, examina a aplicabilidade da proposta em contexto real de ensino.

O mapeamento apresenta uma limitação relacionada ao recorte temporal. O intervalo de cinco anos, embora justificado pelos marcos normativos brasileiros (Resolução CNE/CP nº 2/2017; Parecer CNE/CEB nº 2/2022; Resolução CNE/CEB nº 1/2022) (Brasil, 2017a, 2022), excluiu estudos anteriores a 2020 que poderiam ampliar a análise. Essa limitação pode ser ampliada em pesquisas futuras que estendam o intervalo investigado.

4 ESTUDO EMPÍRICO COM PROFESSORES DOS INSTITUTOS FEDERAIS

Este capítulo apresenta o estudo empírico voltado a investigar as percepções de professores dos IFs sobre o uso do PC e das SS no ensino de programação. A partir dessa investigação, busca-se identificar os componentes do PC e as competências de SS considerados mais adequados ao desenvolvimento dos alunos. A coleta de dados foi organizada em duas etapas: aplicação de questionário e realização de entrevistas. O questionário teve como finalidade captar a percepção dos professores sobre os fatores que influenciam o ensino de programação, enquanto as entrevistas aprofundaram a análise dos componentes do PC e das competências de SS mais relevantes para a formação dos alunos. O conjunto de dados coletados serviu como subsídio para a elaboração das diretrizes do *framework* proposto nesta tese.

4.1 Procedimentos metodológicos

Os procedimentos descritos nesta seção operacionalizam a fase de Diagnóstico do ciclo de pesquisa-ação apresentado na Seção 1.3, detalhando o contexto, os participantes, os critérios de seleção e os instrumentos utilizados no estudo empírico com os professores.

4.1.1 Contexto

Os IFs foram selecionados por estarem inseridos na educação básica e por atuarem na formação de profissionais para o mercado de trabalho na área técnica em computação. A escolha torna pertinente a investigação sobre PC e SS, dada a relevância de promover essas habilidades, fundamentais para os profissionais do século XXI (Brasil, 2021, 2022; SBC, 2025). A coleta de dados respeitou os princípios de confidencialidade, anonimato e participação voluntária dos professores. Os dados foram utilizados exclusivamente para fins acadêmicos e para a melhoria do ensino de programação.

4.1.2 Participantes

Os participantes deste estudo foram professores de Informática dos IFs de todo o Brasil. Os 38 IFs foram convidados a participar da pesquisa. O convite foi enviado por *e-mail* às coordenações de curso, para que o encaminhassem aos docentes, buscando favorecer o alcance nacional e a diversidade dos perfis dos participantes.

No total, 133 professores responderam ao questionário, com representantes das cinco regiões do país. Desses, 13 professores foram selecionados para as entrevistas, distribuídos da seguinte forma: 8 da região Norte, 2 do Nordeste, 1 do Sudeste, 1 do Sul e 1 do Centro-Oeste. A predominância de participantes da região Norte deve-se à maior proximidade do pesquisador com os IFs dessa região, o que facilitou o contato e a mobilização dos docentes.

A seleção dos entrevistados buscou contemplar, na medida do possível, a diversidade regional, as formações acadêmicas, as disciplinas lecionadas e o sexo dos participantes, resultando em 11 homens e 2 mulheres. Quanto às experiências, a seleção não se restringiu à Lógica de Programação, reunindo professores que lecionam Engenharia de Software, Banco de Dados, Programação Web e Design de Interfaces. Essa diversidade ampliou o conjunto de percepções sobre o ensino de programação na área de Informática.

4.1.3 Critérios de inclusão e exclusão

Os critérios de inclusão para o questionário foram possuir formação em cursos da área de Informática, o que assegura familiaridade com o contexto da pesquisa, e aceitar os termos de participação voluntária. Para as entrevistas, os critérios de inclusão foram atuar como professor de Informática em cursos técnicos integrados ao ensino médio, garantindo relação direta com o foco da pesquisa; possuir mais de três anos de experiência na instituição, critério estabelecido para selecionar profissionais com vivência suficiente para fornecer informações relevantes sobre os desafios e práticas no ensino de programação; e concordar com os termos de participação voluntária. Participantes que não atenderam a esses critérios foram excluídos.

4.2 Método de coleta de dados

O processo metodológico foi estruturado em etapas sequenciais, contemplando a elaboração dos instrumentos de coleta, o contato com as coordenações dos cursos de Informática

dos IFs, a aplicação do questionário, a seleção dos participantes para as entrevistas com base nos perfis identificados, a realização das entrevistas e a análise dos dados. A Figura 13 apresenta o fluxo metodológico do estudo e a sequência dessas etapas.

Figura 13: Etapas metodológicas do estudo empírico

Fonte: Autoria própria.

4.2.1 Construção e aplicação do questionário

A primeira fase da coleta de dados consistiu na aplicação de um questionário eletrônico autoaplicado, elaborado no Google *Forms*. O instrumento teve como finalidade captar a percepção docente sobre os fatores que influenciam o ensino de programação. Ao todo, o questionário foi composto por 18 questões, distribuídas em três seções: perfil demográfico, perfil acadêmico e profissional, e percepções e desafios no ensino de programação. Essa estrutura é apresentada na Tabela 4.

Tabela 4: Estrutura do questionário aplicado aos professores

Seção	**Questões**	**Tipo**	**Objetivo**
Perfil demográfico	2	Múltipla escolha	Caracterizar os participantes quanto ao sexo e à região de atuação.
Perfil acadêmico e profissional	5	Múltipla escolha	Coletar dados sobre formação acadêmica, áreas de atuação e tempo de experiência.
Percepções e desafios no ensino de programação	11	Escala *Likert* e questão aberta	Entender como os professores percebem os desafios, as oportunidades e as condições que influenciam a aprendizagem de programação.
Total	18	7 de múltipla escolha, 10 em escala *Likert* e 1 aberta	

Fonte: Autoria própria.

Seguindo as diretrizes de Kitchenham e Pfleeger (2008) para a construção e a validação de questionários autoaplicados, o instrumento foi submetido, antes da aplicação definitiva, a um estudo de validação com cinco professores especialistas de diferentes IFs

e contextos de atuação. Na primeira rodada de avaliação, dois professores apontaram que duas questões apresentavam duplicidade, o que resultou em ajustes para evitar ambiguidades. Na segunda rodada, um professor sugeriu a exclusão de uma questão redundante, com o objetivo de otimizar a estrutura do instrumento. Após esses ajustes, a terceira versão foi considerada adequada e disponibilizada aos participantes por 20 dias, entre novembro e dezembro de 2024. A versão final pode ser consultada no Apêndice C.

4.2.2 Entrevistas semiestruturadas

Na segunda etapa, foram realizadas entrevistas semiestruturadas com 13 professores. Segundo Mattar e Ramos (2021), essa técnica combina um roteiro prévio com flexibilidade para desdobramentos durante o diálogo, permitindo aprofundar a análise de PC e SS no ensino de programação. O roteiro foi ajustado em três versões a partir de um estudo de validação com dois professores, sendo um participante presencial e outro *on-line*. A primeira versão foi reformulada para aprimorar a sequência e a coerência das perguntas; a segunda foi adaptada ao formato *on-line*, com simplificação de questões e inclusão de instruções mais detalhadas; e a versão final consolidou essas melhorias para aplicação nos formatos presencial e *on-line*.

Todas as entrevistas foram gravadas em áudio e transcritas com o consentimento prévio dos professores, assegurando a confidencialidade e o cumprimento das normas éticas. Os contatos com os participantes foram estabelecidos por *e-mail*. A Tabela 5 apresenta um resumo das principais perguntas abordadas durante as entrevistas. O roteiro completo encontra-se disponível no Apêndice C.

Tabela 5: Principais categorias do roteiro de entrevistas

Categoria	**Descrição**	**Exemplo de Pergunta**
Percepção sobre SS	Perguntas sobre como os professores percebem as SS no ensino de programação.	Você conhece o conceito de SS? Se sim, como você o definiria?
Percepção sobre os componentes de PC	Aborda o uso de conceitos-chave, como abstração, reconhecimento de padrões, algoritmos e decomposição de problemas, no ensino de programação.	Você já utilizou componentes de PC, como decomposição, abstração, algoritmos ou reconhecimento de padrões, no ensino de programação? Caso sim, esse uso foi de forma ad hoc ou sistematizada? Poderia compartilhar brevemente sua experiência?
Sugestões para implementação do *framework*	Propõe ideias para promover o PC e SS através de um *framework*.	Você percebe alguma relação entre o desenvolvimento de PC e SS no ensino de programação? Caso sim, como essas dimensões podem se complementar?

Fonte: Autoria própria.

As categorias apresentadas na Tabela 5 foram organizadas a partir de aspectos relacionados ao uso de PC e SS no ensino de programação. Elas abrangem desde a percepção dos professores sobre essas dimensões até sugestões para incorporá-las às práticas pedagógicas. Durante as entrevistas, constatou-se que alguns participantes não estavam familiarizados com os termos *'soft skills'* e 'pensamento computacional', embora reconhecessem palavras associadas, como trabalho em equipe e abstração. Esse resultado indica que algumas práticas relacionadas ao PC e às SS já estavam presentes nas experiências docentes, embora nem sempre fossem reconhecidas ou sistematizadas pelos professores a partir desses conceitos. Assim, a ausência de familiaridade com os termos não significava necessariamente a ausência das práticas, mas revelava uma distância entre práticas já realizadas e sua identificação e organização conceitual.

Para garantir uma compreensão uniforme e evitar interpretações equivocadas, o entrevistador forneceu uma explicação breve e objetiva desses conceitos aos participantes que declararam desconhecê-los. Essa explicação teve como único propósito esclarecer os termos, sem influenciar as respostas, permitindo que os participantes refletissem com base em suas próprias experiências e percepções. Procedimento equivalente não foi adotado para o termo *framework*, frequentemente associado, na formação em Computação, a *frameworks* de desenvolvimento de *software*, com sentido distinto do conceito operacional adotado nesta tese. Parte das respostas do terceiro eixo do roteiro pode, portanto, ter sido formulada a partir dessa outra acepção, aspecto considerado na interpretação dos resultados correspondentes.

4.3 Análise dos dados

Esta seção apresenta a análise dos dados coletados por meio do questionário e das entrevistas com professores dos IFs. A análise do questionário permitiu identificar tendências gerais e subsidiou a seleção dos professores para as entrevistas. As entrevistas complementaram os dados quantitativos, ao aprofundar de forma qualitativa as questões levantadas sobre PC e SS no ensino de programação.

A análise dos dados teve caráter descritivo e exploratório. As respostas do questionário foram tratadas por estatística descritiva, com apuração de frequências e percentuais. As respostas abertas e as entrevistas foram organizadas por categorização direta, com base nos princípios da análise de conteúdo qualitativa (Flick, 2013; Mattar; Ramos, 2021). Entre as categorias estão o nível de conhecimento sobre cada conceito de PC e de SS e

a forma de uso relatada, classificada como informal ou sistematizada. Essas categorias orientaram a síntese e a comparação das percepções dos professores, sem recorrer à construção indutiva de categorias a partir dos próprios dados. Assim, neste capítulo, não se aplicou a ATR proposta por Braun e Clarke (2022). A ATR, com apoio do Atlas.ti, foi reservada à avaliação da aplicação do *framework*, apresentada no Capítulo 7.

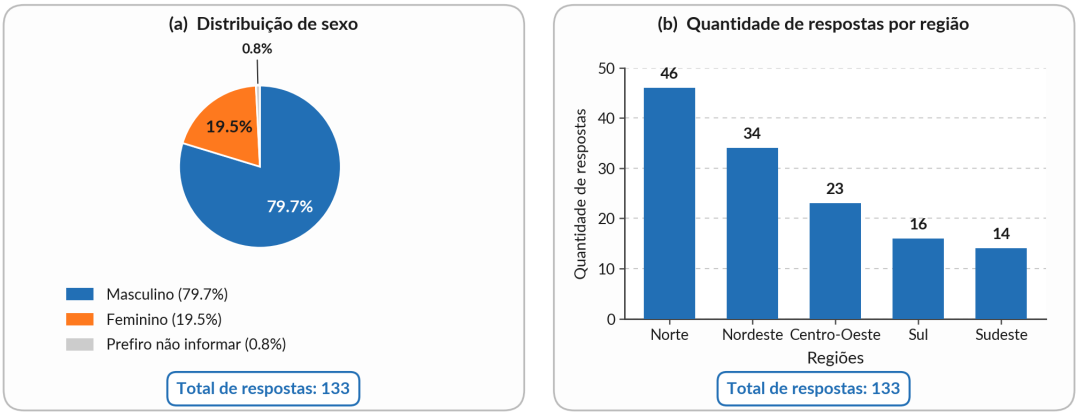
4.3.1 Análise das respostas do questionário

A seguir, são analisadas as respostas do questionário, com foco nos principais desafios e nas sugestões apresentadas pelos docentes.

4.3.1.1 Perfil demográfico dos participantes

A Figura 14a apresenta a distribuição dos participantes por sexo. Dos 133 professores que responderam ao questionário, 106 (79,7%) são do sexo masculino, 26 (19,5%) do feminino e 1 (0,8%) preferiu não informar. Quanto à localização dos IFs em que os respondentes trabalham, a Figura 14b mostra que 46 (34,6%) estão na região Norte, 34 (25,6%) no Nordeste, 23 (17,3%) no Centro-Oeste, 16 (12,0%) no Sul e 14 (10,5%) no Sudeste.

Figura 14: Distribuição dos participantes por sexo e região



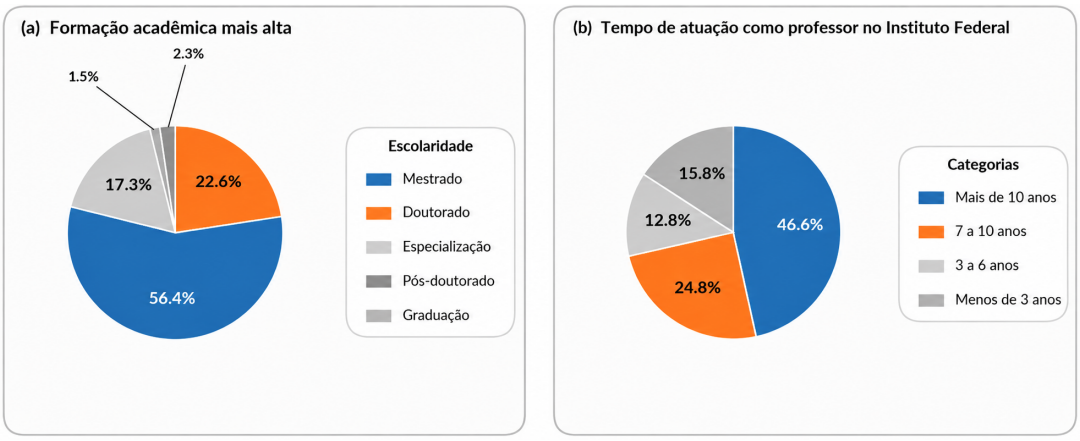
Fonte: Autoria própria.

4.3.1.2 Perfil acadêmico e profissional

No que se refere ao perfil acadêmico dos professores, a Figura 15a mostra que 75 (56,4%) possuem mestrado, seguidos por 30 (22,6%) com doutorado, 23 (17,3%) com especialização, 3 (2,3%) com pós-doutorado e apenas 2 (1,5%) com graduação. Já a Figura 15b apresenta a distribuição dos professores por tempo de atuação nos IFs. A maior parte dos

participantes tem mais de 10 anos de experiência, 62 (46,6%), seguida por aqueles com 7 a 10 anos, 33 (24,8%), menos de 3 anos 21 (15,8%) e, por fim, de 3 a 6 anos 17 (12,8%).

Figura 15: Perfil acadêmico e profissional



Fonte: Autoria própria.

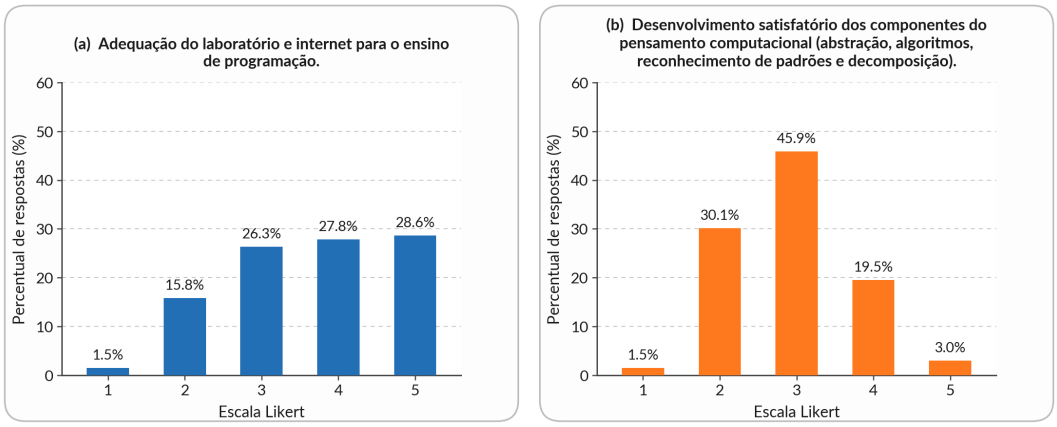
4.3.1.3 Percepções e desafios no ensino de programação

Para investigar os fatores que influenciam o ensino de programação, os professores responderam a uma questão geral composta por afirmativas específicas. Utilizou-se uma escala *Likert* de 1 a 5, em que 1 indicava "discordo totalmente" e 5, "concordo totalmente", para captar as percepções docentes sobre os desafios e as oportunidades no ensino de programação.

1. Infraestrutura e componentes do PC

A Figura 16a apresenta a percepção dos professores sobre a adequação dos laboratórios de informática e da internet para o ensino de programação. A maioria dos respondentes avaliou esse aspecto positivamente, com 37 (27,8%) atribuindo nota 4 e 38 (28,6%) atribuindo nota 5, totalizando mais de 56% das respostas. Outros 35 (26,3%) atribuíram nota 3, indicando uma percepção neutra ou moderada. As notas 1 e 2, que refletem insatisfação, foram atribuídas por 23 (17,3%) participantes. Esses dados sugerem que a infraestrutura é, em geral, considerada adequada, embora parte dos docentes ainda identifique limitações.

Figura 16: Infraestrutura e componentes do PC



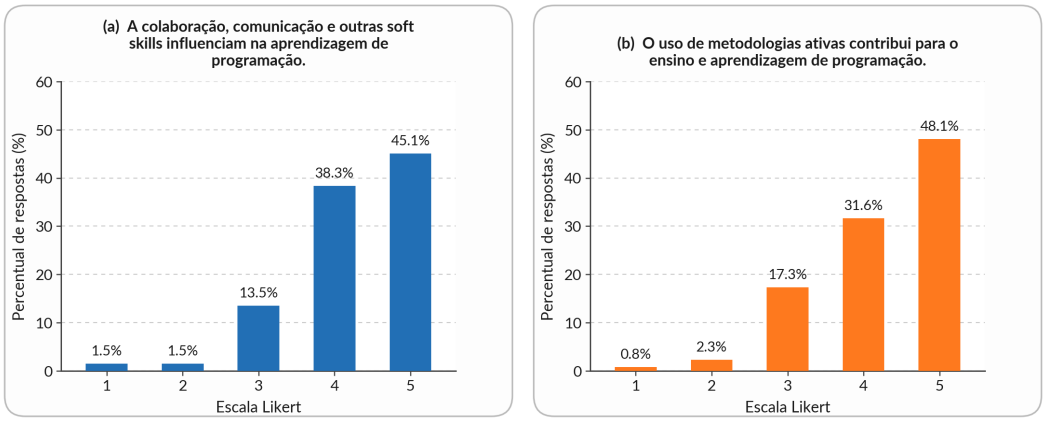
Fonte: Autoria própria.

A Figura 16b apresenta a percepção dos professores sobre o desenvolvimento dos componentes do PC. Os resultados mostram que 61 (45,9%) participantes atribuíram nota 3, indicando um desenvolvimento mediano. Por outro lado, 40 (30,1%) atribuíram nota 2, indicando limitações no desenvolvimento. As notas mais altas, como 4, atribuída por 26 (19,5%) participantes, e 5, por 4 (3,0%), foram indicadas por uma minoria. Apenas 2 (1,5%) atribuíram nota 1, correspondente ao menor nível da escala.

2. Competências de SS e metodologias ativas

No que se refere às SS e às metodologias ativas, os professores foram questionados sobre a influência das SS na aprendizagem de programação. A Figura 17a revela que o maior grupo, 60 participantes (45,1%), concordou totalmente com a afirmativa, enquanto 51 (38,3%) concordaram. Apenas 18 (13,5%) se mantiveram neutros, e a discordância foi baixa, com 4 participantes (3,0%) nas opções 1 e 2, sendo 2 (1,5%) em cada uma.

Figura 17: Competências de SS e metodologias ativas



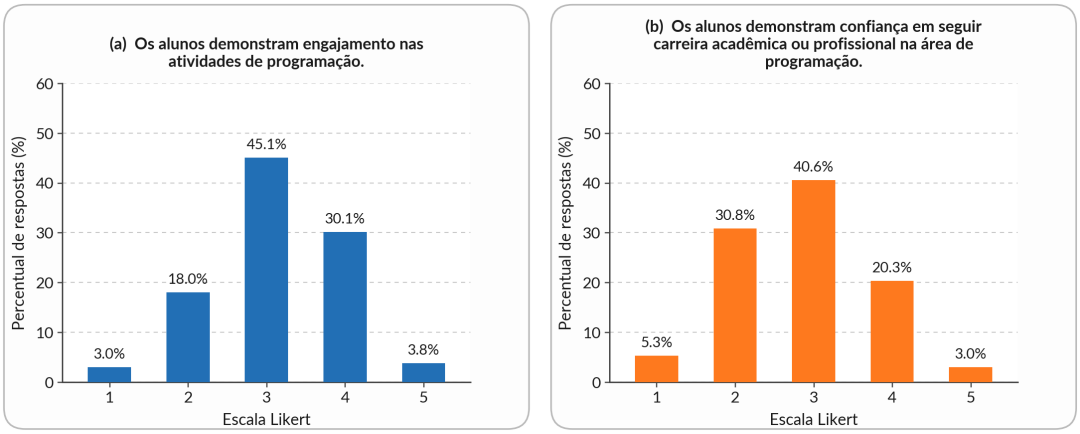
Fonte: Autoria própria.

Em relação à Figura 17b, observa-se que a maioria dos professores concorda que o uso de metodologias ativas contribui para o ensino e a aprendizagem de programação. Na análise das respostas, 64 (48,1%) participantes concordaram totalmente com a afirmativa e 42 (31,6%) concordaram. Apenas uma pequena parcela discordou parcialmente, 3 (2,3%), ou discordou totalmente, 1 (0,8%), enquanto 23 (17,3%) optaram pela posição neutra.

3. Engajamento e seguir carreira na área de programação

A Figura 18a apresenta a percepção dos professores sobre o engajamento dos alunos nas atividades de programação. A maior parte dos respondentes assinalou a opção neutra, com 60 respostas (45,1%), enquanto 40 (30,1%) concordaram parcialmente e 5 (3,8%) concordaram totalmente. Por outro lado, 24 (18,0%) discordaram parcialmente e 4 (3,0%) discordaram totalmente. Os resultados mostram predomínio da posição neutra, com leve inclinação para a concordância, representada por 33,8% nas notas 4 e 5, em comparação à discordância, representada por 21,1% nas notas 1 e 2. A predominância de respostas neutras indica que os professores não apresentaram uma percepção claramente definida sobre o engajamento dos alunos, constituindo um resultado relevante, pois revela dificuldade em identificar uma tendência clara de engajamento nas atividades de programação.

Figura 18: Engajamento e seguir carreira na área



Fonte: Autoria própria.

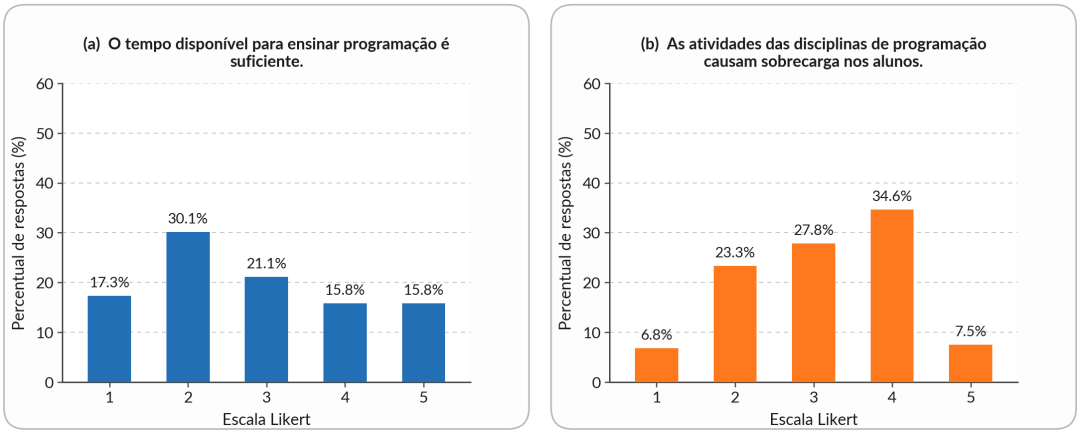
A Figura 18b apresenta a percepção dos professores sobre a confiança dos alunos em seguir carreira acadêmica ou profissional na área de programação. O maior grupo de respondentes assinalou a opção neutra, com 54 respostas (40,6%), seguida de 41 professores (30,8%) com discordância parcial. Por outro lado, 27 participantes (20,3%) concordaram parcialmente, enquanto 4 (3,0%) concordaram totalmente e 7 (5,3%) discordaram totalmente. Os dados refletem uma percepção predominantemente neutra ou de leve discordância, sugerindo que há incertezas ou desafios relacionados à motivação e ao preparo

dos alunos para construírem confiança em suas trajetórias na área.

4.Condições de ensino relacionadas ao tempo e à sobrecarga dos alunos

No que diz respeito ao tempo disponível e à carga de atividades, a Figura 19a apresenta a percepção dos professores sobre a suficiência do tempo para ensinar programação. Nesse aspecto, predomina a discordância, com 63 professores (47,4%) nas notas 1 e 2, contra 42 (31,6%) nas notas 4 e 5 e 28 (21,1%) em posição neutra. A percepção majoritária é de que o tempo é insuficiente. Já a Figura 19b apresenta a percepção sobre a sobrecarga causada nos alunos pelas atividades das disciplinas de programação. Nesse caso, a tendência se inverte para a concordância, com 56 professores (42,1%) nas notas 4 e 5, ante 40 (30,1%) nas notas 1 e 2 e 37 (27,8%) neutros. Os dois resultados apontam para tensões nas condições de ensino: o tempo que os professores consideram insuficiente e a sobrecarga percebida sobre os alunos.

Figura 19: Condições de ensino relacionadas ao tempo e à sobrecarga dos alunos



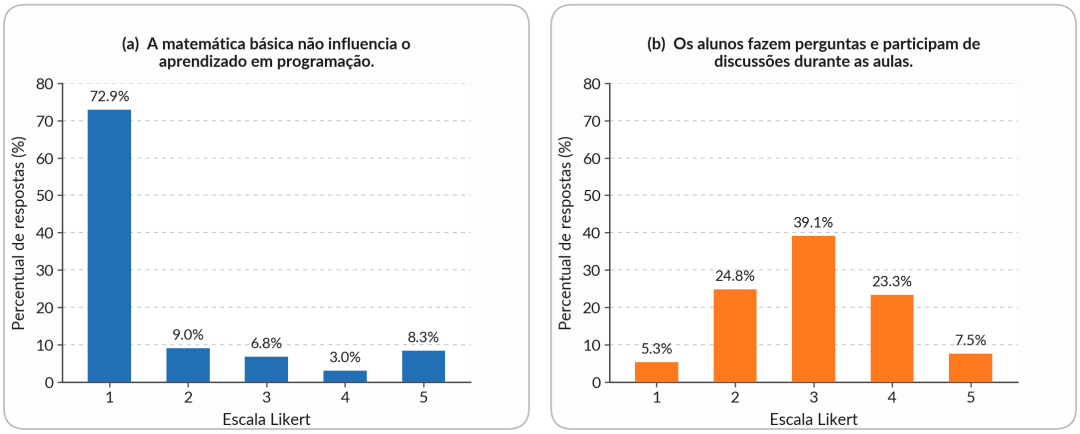
Fonte: Autoria própria.

5. Fatores relacionados à aprendizagem matemática e à participação durante as aulas

A Figura 20a apresenta os resultados referentes à aprendizagem matemática e à participação dos alunos. A primeira afirmativa foi enunciada de forma negativa, ao afirmar que "a matemática básica não influencia o aprendizado em programação". A maioria discordou dessa afirmativa, com 109 professores (82,0%) nas notas 1 e 2, o que indica que esses docentes consideram que a matemática básica influencia a aprendizagem de programação. Apenas 15 (11,3%) concordaram com a ausência de influência e 9 (6,8%) ficaram neutros. Na Figura 20b, referente à participação dos alunos em perguntas e discussões nas aulas, o resultado ficou dividido, com 41 professores (30,8%) nas notas 4 e 5, 40 (30,1%) nas

notas 1 e 2 e 52 (39,1%) em posição neutra, sem predominância clara entre concordância e discordância.

Figura 20: Fatores relacionados à matemática e à participação nas aulas



Fonte: Autoria própria.

4.3.1.4 Análise das respostas abertas

As respostas abertas dos professores foram analisadas com base em sua relação com o tema da pesquisa. Essas falas são apresentadas de forma anônima, pois os respondentes do questionário não foram codificados individualmente, diferentemente dos professores entrevistados, que são identificados posteriormente pelos códigos P1 a P13. A seguir, apresenta-se uma síntese das percepções mais recorrentes e relevantes, organizadas em desafios identificados e sugestões de melhoria para o ensino de programação.

Os dados revelaram dificuldades didáticas e metodológicas no ensino de programação. Entre os desafios mais mencionados está a falta de base em lógica e abstração por parte dos alunos, conforme apontado por um professor: “o problema não é a linguagem, mas a lógica.” Essa dificuldade compromete a interpretação de problemas e a construção de algoritmos, com impacto direto na aprendizagem. Outro desafio recorrente foi a desmotivação dos alunos, que, segundo os professores, veem o curso técnico “apenas como uma obrigação.” Além disso, a ansiedade por soluções rápidas foi apontada como um fator que prejudica o desenvolvimento do pensamento algorítmico.

Para melhorar o ensino de programação, os professores sugeriram a ampliação da carga horária prática e uma maior aproximação com o mercado de trabalho. Um dos participantes apontou a necessidade de “mais oportunidades de estágio e projetos reais.”

Outro ponto mencionado foi a capacitação docente, considerada necessária para a

adoção de metodologias inovadoras no ensino de programação. Conforme um professor afirmou: "são necessários investimentos na formação dos professores."

Por fim, os professores defenderam o uso de abordagens que incentivem a resolução de problemas, a colaboração e a criatividade, tornando a aprendizagem mais envolvente e significativa. As percepções mostram a importância de estratégias que integrem PC e SS, promovendo um ensino de programação alinhado às necessidades do mercado e ao desenvolvimento dos estudantes.

4.3.2 Análise das entrevistas com os professores

As entrevistas com 13 professores dos IFs buscaram explorar suas percepções sobre o ensino de programação, com foco em PC e SS. As perguntas foram organizadas em três eixos principais: percepção sobre SS no ensino de programação, percepção sobre componentes de PC e sugestões para implementação do *framework*. Cada eixo é analisado a seguir, com a apresentação de categorias identificadas, tabelas e falas-chave que destacam os componentes mencionados, as dificuldades relatadas e as estratégias sugeridas. O objetivo é identificar as prioridades e contribuições dos professores para o desenvolvimento de uma abordagem alinhada ao contexto da EPTNM. Para preservar o anonimato, cada um dos 13 professores entrevistados foi identificado por um código sequencial, de P1 a P13, utilizado ao longo da análise.

4.3.2.1 Percepção sobre SS no ensino de programação

No contexto do ensino de programação, investiga-se o papel das SS como complemento ao desenvolvimento de habilidades técnicas e cognitivas. Para explorar o nível de familiaridade dos professores com esse conceito, foi formulada uma questão sobre o conhecimento docente a respeito das SS.

Pergunta: Você conhece o conceito de *soft skills*? Se sim, como você o definiria?

A questão teve como objetivo verificar se os professores possuíam conhecimento sobre o conceito de SS. Aos que declararam desconhecer o termo, aplicou-se a explicação descrita na Seção 4.2.2, fornecida após a resposta inicial que serviu de base para a classificação. As respostas foram categorizadas em três grupos principais: Conhecimento Completo, Conhecimento Parcial e Desconhecimento.

A classificação baseou-se na qualidade da definição apresentada por cada professor, e não na autoavaliação. Professores que declararam conhecer o conceito, mas o definiram

de forma incorreta ou imprecisa, foram enquadrados conforme o grau de aproximação da resposta. Foi o caso de P13, que afirmou conhecer o termo, mas o associou ao cotidiano da programação, definição distante do conceito de SS adotado neste estudo, ficando classificado em Desconhecimento. Os resultados são apresentados na Tabela 6.

Tabela 6: Nível de conhecimento sobre SS

Nível de Conhecimento	**Professores**	**Descrição**
Conhecimento Completo	P1, P6, P7, P10, P11	Definiram corretamente o conceito, citando habilidades interpessoais e comportamentais.
Conhecimento Parcial	P2, P9	Demonstraram familiaridade, mas com definições superficiais ou inseguras.
Desconhecimento	P3, P4, P5, P8, P12, P13	Não conheciam o termo, mas alguns reconheciam práticas relacionadas após explicação do conceito.

Fonte: Autoria própria.

Dos 13 professores entrevistados, 5 apresentaram conhecimento completo sobre o conceito, enquanto 2 demonstraram conhecimento parcial e 6 não conheciam o termo *soft skills*. Falas representativas foram destacadas para cada categoria. Na categoria Conhecimento Completo, P1 afirmou que "*soft skills* são habilidades comportamentais, adquiridas de modo transversal durante a vida acadêmica ou profissional." Já na categoria Desconhecimento, P12 comentou: "não, eu não conheço essa abordagem chamada *soft skills*."

Pergunta: Na sua opinião, as *soft skills* têm algum impacto no ensino de programação? Por quê?

Nesta questão, buscou-se investigar a percepção dos professores sobre o impacto das SS no ensino de programação. Todos os professores entrevistados afirmaram que essas competências possuem impacto positivo nesse contexto. Eles destacaram que as SS são relevantes para preparar os alunos para desafios reais do mercado de trabalho e melhorar a dinâmica de aprendizagem colaborativa. Conforme apontou P7, "programar, inicialmente, não é uma tarefa fácil para ninguém. Quanto mais as pessoas conseguem trabalhar e estudar em conjunto, melhor é o aprendizado. Acredito que o trabalho em equipe e a capacidade de se relacionar bem com os outros fazem muita diferença durante o processo de aprendizado."

P13 reforçou essa percepção ao afirmar que "Sabemos que a programação e o desenvolvimento de uma aplicação exigem colaboração. Existem várias metodologias ensinadas para promover o espírito de equipe e responsabilidade. Por isso, acredito que é de fundamental importância abordar *soft skills* no ensino de programação."

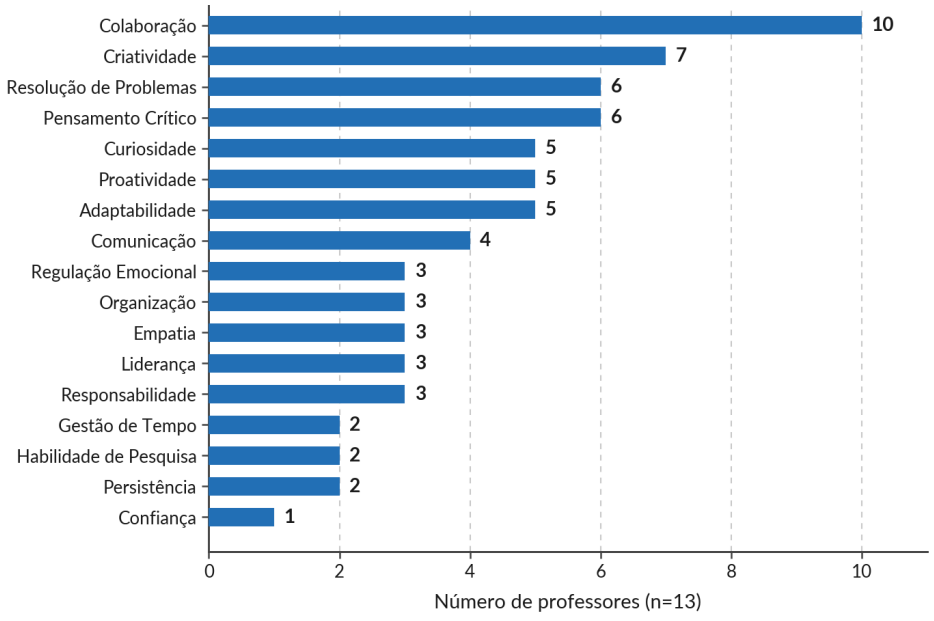
Pergunta: Quais *soft skills* você considera mais relacionadas ao ensino de programação? Selecione até cinco habilidades na lista abaixo e, se desejar, justifique sua escolha

com exemplos práticos.

Os professores foram convidados a identificar até cinco SS consideradas mais relevantes para o ensino de programação. Para facilitar suas escolhas, foi apresentada uma lista de 17 habilidades, selecionadas com base na literatura acadêmica sobre competências de SS no contexto educacional e profissional. A seleção foi fundamentada em autores como Matturro, Raschetti e Fontán (2019), Silva e Albuquerque (2023), WEF (2023), Nudin, Warsito e Wibowo (2022) e Chévez-Coronel *et al.* (2023). Além disso, os participantes tiveram a liberdade de incluir outras habilidades não contempladas, enriquecendo a análise com percepções específicas e exemplos práticos derivados de suas experiências.

A Figura 21 apresenta a frequência com que cada SS foi selecionada pelos 13 professores entrevistados, reunindo colaboração e trabalho em equipe como equivalentes. O comprimento de cada barra corresponde ao número de professores que indicaram a habilidade, registrando apenas a frequência de seleção, sem graduação de relevância.

Figura 21: Frequência das *Soft Skills* mais mencionadas pelos professores



Nota: neste estudo, "trabalho em equipe" está incluído na categoria "Colaboração".

Fonte: Autoria própria.

A distribuição mostra a colaboração como a habilidade mais selecionada (10), reunindo também o trabalho em equipe, seguida da criatividade (7) e de resolução de problemas e pensamento crítico (6 cada). Logo abaixo aparecem curiosidade, proatividade e adaptabilidade (5 cada), além da comunicação (4), enquanto as demais competências surgem com frequências menores. A definição de quais competências integram o *framework*, com a respectiva justificativa teórica e empírica, é apresentada na Tabela 9.

Pergunta: Você utiliza estratégias específicas para desenvolver *soft skills* em suas aulas? Caso utilize, poderia descrever se essas estratégias são aplicadas de forma mais informal (*ad hoc*) ou planejada (sistematizada)?

Com o objetivo de identificar se os professores desenvolvem SS de maneira informal (*ad hoc*) ou sistematizada, os entrevistados foram questionados sobre as práticas adotadas em suas aulas. A análise revelou que a maioria utiliza estratégias informais para promover essas habilidades. Entre as práticas mencionadas, estão trabalhos em equipe, dinâmicas práticas e o uso de exemplos baseados em experiências pessoais. Conforme P9 destacou, "passo trabalhos em grupo para estimular habilidades como comunicação e colaboração, mas nem sempre os alunos engajam." P5 complementou, "incorporo as *soft skills* com base nas experiências pessoais e na vivência profissional que trago para a sala de aula."

Por outro lado, apenas um professor relatou o uso de estratégias sistematizadas. P7 explicou que "Eu faço uma planilha para mapear *soft skills* e entender quais habilidades comportamentais e técnicas os alunos apresentam." Essa abordagem permite uma análise mais detalhada, intencional e acompanhada das competências dos estudantes.

Pergunta: Você encontra dificuldades ao incorporar *soft skills* no ensino de programação? Caso sim, quais? Caso não, pode explicar por quê?

Quanto às dificuldades para incorporar SS nas aulas, os professores relataram que a falta de engajamento dos alunos e a dificuldade de trabalhar em grupo são os principais desafios. A maioria destacou que os alunos ainda não possuem maturidade ou empatia suficientes para colaborar de forma efetiva em atividades coletivas. P11 observou que "os alunos têm dificuldade em controlar o próprio tempo e em colaborar em grupo, pois vêm de uma cultura passiva de aprendizado." P13 complementou que "Percebemos que ainda há muito daquela ideia de 'faço a minha parte e você faz a sua', sem empatia pelo colega." Outro ponto mencionado foi a ausência de materiais didáticos que orientem os professores sobre como aplicar SS no ensino de programação. P5 observou que "Falta material que explique como aplicar *soft skills* de forma prática e direta no ensino de programação."

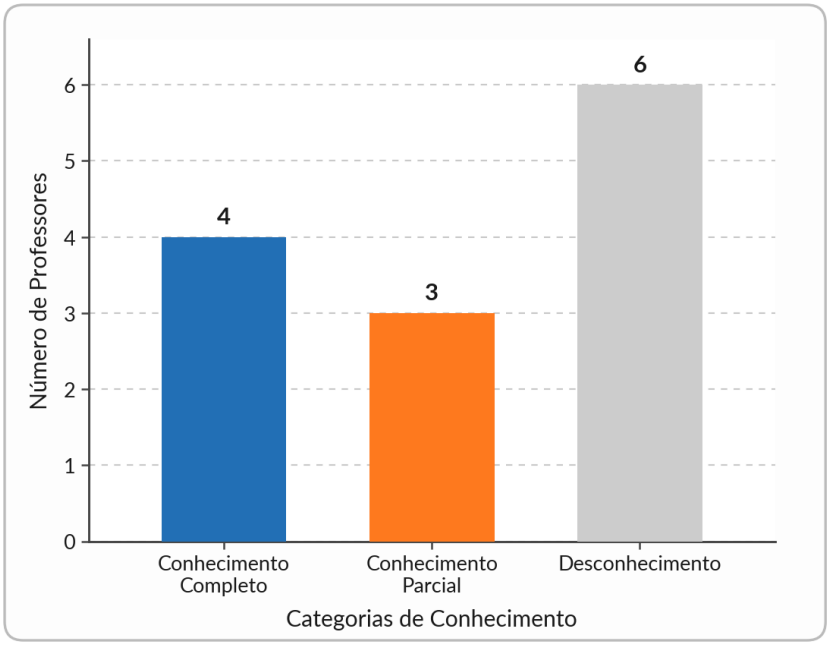
4.3.2.2 Percepção sobre componentes de PC

No ensino de programação, o PC tem sido discutido na literatura como uma abordagem que apoia a resolução de problemas e o desenvolvimento de habilidades técnicas. Para explorar o nível de familiaridade dos professores com esse conceito, foi formulada uma questão sobre o conhecimento dos professores a respeito do PC.

Pergunta: Você conhece o conceito de pensamento computacional? Caso sim, como você o definiria no contexto do ensino de programação?

Buscou-se identificar o nível de conhecimento dos professores sobre o PC e compreender suas percepções acerca da aplicação do conceito no ensino de programação. A Figura 22 apresenta a distribuição dos professores de acordo com seu nível de conhecimento sobre PC, organizado nas categorias conhecimento completo, conhecimento parcial e desconhecimento. A classificação seguiu o critério adotado na seção anterior, considerando a precisão conceitual e o grau de familiaridade demonstrado na resposta de cada professor, e não sua autoavaliação.

Figura 22: Conhecimento dos professores sobre PC



Fonte: Autoria própria.

A análise das respostas revelou que o conhecimento dos professores sobre PC apresenta variações. Alguns entrevistados demonstraram compreensão mais consistente, enquanto outros apresentaram conhecimento limitado ou declararam desconhecer o termo.

Conhecimento Completo: alguns professores destacaram a relação entre o PC e a resolução de problemas no ensino de programação. P6 afirmou que "o pensamento computacional é uma forma organizada de pensar na resolução de problemas." P7 reforçou essa visão, enquanto P11 mencionou que o conceito envolve habilidades como abstração e decomposição.

Conhecimento Parcial: um grupo de professores demonstrou familiaridade com

o termo, mas não soube explicá-lo com maior precisão conceitual. P3 comentou que "conheço o termo, mas não sei explicar formalmente."

Desconhecimento Alguns professores admitiram desconhecer o conceito ou indicaram a necessidade de esclarecimentos adicionais. P1 declarou que "Não conheço o conceito formalmente, mas já ouvi falar."

Pergunta: Você já utilizou componentes de pensamento computacional, como decomposição, abstração, algoritmos ou reconhecimento de padrões, no ensino de programação? Caso sim, esse uso foi de forma *ad hoc* (informal) ou sistematizada (planejada)? Poderia compartilhar brevemente sua experiência?

Essa pergunta teve como objetivo identificar se os professores integram esses componentes de forma espontânea (*ad hoc*) ou planejada (sistematizada), além de obter exemplos práticos de sua aplicação em sala de aula. A Tabela 7 apresenta as categorias identificadas nas respostas, com a descrição de cada tipo de uso e os professores que mencionaram essas práticas.

Tabela 7: Categorias de uso dos componentes de PC

Categoria	**Professores**	**Descrição**
Informal (Ad Hoc)	P1, P2, P3, P5, P7, P8, P9, P10, P11	Aplicação espontânea, geralmente relacionada a atividades práticas ou resolução de problemas.
Sistematizado	P4, P6, P12, P13	Uso planejado e integrado, com enfoque em decomposição, abstração e algoritmos.

Fonte: Autoria própria.

Conforme mostra a Tabela 7, a análise das respostas revelou que os componentes de PC são utilizados no ensino de programação, mas predominam os usos informais, *ad hoc*. Professores como P9 e P11 relataram que aplicam componentes como decomposição e algoritmos de maneira espontânea, muitas vezes adaptados ao contexto imediato da sala de aula. P9 destacou que "eu utilizo, mas sempre de forma informal, sem um planejamento mais profundo."

Por outro lado, alguns professores afirmaram utilizar esses componentes de forma sistematizada, integrando-os ao planejamento pedagógico. P12 comentou que "Trabalho de forma sistematizada, integrando os pilares ao planejamento de projetos." P13 complementou, apontando os benefícios da aplicação planejada: "quando os alunos partem para a prática, utilizando decomposição e abstração, observamos melhor a evolução deles."

Uma outra pergunta abordou a relação entre o desenvolvimento de PC e SS no ensino de programação. Formulada para compreender como essas dimensões se complementam,

ela buscou investigar as percepções dos professores sobre essa integração e seus impactos na aprendizagem dos alunos.

Pergunta: Você percebe alguma relação entre o desenvolvimento de pensamento computacional e *soft skills* no ensino de programação? Caso sim, como essas dimensões podem se complementar?

As respostas dos professores mostraram uma percepção positiva sobre a relação entre PC e SS no ensino de programação. A maioria apontou que essas dimensões são complementares e podem se integrar de forma natural. P13 afirmou que "quando trabalhamos criatividade na decomposição de problemas, vemos como essas competências se integram." Além disso, os professores relataram que a integração dessas habilidades favorece a aprendizagem colaborativa, promovendo organização, empatia e eficiência em atividades em grupo. P9 observou que "Trabalho em equipe e colaboração ajudam a atingir os pilares do pensamento computacional com eficiência."

Apesar dessa percepção positiva, os relatos também indicam que essa integração precisa ser planejada para ocorrer de forma intencional. P13 comentou que "precisamos planejar como aliar as *soft skills* com o pensamento computacional, destacando habilidades em etapas específicas." Por fim, os professores enfatizaram que a integração dessas dimensões no ensino de programação prepara os alunos para contextos profissionais colaborativos. P6 afirmou que "Essas habilidades são fundamentais no ensino de programação e preparam para o trabalho em equipe, algo indispensável no mercado."

4.3.2.3 Sugestões para implementação do *framework*

Para compreender como os professores estruturariam a implementação de um *framework* no ensino de programação, foram exploradas suas percepções sobre o tempo, a estrutura, a preparação e os recursos necessários para sua aplicação.

Pergunta: Se você fosse aplicar um *framework* no ensino de programação, como você organizaria sua implementação? Pode descrever sua preferência em termos de tempo e estrutura?

A análise dos resultados mostrou que os professores apresentaram diferentes sugestões para a implementação do *framework*, com foco em um planejamento estruturado e em recursos adequados. A maioria sugeriu que o *framework* fosse aplicado ao longo de um semestre, permitindo tempo suficiente para o desenvolvimento de PC e SS. P1 comentou que "aplicaria ao longo de um semestre, com foco em aulas práticas, para garantir

maior aprendizado." P7 complementou que "um semestre é essencial para lidar com duas competências."

Outros professores propuseram abordagens alternativas, como dividir a aplicação em blocos específicos ou integrá-la continuamente ao longo de todo o curso. P9 sugeriu: "Inicialmente, explicaria a metodologia em algumas aulas, para depois trabalhar com atividades práticas ao longo do semestre." Já P10 afirmou: "eu aplicaria o *framework* em todo o curso, para que essas habilidades sejam trabalhadas desde o início da formação."

Pergunta: Um treinamento prático ou uma instrução teórica sobre o funcionamento do *framework* seria importante para sua aplicação no ensino de programação? Ambos seriam necessários? Por quê?

Pergunta: Caso uma instrução teórica ou treinamento sejam considerados importantes, qual formato você acredita ser mais adequado para oferecê-los?

Todos os professores que responderam a essas questões consideraram importante algum preparo para aplicar o *framework*. P5 justificou essa necessidade pelo fato de os docentes levarem para a sala de aula sobretudo suas vivências, sem informações sistematizadas sobre a integração de PC e SS. Parte dos entrevistados defendeu a combinação de teoria e prática: P1 sugeriu uma oficina de aplicação, P4 propôs uma etapa teórica inicial seguida de prática, e P9 afirmou que a instrução teórica isolada deixa muito a cargo do professor, sendo o treinamento prático um complemento necessário. P13 também valorizou as duas formas, com ênfase na prática e no apoio de um tutorial para consulta durante a aplicação.

Quanto ao formato, predominou a preferência por materiais *on-line* e de autoestudo, indicada por P3, P4, P8, P10, P11 e P13, que citaram a falta de tempo dos professores e as limitações geográficas. P10 propôs um manual ou sequência didática para autoestudo, e P8 considerou o material *on-line* suficiente, dispensando o presencial. Outros valorizaram o encontro presencial: P9 manifestou preferência por ele, enquanto P2 e P5 o consideraram mais interessante, mas aceitaram o formato *on-line* viabilizado pelas ferramentas de videoconferência. P1 sugeriu um modelo híbrido, com material *on-line* combinado a um encontro presencial. O professor P6 não chegou a abordar esses pontos na entrevista. As respostas a essas duas questões, além de completarem o panorama do diagnóstico, subsidiaram as decisões de estruturação e de preparação para uso do *framework*, detalhadas no capítulo de sua proposição.

Outra pergunta investigou o que os professores consideram mais relevante para apoiar

a aplicação do *framework*, com foco em PC e SS.

Pergunta: Entre recursos pedagógicos, abordagens metodológicas e ferramentas, quais você considera mais relevantes para apoiar a aplicação de um *framework* no ensino de programação com foco em pensamento computacional e *soft skills*?

As sugestões dos professores podem ser organizadas em três categorias. Como recursos pedagógicos, apontaram materiais práticos, como guias detalhados e exemplos reais. Como abordagens metodológicas, apontaram metodologias ativas, em especial a Aprendizagem Baseada em Problemas e a gamificação, e P11 afirmou que "a metodologia ativa de ensino é essencial nesse processo". Como ferramentas, mencionaram recursos tecnológicos, como simuladores e plataformas de gestão de projetos, e P6 comentou que "ferramentas gamificadas e projetos integradores ajudam a engajar os alunos desde o início". P13 sugeriu ainda o uso do Scrum com Trello para a gestão de tempo e o acompanhamento das atividades. Os professores reconhecem o valor desses recursos isoladamente, mas não relatam um conjunto de boas práticas que os articule de forma integrada e sistematizada. É essa lacuna que o *framework* proposto nesta tese busca preencher.

4.4 Resultados e discussão

A partir da análise dos dados obtidos por meio de questionários e entrevistas com professores, foi possível identificar desafios e oportunidades no ensino de programação, especialmente relacionados ao desenvolvimento de PC e SS. Os aspectos identificados são discutidos com base nas evidências da pesquisa e em diálogo com referências da literatura. Quando apropriado, as falas dos professores são incorporadas para enriquecer a análise crítica.

Os dados dos professores dos IFs revelam que os principais desafios no ensino de programação estão relacionados à falta de base em lógica e abstração por parte dos alunos, além de indícios de desmotivação relatados pelos docentes. Esse diagnóstico, já presente nas respostas abertas do questionário, reaparece nas entrevistas. Conforme apontou P4, "O verdadeiro problema não é a linguagem de programação, mas a falta de lógica de programação. Muitos alunos conhecem as funções da linguagem de programação, mas enfrentam dificuldades em interpretar problemas e criar algoritmos para resolvê-los." P7 corroborou esse ponto ao afirmar que "A gente tenta ensinar a lógica ao longo do curso, mas muitos alunos chegam com dificuldades de base. Eles não tiveram contato com a ideia de algoritmos antes, então acabam decorando trechos de código sem entender o que

estão fazendo."

Tais desafios são discutidos na literatura, que aponta barreiras cognitivas enfrentadas pelos alunos ao lidar com conceitos abstratos (Medeiros; Falcão; Ramalho, 2020; Wing, 2006). Esse aspecto também aparece nas respostas ao questionário. Ao serem questionados sobre a influência da matemática básica na aprendizagem de programação, 82% dos professores indicaram que ela influencia, o que reforça a interpretação de que a fragilidade na base matemática e lógica é um fator relevante. A literatura também associa esse ponto à programação, ao apontar a base matemática frágil entre as principais dificuldades de estudantes iniciantes (Ribeiro *et al.*, 2020).

A dificuldade em dominar o pensamento algorítmico compromete a capacidade de decompor problemas e reconhecer padrões, componentes importantes para o desenvolvimento da programação (Rocha *et al.*, 2023; Hudin, 2023; Holanda; Paiva Freire; Silva Coutinho, 2019). Esse resultado dialoga com relatos dos professores, que indicam dificuldades dos alunos em formular soluções algorítmicas, aspecto também discutido por Medeiros, Falcão e Ramalho (2020) e Medeiros, Ramalho e Falcão (2018). Além disso, conceitos como decomposição de problemas, abstração e reconhecimento de padrões são apontados como relevantes para a aprendizagem de programação (Nouri *et al.*, 2020). No entanto, a maioria dos professores entrevistados indicou que esses componentes ainda não são desenvolvidos de maneira sistemática nas aulas.

No questionário, porém, a percepção sobre engajamento e participação dos alunos não foi conclusiva, com predomínio de respostas neutras quanto ao engajamento e distribuição equilibrada quanto à participação em aula. Embora a literatura trate a desmotivação como obstáculo recorrente na aprendizagem de programação (Ribeiro *et al.*, 2020; Morais; Mendes Neto; Osório, 2020), entre os docentes dos IFs ela apareceu mais nas entrevistas e nas respostas abertas do que nas escalas do questionário. Esse resultado sugere um quadro de motivação heterogêneo, e não de baixo engajamento generalizado.

A ausência de estratégias pedagógicas estruturadas dificulta o desenvolvimento dos alunos. Estudos indicam que abordagens orientadas e bem planejadas podem proporcionar avanços importantes na aprendizagem de programação (Rocha *et al.*, 2023; Valls Pou; Canaleta; Fonseca, 2022). O professor P6 observou que "A falta de estratégias estruturadas no ensino de programação compromete a capacidade dos alunos de desenvolver pensamento lógico e resolver problemas de forma autônoma."

Além das dificuldades cognitivas, o questionário apontou limitações nas condições de ensino. Predominou a percepção de que o tempo disponível para ensinar programação

é insuficiente (47,4% nas notas 1 e 2), e uma parcela expressiva relatou sobrecarga percebida nos alunos (42,1% nas notas 4 e 5). Esses dados dialogam com a literatura, que trata a aprendizagem de programação como atividade que demanda estudo, dedicação e disponibilidade de tempo (Silva; Moreira, 2021), e sustentam a leitura, já apontada no mapeamento, de que parte importante dos obstáculos está nas condições oferecidas para o ensino, e não apenas nos alunos.

No que se refere às SS, a literatura enfatiza sua importância para o sucesso acadêmico e profissional (Chévez-Coronel *et al.*, 2023; Younis *et al.*, 2019). Os professores entrevistados reconhecem o impacto positivo das SS no ensino de programação, mas relataram dificuldades para incorporá-las de forma estruturada. A baixa autonomia dos alunos e a dificuldade de trabalhar em equipe foram os principais desafios apontados. P11 associou esses desafios à pouca familiaridade dos alunos com metodologias que exijam maior proatividade, em linha com o relatado na análise das entrevistas.

Outras habilidades, como criatividade, proatividade e adaptabilidade, foram consideradas relevantes para a resolução de problemas e para a autonomia dos alunos. Na fala de P5, "A programação é resolução de problemas e, para resolver problema a gente precisa ser muitas vezes criativo, encontrar soluções que não existem ou adaptar soluções." Esse aspecto se alinha ao estudo de Chévez-Coronel *et al.* (2023).

A gamificação e a Aprendizagem Baseada em Problemas são estratégias apontadas na literatura como favoráveis ao desenvolvimento do PC e das SS, promovendo habilidades como criatividade e pensamento crítico (Valls Pou; Canaleta; Fonseca, 2022; Younis *et al.*, 2019). Os professores entrevistados reforçam essa percepção ao sugerir que metodologias ativas podem contribuir para uma melhor integração entre essas dimensões.

Os dados indicam que os professores reconhecem a importância de um ensino estruturado e sistematizado para PC e SS, mas apontam a necessidade de diretrizes claras para seu desenvolvimento. Como relatado na análise das entrevistas sobre implementação do *framework*, P9 propõe introduzir a metodologia em algumas aulas iniciais e só então passar às atividades práticas, distribuídas ao longo do semestre. Esse relato reforça a necessidade de um planejamento gradual, que permita aos alunos assimilar os conceitos de PC e SS de forma progressiva.

Apesar do reconhecimento da importância do PC e das SS no ensino de programação, sua promoção ainda ocorre de forma fragmentada e pouco sistematizada na EPTNM, em especial nos cursos de computação. A falta de estrutura pedagógica, as dificuldades cognitivas dos alunos, os indícios de desmotivação e a necessidade de suporte aos professores

constituem os principais desafios identificados.

Algumas limitações decorrem do instrumento de coleta. No questionário, os componentes do PC foram reunidos em uma única afirmação, o que dificultou análises isoladas sobre abstração, decomposição, reconhecimento de padrões e algoritmos. Além disso, as respostas do questionário sobre SS podem estar sujeitas ao viés de desejabilidade social, e a questão sobre metodologias ativas não diferenciou o uso efetivo dessas estratégias do reconhecimento de seu valor teórico. Também não foram apresentadas definições operacionais para termos como engajamento e metodologias ativas.

Além dessas limitações do instrumento, este estudo não analisou diretamente o impacto das estratégias sugeridas, o que representa uma limitação a ser explorada em pesquisas futuras. Nesse contexto, propõe-se um *framework* que auxilie os professores no planejamento e na aplicação de estratégias para promover PC e SS em suas aulas, favorecendo uma implementação estruturada e adaptável ao contexto educacional.

4.5 Considerações finais do capítulo

Este capítulo discutiu os desafios e as oportunidades sobre o desenvolvimento dos componentes do PC e das competências de SS no ensino de programação nos IFs. A análise das respostas de 133 professores ao questionário e das entrevistas com 13 docentes revelou obstáculos que afetam a aprendizagem dos estudantes e a prática pedagógica. Os resultados indicam que, na percepção dos professores, a maioria dos estudantes enfrenta dificuldades na compreensão de lógica e abstração, o que compromete a aprendizagem da programação. Somam-se a isso indícios de desmotivação relatados pelos docentes e a ausência de estratégias didáticas sistematizadas para promover PC e SS.

Quanto ao PC, verificou-se que os componentes decomposição, reconhecimento de padrões, abstração e algoritmos são trabalhados de maneira informal, sem uma abordagem progressiva e estruturada. Quanto às SS, competências como colaboração, criatividade e pensamento crítico enfrentam desafios de implementação, especialmente quando associadas a atividades de resolução de problemas. Os professores apontaram a necessidade de estratégias sistematizadas que favoreçam o avanço dessas competências. Métodos como gamificação e metodologias ativas foram apontados como recursos que podem contribuir para o engajamento e para a aprendizagem autônoma dos estudantes.

Este capítulo dialogou com o segundo objetivo específico, ao investigar as percepções e práticas de professores de programação dos IFs quanto ao uso do PC e das SS. Os

achados respondem, em parte, à questão de pesquisa, ao indicar que os componentes do PC e as competências de SS são reconhecidos pelos professores como relevantes para apoiar o ensino de programação na EPTNM. No entanto, os dados também mostram que esses elementos ainda são trabalhados de modo informal e fragmentado, sem diretrizes que orientem sua integração à prática pedagógica. Esse resultado converge com a lacuna principal identificada no mapeamento apresentado no Capítulo 3 e com os fundamentos discutidos no Capítulo 2.

Ao reunir o que a literatura indica e o que os professores relatam neste capítulo, delimitam-se os achados empíricos que sustentam a construção do *framework*. Os componentes do PC mais recorrentes e as competências do modelo 4C, confirmados pelas duas fontes, definem o núcleo da proposta. Além disso, a demanda por sistematização e por recursos integrados orienta suas diretrizes. Esses resultados servem de base para o terceiro objetivo específico, apresentado no próximo capítulo, em que o *framework* SoPensa é descrito com diretrizes, estrutura e abordagens metodológicas.

5 SOPENSA: *FRAMEWORK* COM BASE NO PENSAMENTO COMPUTACIONAL E *SOFT SKILLS*

Este capítulo apresenta o SoPensa, um *framework* conceitual-operacional voltado à resolução de problemas, concebido para apoiar professores na promoção integrada do PC e das SS no ensino de programação na EPTNM. Sua construção foi embasada no referencial teórico, no MSL e na análise das percepções de professores de IFs, obtidas por meio de questionários e entrevistas. O nome SoPensa resulta da combinação de *Soft Skills* e "Pensamento Computacional" e sintetiza a proposta de integrar essas dimensões no ensino de programação. O *framework* fundamenta-se nas teorias pedagógicas da AS, de Ausubel, e do Construcionismo, de Papert, que sustentam a valorização dos conhecimentos prévios, a construção ativa do conhecimento e a aprendizagem orientada por atividades contextualizadas.

5.1 Estrutura e componentes do *framework* SoPensa

Após o estudo exploratório, a partir da análise da literatura e das percepções dos professores, foi possível identificar lacunas no ensino de programação na EPTNM, especialmente quanto ao apoio docente para promover, de forma sistematizada, o PC e as SS. Essas dimensões estão relacionadas ao engajamento dos alunos e à sua preparação acadêmica e profissional. Os achados do MSL e do estudo empírico com professores de IFs, apresentados nos Capítulos 3 e 4, reforçaram a necessidade de uma abordagem capaz de organizar essa relação no planejamento e na prática pedagógica.

Os componentes de PC e as competências de SS que estruturam o *framework* foram definidos a partir de uma base teórica da literatura e, em seguida, analisados à luz dos dados empíricos. No PC, partiu-se do conjunto consensual dos quatro componentes, decomposição, reconhecimento de padrões, abstração e algoritmos, reconhecido nos trabalhos que organizam o conceito (Wu *et al.*, 2024; BBC, 2024). Nas SS, adotou-se o modelo 4C, que reúne colaboração, comunicação, criatividade e pensamento crítico como competências centrais para o século XXI (Jalinus; Putra *et al.*, 2024; Partnership for 21st

Century Learning, 2019; Hu, 2024). Sobre esse fundamento, o MSL (Capítulo 3) e as entrevistas com os professores (Capítulo 4) foram usados para verificar a presença dessas competências no ensino de programação e orientar sua priorização pela frequência com que aparecem.

Os dois conjuntos teóricos encontraram respaldo nos dados empíricos. No PC, os quatro componentes também aparecem entre os mais frequentes no mapeamento. Nas SS, colaboração e criatividade figuram entre as mais citadas nas duas fontes empíricas, e a comunicação destaca-se sobretudo no MSL. O pensamento crítico, embora pouco frequente no mapeamento, é confirmado pelas entrevistas e mantido por integrar o modelo 4C. A resolução de problemas, embora muito citada, não foi tomada como componente, pois atravessa o PC, as SS e a própria programação, sendo tratada como objetivo transversal do *framework*. Colaboração e trabalho em equipe são considerados equivalentes e agrupados neste estudo.

As Tabelas 8 e 9 sintetizam essa decisão, relacionando cada componente e cada competência à sua base teórica e à evidência empírica correspondente. A Tabela 8 reúne os quatro componentes consensuais do PC e também os que aparecem na literatura sem integrar esse conjunto, de modo a registrar as razões de sua não incorporação. A Tabela 9 concentra-se nas competências que orientaram a decisão, ou seja, as quatro do modelo 4C, a resolução de problemas como eixo transversal e as competências de maior frequência situadas fora do 4C, cuja não incorporação precisa ser justificada. As competências de baixa frequência, já registradas na Figura 21, não são retomadas linha a linha por não alterarem a decisão.

Tabela 8: Quadro de decisão dos componentes de PC

Componente	**Fundamentação teórica**	**MSL (n/18)**	**Entrevistas**	**Decisão**	**Base da decisão**
Decomposição	Componente do PC	14	Uso confirmado	Incluído	Componente consensual; mais frequente no MSL
Reconhecimento de padrões	Componente do PC	3	Uso confirmado	Incluído	Componente consensual; 4º no MSL
Abstração	Componente do PC	10	Uso confirmado	Incluído	Componente consensual; alta frequência no MSL
Algoritmos	Componente do PC	13	Uso confirmado	Incluído	Componente consensual; alta frequência no MSL
Depuração	—	2	—	Excluído	Fora do conjunto consensual; baixa frequência
Automação	—	0	—	Excluído	Fora do conjunto consensual; baixa frequência
Paralelismo	—	1	—	Excluído	Fora do conjunto consensual; baixa frequência
Avaliação	—	1	—	Excluído	Fora do conjunto consensual; baixa frequência

Fonte: Autoria própria.

Tabela 9: Quadro de decisão das competências de SS

Competência	Fundamentação teórica	MSL (n/18)	Entrev. (n/13)	Decisão	Base da decisão
Colaboração	Modelo 4C	17	10	Incluída	4C; 1ª nas duas fontes empíricas
Comunicação	Modelo 4C	13	4	Incluída	4C; alta frequência no MSL
Criatividade	Modelo 4C	7	7	Incluída	4C; frequente nas duas fontes
Pensamento crítico	Modelo 4C	2	6	Incluída	4C; confirmada nas entrevistas
Resolução de problemas	Transversal	4	6	Transversal	Objetivo do *framework*; não tomada como componente
Adaptabilidade	—	0	5	Excluída	Fora do 4C; ausente no MSL
Regulação emocional	—	2	3	Excluída	Fora do 4C; baixa frequência
Curiosidade	—	0	5	Excluída	Fora do 4C; presente só nas entrevistas
Proatividade	—	0	5	Excluída	Fora do 4C; presente só nas entrevistas

Fonte: Autoria própria.

Com base nesses resultados, o *framework* SoPensa foi estruturado para integrar essas dimensões ao ensino de programação, articulando os componentes do PC, as competências de SS e as teorias da AS e do Construcionismo. A Figura 23 reúne os componentes do PC e as competências de SS mais mencionados na literatura e na percepção dos professores, mostrando a convergência entre as duas fontes que sustentou a seleção.

Figura 23: Principais componentes do PC e de SS considerados para o *framework*

Fonte: Autoria própria.

As percepções dos professores reforçam a necessidade de estratégias estruturadas para integrar o PC e as SS ao ensino de programação. Para responder a essa lacuna, o *framework* SoPensa foi concebido como uma abordagem sistematizada voltada à EPTNM. A Tabela 10 sintetiza os componentes do PC identificados no MSL e no estudo empírico, relacionando-os às fontes da literatura e à percepção dos professores.

Tabela 10: Síntese da literatura e das percepções dos professores sobre PC

Componente	**Base na literatura**	**Síntese da percepção docente e do MSL**
Decomposição	Consiste em dividir um problema complexo em partes menores e mais gerenciáveis, facilitando sua compreensão e permitindo uma abordagem mais eficiente para sua resolução (Fang *et al.*, 2022; Wing, 2006, 2014).	Os professores reconhecem a importância da decomposição no ensino de programação, mas relataram que os alunos enfrentam dificuldades ao segmentar problemas complexos em partes menores e gerenciáveis.
Reconhecimento de padrões	Consiste em identificar padrões ou similaridades entre diferentes dados ou situações, possibilitando a aplicação de soluções já conhecidas a novos problemas semelhantes (Grover; Pea, 2013; Wing, 2006; Farias; Barone, 2023; Hudin, 2023).	Os professores destacaram o reconhecimento de padrões como relevante para a criação de algoritmos, mas apontaram que os alunos têm dificuldades em desenvolvê-lo de forma satisfatória.
Abstração	Permite modelar problemas e focar nos aspectos importantes, eliminando detalhes irrelevantes para uma solução mais eficiente (Brennan; Resnick *et al.*, 2012; Wing, 2014; Barr; Harrison; Conery, 2011).	Os professores destacaram que a prática da abstração favorece a evolução dos alunos, mas relataram dificuldades na aplicação desse componente em contextos reais.
Algoritmos	Facilita a construção do pensamento lógico em programação, organizando e estruturando a resolução de problemas complexos de forma sistemática (Yusoff *et al.*, 2020; Grover; Pea, 2013; Wing, 2014).	Os professores apontaram que, embora os alunos conheçam funções da linguagem, enfrentam dificuldades na interpretação de problemas e na criação de algoritmos, o que reforça a importância da prática para seu aprimoramento.

Fonte: Autoria própria.

A Tabela 11 apresenta uma síntese das competências de SS identificadas no MSL e no estudo empírico, relacionando-as às fontes da literatura e à percepção dos professores.

Tabela 11: Síntese da literatura e das percepções dos professores sobre SS

Competência	**Base na literatura**	**Síntese da percepção docente e do MSL**
Colaboração	Possibilita o trabalho em equipe, o compartilhamento de conhecimentos e o uso de recursos, favorecendo a construção de soluções mais eficientes (Resnick *et al.*, 2009; Barr; Harrison; Conery, 2011; Kapustina; Matyushina, 2023).	Os professores apontaram essa competência como a mais relevante, sendo também a mais citada no MSL.
Comunicação	Envolve a troca de informações, a expressão de ideias e a interação entre os participantes, sendo relevante para a resolução colaborativa de problemas (Papert, 1990; Matturro; Raschetti; Fontán, 2019).	Os professores relacionaram a comunicação ao trabalho em equipe e à expressão de dúvidas e ideias pelos alunos, sendo a segunda competência mais citada no MSL.
Criatividade	Contribui para impulsionar a inovação e a elaboração de alternativas na resolução de problemas (Wing, 2011; Grover; Pea, 2013; Tsortanidou; Daradoumis; Barberá, 2023).	Os professores destacaram a criatividade como uma competência importante para enfrentar desafios técnicos e desenvolver soluções inovadoras.
Pensamento crítico	Permite analisar problemas de forma lógica, criteriosa e fundamentada (Valls Pou; Canaleta; Fonseca, 2022; Resnick *et al.*, 2009; Younis *et al.*, 2019).	A literatura e os professores apontam o pensamento crítico como uma competência relevante para analisar problemas, avaliar alternativas e justificar decisões no ensino de programação.

Fonte: Autoria própria.

O *framework* SoPensa foi estruturado com base em teorias da aprendizagem a fim de favorecer seu alinhamento com fundamentos pedagógicos reconhecidos. Para isso, adotaram-se a AS Ausubel (1963) e Ausubel, Novak e Hanesian (1980) e o Construcionismo (Papert, 1980). A AS enfatiza que novos conceitos devem se conectar ao conhecimento prévio para serem assimilados de forma efetiva, enquanto o Construcionismo propõe que a aprendizagem ocorre por meio da experimentação ativa e da construção do conhecimento em contextos práticos.

A partir dessas teorias, o *framework* organiza-se em quatro fases principais: Diagnosticar e Organizar, Contextualizar e Ambientar, Desafiar e Mediar, e Analisar e Consolidar. Cada fase é conduzida pelo professor, que atua como mediador do processo, promovendo a conexão entre os componentes do PC e as competências de SS. O fluxo entre as fases é dinâmico e iterativo, permitindo revisões e refinamentos ao longo da mediação e acompanhamento, de modo a ajustar a progressão à aprendizagem dos alunos. A Tabela 12 relaciona cada fase à teoria que a fundamenta e à respectiva justificativa pedagógica.

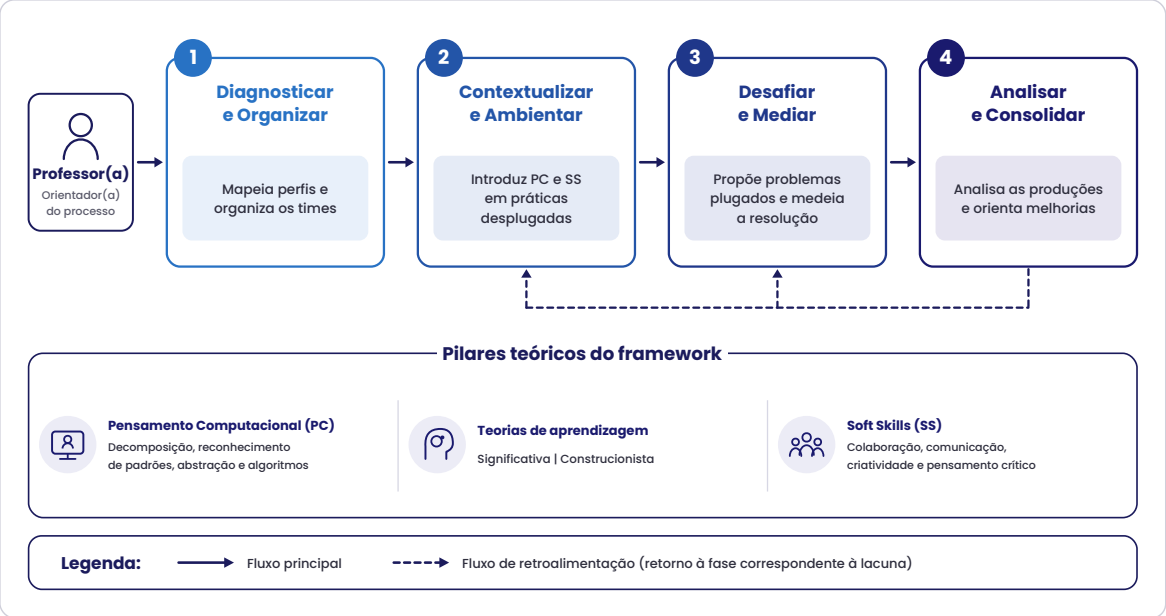
Tabela 12: Relação entre as teorias de aprendizagem e as fases do *framework* SoPensa

Fase do *framework*	**Teoria aplicada**	**Justificativa pedagógica**
1. Diagnosticar e Organizar	Aprendizagem Significativa	Com base em Ausubel, Novak e Hanesian (1980), fundamenta o diagnóstico dos conhecimentos prévios dos alunos e a organização do ensino a partir de suas experiências, favorecendo a assimilação de novos conceitos.
2. Contextualizar e Ambientar	Aprendizagem Significativa e Construcionismo	Com base em Ausubel, Novak e Hanesian (1980) e (Papert, 1980), orienta a conexão entre novos conceitos e conhecimentos prévios e a construção ativa do conhecimento por meio de situações contextualizadas.
3. Desafiar e Mediar	Construcionismo	Com base em (Papert, 1980), sustenta a aprendizagem por meio da experimentação, da resolução de problemas e da construção de soluções, favorecendo a mobilização da colaboração, comunicação, criatividade e pensamento crítico.
4. Analisar e Consolidar	Aprendizagem Significativa e Construcionismo	Com base em Ausubel, Novak e Hanesian (1980) e (Papert, 1980), apoia a consolidação conceitual, a revisão das soluções e o aprimoramento iterativo das produções, favorecendo reflexão, avaliação e reorganização dos conhecimentos.

Fonte: Autoria própria.

A Figura 24 apresenta a estrutura conceitual do *framework*, mostrando como essas fases se conectam para apoiar o professor na condução de práticas de ensino que promovam os componentes do PC e as competências de SS na área de programação.

Figura 24: Fluxo conceitual do *framework* SoPensa



Fonte: Autoria própria.

5.1.1 Descrição das fases e ações do professor

Esta seção detalha, fase a fase, os objetivos e as ações docentes previstas no *framework*. Cada tabela organiza os passos em duas dimensões: o que o professor faz e como essa ação é operacionalizada na prática.

Fase 1: Diagnosticar e Organizar.

Objetivo: Mapear os conhecimentos prévios dos alunos e organizar o ambiente de aprendizagem para o ensino de programação, favorecendo sua adaptação às atividades propostas. A fase permite identificar diferentes níveis de experiência e orientar a definição das estratégias de ensino. A Tabela 13 apresenta a Fase 1 do *framework* SoPensa.

Tabela 13: Fase 1: Diagnosticar e Organizar

Passo	**O que o professor faz?**	**Como fazer na prática?**
1. Diagnosticar conhecimentos prévios	Identificar o nível de familiaridade dos alunos com conceitos de PC, SS e programação.	Aplicar um questionário diagnóstico ou atividade inicial envolvendo desafios básicos de lógica de programação, PC e SS.
2. Observar componentes do PC e das SS	Identificar como os alunos interagem, trabalham em equipe e lidam com desafios.	Promover dinâmicas colaborativas e analisar a forma como os alunos resolvem problemas em time.
3. Identificar dificuldades e perfis	Mapear quais alunos precisam de suporte e quais podem apoiar colegas na aprendizagem de programação.	Criar categorias básicas, como alunos com mais facilidade em lógica, alunos mais colaborativos e alunos que precisam de mais apoio.
4. Organizar o ambiente de aprendizagem	Formar times heterogêneos, considerando perfis técnicos e interpessoais.	Criar times diversificados, favorecendo a distribuição equilibrada dos conhecimentos prévios e do desenvolvimento de PC e SS.

Fonte: Autoria própria.

Fase 2: Contextualizar e Ambientar.

Objetivo: introduzir os alunos aos componentes do PC e às competências de SS no ensino de programação, favorecendo uma aprendizagem inicial por meio de atividades práticas, como a Computação Desplugada (Brackmann, 2018; Sim; Teow; Lau, 2021), antes da transição para a programação com código. A Tabela 14 apresenta a Fase 2 do SoPensa.

Tabela 14: Fase 2: Contextualizar e Ambientar

Passo	**O que o professor faz?**	**Como fazer na prática?**
1. Introduzir PC e SS	Explicar conceitos de PC e SS e sua importância na resolução de problemas computacionais.	Relacionar os componentes do PC e as competências de SS com a aprendizagem de programação, demonstrando como auxiliam no desenvolvimento de soluções eficientes.
2. Ambientar os alunos com Computação Desplugada	Proporcionar uma experiência de aprendizagem sem o uso direto de computadores.	Aplicar desafios físicos e jogos de lógica que simulem a resolução de problemas computacionais, permitindo que os alunos internalizem conceitos fundamentais antes de codificar.
3. Relacionar conceitos computacionais com problemas reais	Associar PC e SS a desafios práticos do dia a dia.	Criar cenários nos quais os alunos precisem resolver problemas utilizando componentes do PC e competências de SS de maneira desplugada (Brackmann, 2018; Sim; Teow; Lau, 2021).
4. Consolidar a base lógica para a fase seguinte	Garantir que os alunos consolidem o raciocínio computacional ainda sem o uso do computador.	Representar as soluções por meio de pseudocódigo e fluxogramas em atividades desplugadas, preparando a transição para a programação, que ocorre na Fase 3.

Fonte: Autoria própria.

Fase 3: Desafiar e Mediar.

Objetivo: aplicar os componentes do PC e mobilizar competências de SS na resolução de problemas reais. Nesta fase, os alunos desenvolvem soluções por meio da programação, realizam trabalho colaborativo e utilizam criatividade, pensamento crítico e comunicação para justificar as decisões tomadas. A Tabela 15 apresenta a Fase 3 do SoPensa.

Tabela 15: Fase 3: Desafiar e Mediar

Passo	**O que o professor faz?**	**Como fazer na prática?**
1. Apresentar um problema computacional	Propor um desafio realista que exija a aplicação dos componentes do PC e a mobilização das competências de SS.	Utilizar aprendizagem baseada em problemas e desafios (Jalinus; Putra *et al.*, 2024), fornecendo um contexto real e incentivando os alunos a formular soluções.
2. Aplicar os componentes do PC	Mediar a resolução do problema com base em decomposição, reconhecimento de padrões, abstração e algoritmos.	Incentivar os alunos a dividir o problema em partes menores, identificar padrões, abstrair informações relevantes e elaborar um plano antes da codificação.
3. Introduzir a programação	Mediar a transição das atividades desplugadas para a programação.	Conduzir a codificação iniciando pela programação visual em blocos e avançando para a programação textual, garantindo a compreensão da lógica, da sintaxe e da estrutura (Taylor *et al.*, 2019; Deng *et al.*, 2020).
4. Mobilizar competências de SS	Estimular colaboração, comunicação, criatividade e pensamento crítico durante a resolução do problema.	Incentivar a interação entre os alunos, a apresentação de soluções, a justificativa de decisões e a discussão sobre a eficiência das estratégias adotadas.
5. Desenvolver um projeto aplicado	Consolidar a aprendizagem por meio da criação de uma solução prática e programável.	Orientar os alunos no desenvolvimento de um projeto em linguagem formal de programação, aplicando componentes do PC e competências de SS na resolução de um problema relevante.
6. Conduzir discussões e reflexões	Apoiar os alunos na análise das soluções e no aprimoramento das decisões tomadas.	Promover debates, *feedback* técnico e reflexão colaborativa sobre as soluções propostas, incentivando a justificativa e o aperfeiçoamento das decisões.

Fonte: Autoria própria.

Fase 4: Analisar e Consolidar.

Objetivo: realizar a análise crítica da aprendizagem, verificando em que medida os alunos consolidaram os componentes do PC e as competências de SS consideradas ao longo do processo. A Tabela 16 apresenta a Fase 4 do SoPensa.

Tabela 16: Fase 4: Analisar e Consolidar

Passo	**O que o professor faz?**	**Como fazer na prática?**
1. Sistematizar os registros do processo	Reunir registros, produções dos alunos e observações realizadas ao longo das fases anteriores.	Organizar as fichas de observação das fases, as rubricas e as produções dos times para identificar avanços e dificuldades.
2. Analisar os componentes do PC e as competências de SS	Verificar como os alunos mobilizaram os componentes do PC e as competências de SS nas soluções desenvolvidas.	Utilizar rubrica de acompanhamento e observação direta para documentar evidências de progresso e identificar lacunas de aprendizagem (Carvalho; Soares, 2025; Lima; Ferreira; Dias, 2024).
3. Retomar a autoavaliação e conduzir o diálogo de feedback	Estimular a reflexão dos alunos sobre sua participação, suas dificuldades e sua evolução ao longo do processo.	Retomar a autoavaliação dos times, respondida nas fases anteriores, e conduzir o diálogo de feedback, favorecendo comunicação, pensamento crítico e reflexão sobre a aprendizagem de PC e SS (Mioto *et al.*, 2019; Desrochers *et al.*, 2019).
4. Consolidar aprendizagens e orientar retomadas	Destacar os avanços obtidos, indicar pontos que precisam ser retomados e orientar ajustes nas soluções.	Realizar um momento de reflexão coletiva e, quando necessário, encaminhar os times à fase correspondente para reelaborar aspectos ainda não consolidados.

Fonte: Autoria própria.

5.1.2 Fluxo de aplicação do SoPensa

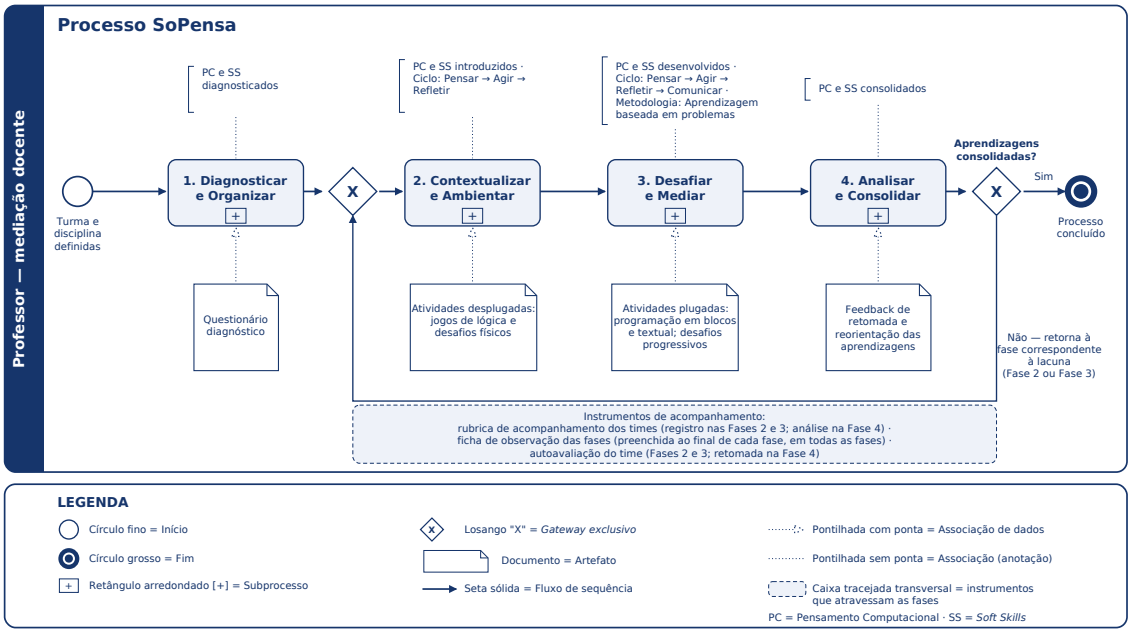
O SoPensa organiza-se em uma sequência de quatro fases desenvolvidas ao longo de um semestre. As ferramentas, metodologias e atividades podem ser ajustadas pelo professor conforme os objetivos pedagógicos, os recursos disponíveis e os resultados observados durante a aplicação.

Em cada fase, são definidas a entrada, a atividade, a saída e as dimensões consideradas. Na Fase 1, os componentes do PC e as competências de SS são mapeados para caracterizar o perfil inicial dos alunos e orientar a formação dos times. A formação dos times é orientada pelos resultados do diagnóstico, considerando os componentes do PC e as competências de SS definidos no *framework*. O professor pode utilizar diferentes estratégias de composição dos times de acordo com os objetivos pedagógicos e as características da turma.

Nas Fases 2 e 3, essas dimensões são desenvolvidas por meio do ciclo formativo Pensar, Agir e Refletir, acrescido, na Fase 3, da etapa Comunicar. A apresentação pública das soluções ocorre nessa etapa. Na Fase 4, as produções e os registros acumulados são analisados para identificar avanços, dificuldades e aspectos que ainda precisam ser consolidados. Nessa fase, a comunicação e o pensamento crítico são mobilizados por meio do diálogo de *feedback* e da autoavaliação dos times, sem a realização de uma nova apresentação pública.

A Fase 4 não encerra necessariamente o processo. Quando a análise identifica uma lacuna, os times retornam à Fase 2 ou à Fase 3, conforme a natureza do problema identificado. O retorno à Fase 2 ocorre quando é necessário retomar a contextualização e as atividades desplugadas, enquanto o retorno à Fase 3 ocorre quando a dificuldade está relacionada à resolução dos desafios, à programação ou à construção das soluções. Após essa retomada, os times seguem novamente para a Fase 4, na qual as produções revisadas são analisadas. Esse mecanismo permite que os componentes do PC e as competências de SS sejam retomados na fase correspondente à lacuna. O encadeamento das fases e o mecanismo de retroalimentação são apresentados na Figura 25.

Figura 25: Fluxo operacional do *framework* SoPensa



Fonte: Autoria própria.

O *framework* define a progressão pedagógica das atividades desplugadas, realizadas na Fase 2, para as atividades plugadas, desenvolvidas na Fase 3. Nessa progressão, a programação avança dos ambientes visuais em blocos para as linguagens textuais. O *framework* não determina uma ferramenta ou linguagem específica, cabendo ao professor selecioná-la de acordo com os objetivos pedagógicos e os recursos disponíveis. Para apoiar sua aplicação, cada fase apresenta ferramentas e metodologias sugeridas. Esses recursos não são prescritivos e podem ser substituídos por alternativas equivalentes já utilizadas pelo docente.

- **Fase 1 (Diagnosticar e Organizar):** plataformas de questionários digitais e dinâmicas de observação para mapear os perfis dos alunos e organizar os times.
- **Fase 2 (Contextualizar e Ambientar):** jogos de lógica e atividades de Computação Desplugada.
- **Fase 3 (Desafiar e Mediar):** ambientes de programação visual em blocos e linguagens textuais.
- **Fase 4 (Analisar e Consolidar):** rubrica de acompanhamento, ficha de observação das fases, diálogo de *feedback* e autoavaliação do time.

5.2 Operacionalização do *framework* SoPensa

O SoPensa é operacionalizado pelo professor, que conduz as quatro fases em sua prática docente regular, apoiado por instrumentos de acompanhamento do desenvolvimento do PC e das SS. Por não prescrever materiais didáticos nem linguagens específicas, o *framework* mantém-se compatível com os recursos que o docente já adota. Embora estabeleça as fases, os objetivos e as funções de cada etapa, sua aplicação não exige a reprodução das atividades utilizadas nesta pesquisa. O professor pode selecionar ou substituir estratégias, atividades e instrumentos de acordo com o perfil da turma, os objetivos de aprendizagem e as condições disponíveis, desde que sejam preservadas as funções previstas em cada fase. Dessa forma, o *framework* orienta a organização do processo sem estabelecer um roteiro único de aplicação.

5.2.1 Instrumentos para o acompanhamento docente

O acompanhamento do desenvolvimento do PC e das SS é contínuo e qualitativo, apoiado pelos instrumentos a seguir, aplicados pelo professor durante as aulas. Os instrumentos completos encontram-se no Apêndice E.

- **Ficha de Observação das Fases:** registro qualitativo de como ocorreu o processo em cada fase, destacando o que funcionou, o que não funcionou e os ajustes necessários.
- **Rubrica de Acompanhamento dos Times:** critérios estruturados para acompanhar e documentar o progresso dos times em PC e SS.
- **Autoavaliação do Time:** reflexão dos alunos sobre o próprio aprendizado e a organização do time.

Os instrumentos acompanham o processo ao longo das fases, com papéis distintos. A rubrica de acompanhamento registra o desenvolvimento das competências dos times nas Fases 2 e 3 e é retomada na análise da Fase 4. A ficha de observação das fases é preenchida pelo professor ao final de cada fase, registrando o andamento do processo. A autoavaliação do time é preenchida ao final das Fases 2 e 3 e retomada na análise da Fase 4, registrando a percepção dos integrantes sobre a organização e a aprendizagem do próprio time. Na Fase 4, os registros acumulados são analisados para identificar lacunas, consolidar as aprendizagens e orientar as retomadas às fases anteriores.

5.3 Considerações finais do capítulo

Este capítulo apresentou a estrutura e os componentes do *framework* SoPensa, organizado em quatro fases, e as diretrizes para sua aplicação no ensino de programação. O *framework* foi concebido para apoiar o professor no trabalho com os componentes do PC e as competências de SS no contexto do ensino técnico integrado ao ensino médio. Além da estrutura, foram apresentadas as formas de operacionalização das fases pelo professor e os instrumentos de acompanhamento dos componentes do PC e das competências de SS.

A proposta responde ao terceiro objetivo específico da tese, ao converter os resultados reunidos nos capítulos anteriores em uma estrutura organizada. Os componentes do PC mais recorrentes e as competências do modelo 4C, identificados no mapeamento do Capítulo 3 e no estudo empírico do Capítulo 4, definem o núcleo do *framework*, enquanto a demanda por sistematização apontada pelos professores orienta suas diretrizes. Mais do que descrever um modelo, o SoPensa oferece ao professor um percurso definido para associar dois eixos que, na literatura e na prática, costumam ser trabalhados de forma isolada. A aplicação da proposta em contexto real de ensino é examinada no Capítulo 6.

6 APLICAÇÃO DO SOPENSA

Este capítulo apresenta o processo de aplicação do *framework* SoPensa em sala de aula. O objetivo é descrever a apreciação por especialistas e a aplicação do *framework*, considerando o papel do professor como mediador, o envolvimento dos estudantes, os instrumentos de acompanhamento, os desafios enfrentados e as adaptações necessárias ao longo das atividades. O capítulo está organizado em três partes. A primeira etapa corresponde à apreciação por especialistas, na qual foram analisados o *framework*, suas fases, atividades e instrumentos. A segunda caracteriza o ambiente e os participantes da aplicação. A terceira descreve a aplicação das quatro fases do SoPensa e os ajustes decorrentes das situações observadas em sala de aula.

6.1 Apreciação do *framework* por especialistas

Antes da aplicação em sala de aula, realizou-se a apreciação do *framework* por especialistas, com o objetivo de examinar a clareza, a coerência pedagógica e a viabilidade prática de suas fases, fluxos e instrumentos. Buscou-se verificar se o planejamento apresentado no Capítulo 5 apresentava condições de aplicação no contexto da disciplina. A apreciação contou com docentes especialistas, sem a participação de estudantes, pois não se pretendia medir resultados de aprendizagem, mas avaliar a compreensão da estrutura e a viabilidade de aplicação do *framework*. Como a pesquisa se concentra no papel do professor como mediador, essa etapa permitiu verificar a adequação dos procedimentos previstos às práticas de sala de aula.

A reunião de apreciação ocorreu antes do início das aulas e teve duração aproximada de duas horas. A apreciação contou com dois especialistas em ensino de Lógica de Programação, ambos graduados em Sistemas de Informação e atuantes no Instituto Federal do Acre (IFAC). O Especialista 1, mestre, possui nove anos de experiência no ensino de Lógica de Programação, enquanto o Especialista 2, doutor, possui treze anos de experiência nesse mesmo componente curricular. Os especialistas foram convidados por *e-mail*

e participaram de forma voluntária. A seleção foi intencional, considerando a experiência específica no ensino de programação na educação profissional técnica, o que favoreceu uma análise fundamentada da estrutura do *framework*. Nenhum deles atuou na etapa de aplicação do *framework* em sala de aula.

A análise foi orientada por dez critérios relacionados à nomenclatura, à mediação docente, à retroalimentação, à organização das atividades, à representação do processo, à decisão final, à autoavaliação, aos instrumentos de acompanhamento, ao alinhamento curricular e à progressão pedagógica. Cada critério foi classificado como atendido, parcialmente atendido ou não atendido.

Além das classificações, foram apresentadas recomendações para aprimorar a proposta em seus aspectos conceituais e operacionais. Na reunião, o *framework* foi apresentado aos especialistas, que esclareceram dúvidas e, em seguida, responderam individualmente ao formulário de apreciação. As observações foram sistematizadas na Tabela 17, que reúne os critérios apreciados, as classificações atribuídas, as recomendações e os ajustes realizados. As recomendações foram incorporadas, com duas exceções.

A criação de um instrumento individual por aluno não foi acatada, pois se manteve o acompanhamento por time. Já a aplicação da autoavaliação na Fase 4 foi incorporada de forma parcial: em vez de uma nova aplicação, definiu-se a retomada, nessa fase, das respostas produzidas ao final das Fases 2 e 3, evitando a sobreposição de instrumentos em uma etapa destinada à análise e à consolidação. Após a incorporação das demais mudanças, a versão revisada foi novamente apreciada pelos especialistas, que a consideraram adequada para aplicação. O instrumento de apreciação encontra-se no Apêndice D.

Tabela 17: Resultados da apreciação do *framework* SoPensa pelos especialistas

Dimensão	**Critério apreciado**	**E1**	**E2**	**Recomendação apresentada**	**Ajuste realizado**
Nomenclatura	Consistência dos termos utilizados no *framework*	A	PA	O Especialista 2 recomendou substituir o termo "grupos" por "times".	O termo "times" foi adotado no *framework* e nos instrumentos de acompanhamento.
Mediação docente	Adequação do nome e da função atribuída à Fase 3	PA	A	O Especialista 1 recomendou substituir "Desafiar e Guiar" por "Desafiar e Mediar".	A Fase 3 passou a ser denominada "Desafiar e Mediar", reforçando o papel do professor como mediador.
Retroalimentação	Clareza do retorno à fase correspondente à lacuna identificada	PA	PA	O Especialista 1 recomendou que o fluxo retornasse ao ponto relacionado à dificuldade identificada. O Especialista 2 também recomendou o retorno à fase correspondente.	O fluxo foi ajustado para permitir o retorno à Fase 2 ou à Fase 3, conforme a lacuna identificada.
Organização das atividades	Adequação das atividades à função de cada fase	A	PA	O Especialista 2 recomendou manter os jogos na fase de contextualização e ambientação.	Os jogos e os desafios desplugados foram mantidos na Fase 2.
Representação do processo	Clareza e sequência lógica do fluxo de aplicação	PA	A	O Especialista 1 recomendou representar o processo por meio da notação Business Process Model and Notation (BPMN).	Foi elaborado um diagrama BPMN com as fases, as atividades, os instrumentos, os pontos de decisão e os fluxos de retorno.
Decisão final	Clareza do ponto de decisão após a Fase 4	A	PA	O Especialista 2 recomendou explicitar a decisão entre concluir o processo e retomar a fase correspondente à lacuna.	Foi incluído um gateway após a Fase 4, direcionando o fluxo para a conclusão ou para a retomada da Fase 2 ou da Fase 3.
Autoavaliação	Adequação dos momentos destinados à autoavaliação	A	PA	O Especialista 2 recomendou aplicar a autoavaliação nas Fases 2, 3 e 4.	A autoavaliação do time foi respondida nas Fases 2 e 3, e suas respostas foram retomadas na análise da Fase 4, sem nova aplicação.
Progressão pedagógica	Organização gradual das atividades	A	PA	O Especialista 2 recomendou introduzir as atividades progressivamente, sem antecipar conteúdos.	As atividades foram organizadas do nível introdutório ao mais complexo, respeitando a sequência dos conteúdos da disciplina.
Instrumentos de acompanhamento	Quantidade, pertinência e viabilidade dos instrumentos	NA	A	O Especialista 1 recomendou criar um instrumento de acompanhamento individual para cada aluno.	A recomendação não foi incorporada, mantendo-se o acompanhamento por time.
Alinhamento curricular	Correspondência das atividades com a ementa da disciplina	NA	A	O Especialista 1 recomendou adaptar as atividades à ementa.	As atividades foram adaptadas à ementa da disciplina.

Nota: E1 = Especialista 1; E2 = Especialista 2; A = atendido; PA = parcialmente atendido; NA = não atendido.

Fonte: Autoria própria.

Os ajustes incorporados a partir dessa apreciação foram adicionados ao planejamento definitivo e serviram de base para a etapa seguinte da pesquisa. Essa etapa correspondeu a um novo estudo, realizado com participantes distintos daqueles envolvidos na apreciação do *framework*, envolvendo dois professores e suas respectivas turmas, com o objetivo de analisar a aplicação do SoPensa em ambiente real de ensino.

6.2 Contexto e caracterização do ambiente de estudo

A pesquisa centra-se na aplicação do *framework* SoPensa pelos docentes, no ensino de programação da EPTNM. O objeto principal de análise é o professor e sua prática pedagógica, e os dados coletados com os estudantes complementam essa análise, oferecendo

uma leitura qualitativa sobre como os componentes do PC e as competências de SS foram mobilizados ao longo das atividades.

A pesquisa foi realizada no IFAC, autarquia vinculada ao MEC, criada pela Lei nº 11.892, de 29 de dezembro de 2008. O IFAC tem como missão promover a educação profissional, científica e tecnológica de qualidade, pautada em valores como ética, inclusão e inovação. Dentre as unidades que compõem a instituição, foi selecionado o *Campus* Rio Branco, por oferecer cursos técnicos na área de Informática e pelo fácil acesso do pesquisador ao ambiente de estudo. O *campus* dispõe de infraestrutura adequada e corpo docente especializado, condições que contribuíram para a aplicação do *framework* SoPensa em turmas do curso técnico integrado ao ensino médio.

A intervenção ocorreu ao longo do primeiro semestre letivo de 2025. Os professores conduziram as atividades com autonomia para adaptá-las à prática regular e o pesquisador participou das aulas, junto aos professores, como apoio técnico e metodológico e com o registro contínuo das observações.

6.2.1 Professores participantes

Os professores participantes foram selecionados com base nos resultados de um estudo prévio e em critérios relacionados à formação e à experiência docente. Para participar da pesquisa, os docentes deveriam ser graduados em Computação, pertencer ao quadro efetivo do IFAC, possuir, no mínimo, três anos de experiência no ensino de Lógica de Programação e concordar em participar mediante a assinatura do Termo de Consentimento Livre e Esclarecido. O convite foi enviado por *e-mail* institucional. Dois docentes atenderam aos critérios estabelecidos e participaram da aplicação.

A adoção desses critérios permitiu que a aplicação fosse conduzida por professores com experiência no ensino de Lógica de Programação, contribuindo para a adequada execução das atividades previstas. A Tabela 18 caracteriza os participantes quanto ao curso em que atuavam, à formação acadêmica, à disciplina lecionada e ao tempo de experiência docente.

Tabela 18: Caracterização dos professores participantes

Sujeito	**Curso**	**Formação**	**Disciplina**	**Atuação no curso**
Professor 1	Informática em Redes	Ciência da Computação	Lógica de Programação	5 anos
Professor 2	Informática para Internet	Sistemas de Informação	Lógica de Programação	8 anos

Fonte: Autoria própria.

6.2.2 Descrição das turmas

Para a aplicação do *framework* em sala de aula, foram selecionadas duas turmas do primeiro ano do curso técnico integrado ao ensino médio, uma de Informática para Internet e outra de Informática em Redes. Cada turma possuía 35 estudantes regularmente matriculados, com idades entre 15 e 16 anos. A escolha dessas turmas ocorreu porque os estudantes ainda não haviam participado de atividades que integrassem PC e SS ao ensino de programação, possibilitando a aplicação do *framework* em uma etapa introdutória da formação técnica.

6.2.3 Condições de aplicação

A disciplina de Lógica de Programação integra o primeiro ano dos dois cursos. O *framework* SoPensa foi aplicado no primeiro semestre, durante o desenvolvimento dos conteúdos básicos comuns às duas turmas, que incluíram lógica proposicional e algoritmos. Conteúdos específicos, como Shell Script, no curso de Informática em Redes, são ministrados no segundo semestre e não integraram a intervenção. Dessa forma, as atividades foram desenvolvidas com os mesmos conteúdos nas duas turmas. As ementas completas encontram-se nos Anexos A e B.

Para a aplicação do *framework*, foram utilizados computadores, projetor multimídia, quadro branco e materiais complementares. O apoio institucional compreendeu a autorização da Direção-Geral para a participação das turmas, o acompanhamento da coordenação do curso e a disponibilização da infraestrutura necessária para a realização das atividades. Para preservar o anonimato nos registros da aplicação em sala, os estudantes foram identificados por códigos atribuídos por turma, de A01 a A35 em cada curso. Os respondentes do questionário final receberam identificação própria, descrita no Capítulo 7.

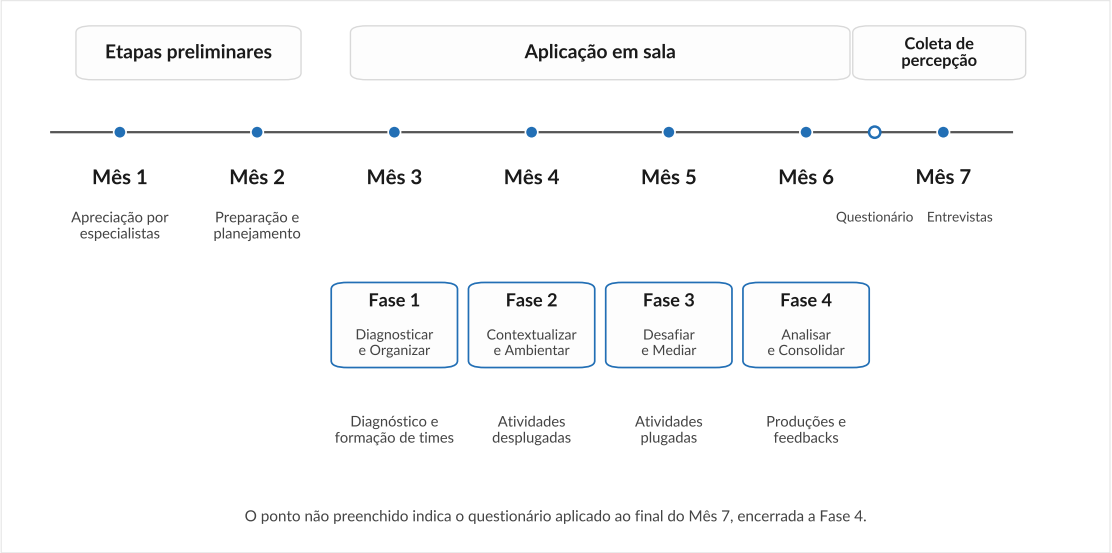
6.3 Relato de aplicação do *framework* SoPensa

Esta seção relata a experiência de aplicação do *framework* SoPensa em sala de aula. Por se tratar de uma aplicação concomitante à disciplina, o pesquisador participou das aulas, em apoio aos professores, com atuação de natureza técnica e operacional. Na Fase 1, diante da ausência de um sistema automatizado, o pesquisador tabulou os questionários diagnósticos e executou o algoritmo de formação dos times, cuja composição foi consentida

pelos professores antes do prosseguimento. Na Fase 2, esclareceu dúvidas dos professores quanto ao formato de aplicação das atividades. Na Fase 3, apoiou o alinhamento entre os conteúdos da disciplina e as atividades do *framework*, cabendo aos professores a decisão sobre como e quando integrá-los. A condução das aulas permaneceu sob responsabilidade dos professores, enquanto o pesquisador acompanhou o processo e registrou as observações em diário reflexivo.

A Figura 26 sintetiza a linha do tempo da intervenção, das etapas preliminares à coleta de percepção, com a aplicação em sala distribuída nas quatro fases do *framework*, do Mês 4 ao Mês 7 da linha do tempo. Inicialmente, descrevem-se a preparação da equipe de trabalho e o planejamento das atividades; em seguida, apresenta-se a aplicação organizada nas quatro fases que compõem o *framework*, considerando as ações realizadas e os resultados imediatos observados em cada fase.

Figura 26: Síntese da linha do tempo da aplicação do SoPensa



Fonte: Autoria própria.

A aplicação ocorreu de forma pontual e integrada ao desenvolvimento dos conteúdos da disciplina, não ocupando todas as aulas previstas no semestre. Na experiência realizada, cada fase demandou aproximadamente três a quatro aulas, com variações entre as turmas em função da disponibilidade de encontros, do planejamento dos professores e do andamento das atividades. Esse intervalo corresponde à duração observada nesta aplicação e constitui uma referência, e não uma duração fixa para a utilização do *framework*, uma vez que o ritmo de cada fase pode ser ajustado às características da turma, aos conteúdos trabalhados e às necessidades dos estudantes.

6.3.1 Preparação da equipe de trabalho

Confirmada a participação dos dois professores, o pesquisador reuniu-se com os docentes, no Mês 3, para apresentar o funcionamento do SoPensa.

Realizaram-se duas reuniões, com duração aproximada de 40 minutos cada, nas quais foram apresentados os componentes do PC, as SS, as fases e os fluxos de atividades do *framework*. Nessa ocasião, ficou definido que os professores conduziriam as aulas, enquanto o pesquisador acompanharia a aplicação, apoiando os docentes e observando o processo. Essa preparação inicial foi importante para alinhar expectativas, esclarecer dúvidas e assegurar que a aplicação ocorresse de forma colaborativa e organizada.

6.3.2 Planejamento das atividades

O planejamento iniciou-se ainda no Mês 3, em alinhamento com os professores de Lógica de Programação. Foram apresentados o questionário diagnóstico, os conteúdos relacionados ao PC e às SS, as atividades plugadas e desplugadas, os jogos e as situações-problema.

A partir desse processo, os professores adaptaram os conceitos e as atividades à organização de suas disciplinas, de modo a integrá-los ao plano de ensino sem comprometer a ementa anual do projeto pedagógico do curso. O calendário foi estruturado para um semestre letivo, conforme a recomendação dos professores e a carga horária da disciplina. E parte das aulas foi destinada às atividades do SoPensa, para que os componentes do PC e as competências de SS fossem trabalhadas de forma gradual e contextualizada.

Além da infraestrutura já descrita anteriormente, o planejamento previu a confecção de materiais impressos para as atividades desplugadas. Também foram organizados os instrumentos de diagnóstico e de acompanhamento, como o questionário diagnóstico, a ficha de observação das fases, a rubrica de acompanhamento dos times e a autoavaliação do time, que permitiram acompanhar o desenvolvimento das etapas de forma sistemática.

6.3.3 Fase 1: Diagnosticar e Organizar

A Fase 1 iniciou-se na primeira semana do Mês 4, quando o professor, previamente orientado, aplicou um questionário diagnóstico para levantar informações sobre os componentes do PC, as SS e os conhecimentos prévios de programação. O instrumento, apresentado no Apêndice E, é composto por onze questões: uma de identificação do curso, uma sobre o contato prévio com programação, quatro relacionadas aos componentes do PC, quatro

voltadas às competências de SS e uma questão aberta para complementações.

A aplicação ocorreu no primeiro dia de encontro da disciplina, com a participação de 70 estudantes, sendo 35 do curso técnico integrado em Informática para Internet e 35 do curso técnico integrado em Informática em Redes. Esse tipo de diagnóstico inicial contribui para identificar percepções, conhecimentos prévios e lacunas de aprendizagem, oferecendo suporte ao planejamento pedagógico (Ausubel, 2003).

6.3.3.1 Resultados do diagnóstico por turma

Após a aplicação do questionário, as respostas foram classificadas em duas categorias: adequado e em desenvolvimento. Como o questionário era composto por questões objetivas, adotou-se um critério direto de correção. As respostas corretas foram classificadas como adequado e receberam o valor 1, enquanto as incorretas foram classificadas como em desenvolvimento e receberam o valor 0. Essa quantificação simples permitiu caracterizar o perfil das turmas e orientar a formação equilibrada dos times. A Tabela 19 apresenta os resultados das respostas por componente do PC e por SS em cada turma.

Tabela 19: Distribuição dos estudantes por componente do PC e competência de SS

Componente / Competência	**Inf. para Internet**		**Inf. em Redes**	
	Adequado	**Em desenv.**	**Adequado**	**Em desenv.**
Componentes do PC				
Decomposição	19	16	18	17
Reconhecimento de padrões	30	5	28	7
Abstração	9	26	10	25
Algoritmos	18	17	15	20
Competências de SS				
Colaboração	22	13	29	6
Comunicação	14	21	15	20
Criatividade	18	17	12	23
Pensamento crítico	25	10	14	21

Fonte: Autoria própria.

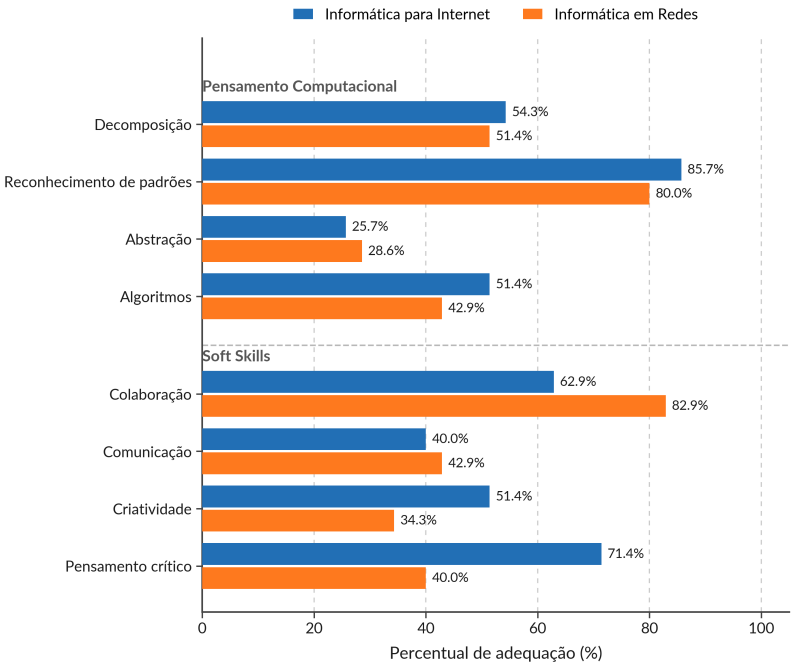
A leitura dos dados foi realizada por turma, de forma descritiva, uma vez que esta etapa diagnóstica teve a finalidade de caracterizar o perfil inicial das turmas, e não de comparar os cursos ou o desempenho dos estudantes. Também se perguntou sobre o contato prévio com programação. Poucos estudantes relataram experiência anterior, em geral em atividades de robótica educacional, sendo cinco em Informática para Internet e quatro em Informática em Redes. Essa informação foi considerada na composição dos times, distribuindo os estudantes com experiência entre as equipes.

6.3.3.2 Distribuição percentual de PC e SS por turma

A Figura 27 apresenta a distribuição percentual dos componentes do PC e das competências de SS em cada turma, destacando os pontos fortes e as fragilidades que orientaram a formação dos times. O Reconhecimento de padrões foi o componente com maior nível de adequação nas duas turmas (85,7% e 80,0%), enquanto a Abstração apresentou os menores índices (25,7% e 28,6%), resultado que dialoga com estudos que apontam essa dificuldade entre estudantes iniciantes (Qian; Lehman, 2017; Cheah, 2020).

Entre as competências de SS, o Pensamento crítico destacou-se em Informática para Internet (71,4%), enquanto a Colaboração apresentou o maior índice em Redes (82,9%). Embora não tivessem caráter avaliativo individual, esses dados ofereceram ao professor uma visão ampliada sobre quais estudantes apresentavam maior domínio em determinadas habilidades e quais estudantes necessitavam de maior apoio dos colegas durante as aulas.

Figura 27: Distribuição dos componentes de PC e competências de SS por turma



Fonte: Autoria própria.

6.3.3.3 Formação dos times equilibrados

A formação dos times teve finalidade exclusivamente pedagógica, sem caráter avaliativo individual. O objetivo foi favorecer condições equilibradas de trabalho colaborativo, preservar a heterogeneidade interna dos perfis e oferecer ao professor um panorama de cada turma. A formação baseou-se nas respostas do questionário diagnóstico, codificadas como

1, para desempenho “adequado”, e 0, para desempenho “em desenvolvimento”, o que possibilitou um tratamento quantitativo simples e formativo.

Cada estudante i recebeu o valor $x_{i,d} \in \{0,1\}$ em cada uma das oito dimensões d avaliadas, sendo quatro componentes do PC e quatro competências de SS. As médias individuais de PC e de SS foram dadas por:

$$M_{PC}^{(i)} = \frac{1}{4}\sum_{c=1}^{4} x_{i,c}^{PC}, \qquad M_{SS}^{(i)} = \frac{1}{4}\sum_{k=1}^{4} x_{i,k}^{SS}. \tag{6.1}$$

A média de cada time t, com $n_t = 5$ integrantes, foi obtida pela combinação entre a média de PC e a média de SS:

$$M_{PC}^{(t)} = \frac{1}{n_t}\sum_{i\in t} M_{PC}^{(i)}, \quad M_{SS}^{(t)} = \frac{1}{n_t}\sum_{i\in t} M_{SS}^{(i)}, \quad M_{Geral}^{(t)} = \frac{M_{PC}^{(t)} + M_{SS}^{(t)}}{2}. \tag{6.2}$$

A composição dos times seguiu dois critérios, em ordem de prioridade. O primeiro, denominado cobertura por habilidade, buscou assegurar que cada time tivesse ao menos um integrante com nível adequado em cada dimensão analisada:

$$\sum_{i\in t} x_{i,d} \geq 1, \qquad \forall\, t,\, \forall\, d \in \{1,\ldots,8\}. \tag{6.3}$$

Essa condição indica que, para cada time (t) e para cada uma das oito dimensões avaliadas (d), deveria haver pelo menos um estudante com desempenho adequado. Assim, procurou-se evitar a formação de times sem nenhum integrante com domínio inicial em determinada dimensão.

O segundo critério, denominado equilíbrio entre os times, buscou reduzir a variação entre as médias dos times:

$$\Delta_{PC} = \max_t M_{PC}^{(t)} - \min_t M_{PC}^{(t)}, \qquad \Delta_{SS} = \max_t M_{SS}^{(t)} - \min_t M_{SS}^{(t)}. \tag{6.4}$$

Na prática, os estudantes foram ordenados pela média de SS, com a média de PC como critério de desempate, e distribuídos pelo método *round-robin*. Em seguida, aplicaram-se trocas pontuais ($1 \leftrightarrow 1$) entre times, que primeiro eliminam eventuais lacunas de cobertura e depois reduzem a dispersão das médias até a menor variação encontrada. O procedimento contribuiu para preservar a heterogeneidade interna dos times e favorecer

o equilíbrio entre eles.

As Tabelas 20 e 21 apresentam os times formados em cada curso, com suas médias.

Tabela 20: Times do curso de Informática para Internet

Time	Integrantes	Média PC	Média SS	Média Geral
T1	A01, A26, A27, A31, A33	0,55	0,55	0,55
T2	A02, A10, A16, A20, A23	0,55	0,55	0,55
T3	A03, A07, A25, A34, A35	0,55	0,55	0,55
T4	A04, A06, A17, A19, A30	0,55	0,60	0,58
T5	A05, A08, A11, A22, A32	0,55	0,55	0,55
T6	A09, A13, A14, A21, A24	0,55	0,60	0,58
T7	A12, A15, A18, A28, A29	0,50	0,55	0,53

Fonte: Autoria própria.

Tabela 21: Times do curso de Informática em Redes

Time	Integrantes	Média PC	Média SS	Média Geral
T1	A01, A02, A16, A25, A35	0,50	0,50	0,50
T2	A03, A11, A14, A20, A22	0,50	0,50	0,50
T3	A04, A13, A23, A27, A31	0,50	0,50	0,50
T4	A05, A09, A10, A26, A30	0,50	0,50	0,50
T5	A06, A12, A17, A18, A24	0,50	0,50	0,50
T6	A07, A08, A15, A19, A32	0,55	0,50	0,53
T7	A21, A28, A29, A33, A34	0,50	0,50	0,50

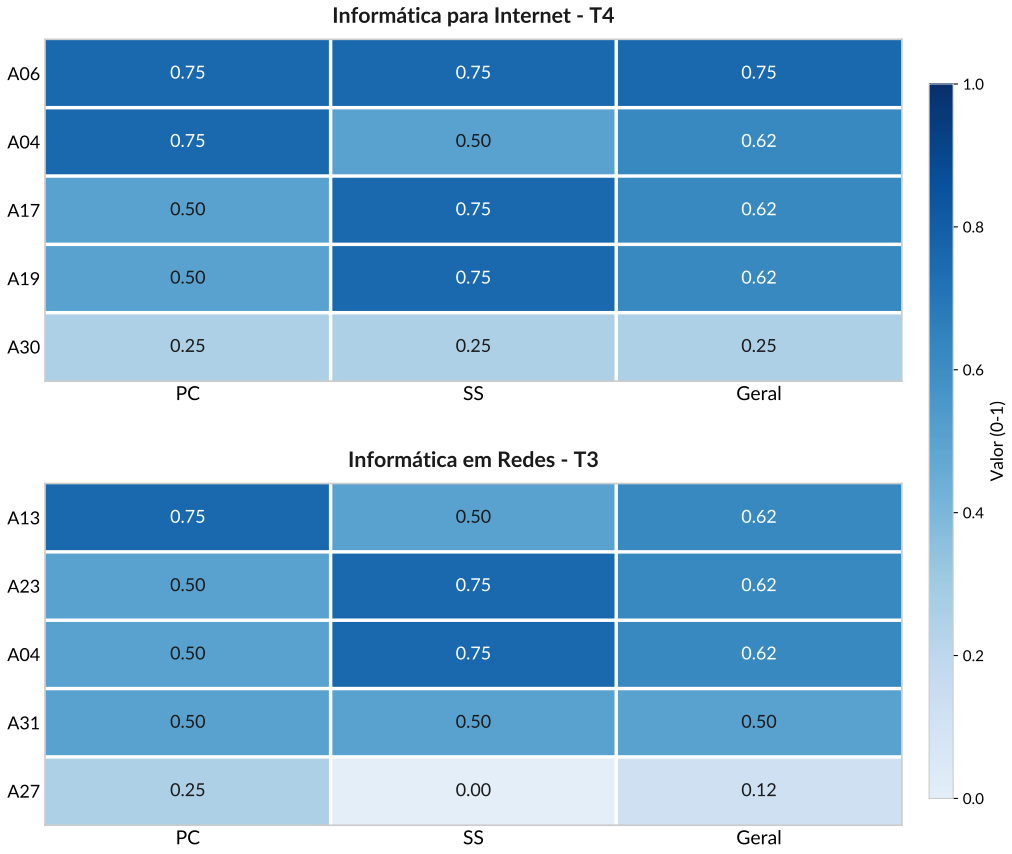
Fonte: Autoria própria.

Em cada turma, as diferenças entre as médias dos times foram pequenas. Em Informática para Internet, obtiveram-se $\Delta_{PC} = 0{,}05$ e $\Delta_{SS} = 0{,}05$, com variação máxima de 0,05 nas médias gerais; em Informática em Redes, $\Delta_{PC} = 0{,}05$ e $\Delta_{SS} = 0{,}00$, com variação máxima de 0,025 nas médias gerais. Esses valores indicam o equilíbrio obtido na formação dos times. Além disso, cada time reuniu ao menos um integrante com nível adequado em cada componente do PC e em cada competência de SS, o que favorece o apoio entre pares durante as atividades.

6.3.3.4 Mapa de calor dos componentes de PC e competências de SS por time

Para mostrar a distribuição interna das dimensões, elaborou-se um mapa de calor com os resultados individuais dos estudantes em PC, SS e média geral. A Figura 28 apresenta dois times, um de cada curso, selecionados aleatoriamente apenas para ilustrar a distribuição interna das dimensões, sem critério de desempenho. Cada linha corresponde a um estudante, e as colunas representam as três medidas analisadas: PC, SS e média geral. A escala de cores varia de 0 a 1, e os valores nas células indicam os resultados obtidos.

Figura 28: Mapa de calor dos componentes de PC e das competências de SS em dois times representativos



Fonte: Autoria própria.

A representação mostra a heterogeneidade interna dos times, reunindo integrantes com maior e menor domínio em um mesmo grupo, mas com médias coletivas próximas. Essa composição contribuiu para que, nas fases seguintes, o foco recaísse sobre o desenvolvimento dos componentes do PC e das competências de SS e não sobre as diferenças prévias. As médias individuais e os mapas de calor de todos os times constam no Apêndice F.

6.3.3.5 Acompanhamento da fase

Durante a Fase 1, os professores utilizaram a Ficha de Observação das Fases para registrar a aplicação do diagnóstico e a organização dos times. Os registros documentaram a participação dos estudantes na atividade inicial e serviram de linha de base para o acompanhamento das fases seguintes. A ficha encontra-se no Apêndice E.

Com base no diagnóstico, os professores compreenderam melhor o perfil das turmas e ajustaram as estratégias didáticas. Essas informações orientaram o início da Fase 2, dedicada à introdução prática dos conceitos.

6.3.4 Fase 2: Contextualizar e Ambientar

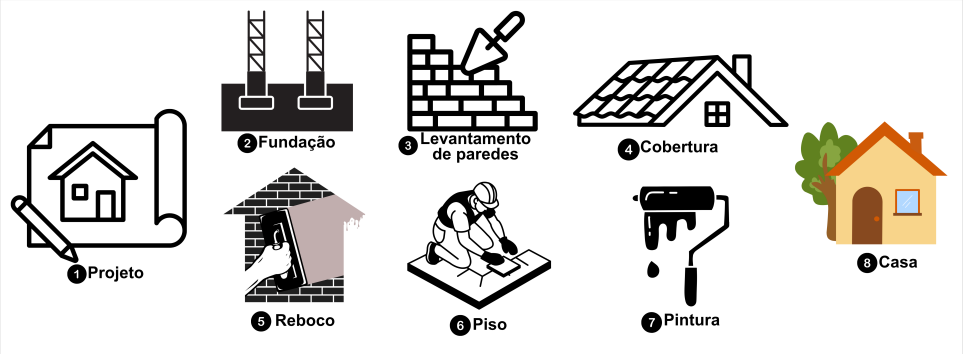
Concluída a Fase 1, o diagnóstico inicial permitiu aos professores compreender o perfil das turmas e organizar times equilibrados, orientando o planejamento das etapas seguintes do *framework*. A Fase 2 marcou o início efetivo da aplicação do SoPensa em sala de aula, momento em que os professores introduziram os fundamentos dos componentes do PC e das competências de SS por meio de atividades dialogadas e vivências desplugadas. A opção por iniciar sem o uso do computador atendeu ao perfil observado no diagnóstico, uma vez que os estudantes apresentavam pouca experiência prévia em programação e precisavam construir uma base comum antes de lidar com código. Na elaboração das atividades, partiu-se de situações ligadas à realidade dos estudantes, incorporando, quando pertinente, elementos regionais, de modo a tornar os problemas mais próximos e contextualizados.

A mediação docente na Fase 2 apoiou-se no ciclo formativo do *framework*, organizado nas etapas Pensar, Agir e Refletir. Por se tratar de uma etapa desplugada e de ambientação, a comunicação entre os integrantes ocorreu internamente aos times, principalmente por meio da discussão dos resultados, sem se constituir como etapa formal de apresentação à turma. As aulas combinaram exposição dialogada, dinâmicas coletivas e desafios físicos, em um ambiente de aprendizagem fundamentado em princípios construcionistas e na aprendizagem significativa (Papert, 1980; Ausubel; Novak; Hanesian, 1980).

6.3.4.1 Planejamento e estratégias didáticas

O ensino dos componentes do PC e das competências de SS começou por meio de duas aulas dialogadas, cada uma dedicada a dois componentes do PC, conduzidas a partir do exemplo cotidiano da construção de uma casa. A decomposição foi trabalhada pela divisão do processo de construção em etapas menores, desde o projeto até o acabamento, como apresenta a Figura 29.

Figura 29: Processo de construção de uma casa: exemplo de decomposição



Fonte: Autoria própria.

O reconhecimento de padrões foi explorado a partir da representação da mesma casa em versões de cores diferentes, que indicaram a permanência da estrutura apesar das variações, conforme a Figura 30.

Figura 30: Reconhecimento de padrões a partir de um modelo base

Fonte: Autoria própria.

A abstração foi explorada a partir da pergunta sobre quais elementos seriam necessários para que alguém reconhecesse uma casa, acompanhada por uma sequência de imagens com simplificação progressiva, da casa detalhada à forma essencial, conforme apresentada na Figura 31.

Figura 31: Abstração: simplificação progressiva

Fonte: Autoria própria.

Os algoritmos foram introduzidos pela organização das etapas de construção da casa em uma sequência de comandos, representada em Portugol, conforme apresentada na Figura 32.

Figura 32: Algoritmo em Portugol

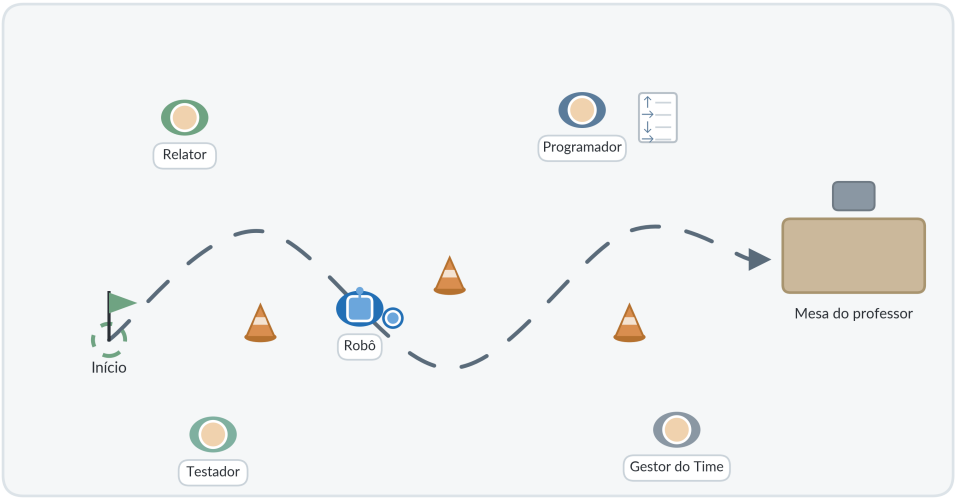
Fonte: Autoria própria.

As competências de SS, colaboração, comunicação, criatividade e pensamento crítico, foram apresentadas em uma aula expositiva ao final do ciclo dialogado, com exemplos do cotidiano profissional do programador, e relacionadas aos componentes do PC já trabalhados. O planejamento priorizou a integração entre os componentes do PC e essas competências, enquanto os professores acompanharam o progresso das turmas por meio de intervenções e ajustes conforme as necessidades identificadas em sala.

6.3.4.2 Atividade desplugada: Robô HidroLógico

Ao final das aulas dialogadas, os professores propuseram a primeira atividade desplugada da fase, denominada Robô HidroLógico, aplicada nas turmas de Informática para Internet e de Informática em Redes. A atividade foi planejada para mobilizar, em um mesmo exercício, os quatro componentes do PC e as competências de SS. Cada time, com cinco integrantes, conduziu um copo de água do ponto inicial até a mesa do professor por meio de instruções escritas pelos próprios estudantes. Um colega assumiu o papel de robô-executor, cumprindo os comandos sem improvisar. O percurso incluiu obstáculos demarcados no chão, que precisavam ser contornados, e cada erro de trajeto exigia que o time refizesse o algoritmo. A Figura 33 apresenta o esquema da atividade, no qual o time programava os comandos para percorrer o trajeto, contornar os obstáculos e levar o copo de água até a mesa do professor, com cada papel posicionado no espaço da sala.

Figura 33: Esquema da atividade Robô HidroLógico



Fonte: Autoria própria.

A atividade foi aplicada com os times equilibrados formados na Fase 1, os quais foram mantidos durante a Fase 2. Os cinco papéis do time foram definidos antes da execução, com rodízio a cada aula. O planejamento e a decomposição do percurso foram realizados coletivamente, sob coordenação do Gestor do Time, enquanto o Programador ficou responsável por converter o plano coletivo em comandos. A Tabela 22 descreve os papéis e suas responsabilidades.

Tabela 22: Papéis e responsabilidades na atividade Robô HidroLógico

Papel	**Responsabilidade principal**
Relator	Registrou e sintetizou as ideias, descreveu as estratégias e relatou as dificuldades e os avanços do time.
Programador	Converteu o plano coletivo em uma sequência de comandos precisos e ordenados (andar, virar, pegar, soltar, retornar), prontos para execução.
Robô	Executou exatamente os comandos escritos, sem improvisar, e conduziu o copo de água sem derramá-lo ao longo do percurso.
Testador	Leu as instruções em voz alta, acompanhou a execução e identificou erros, ambiguidades e inconsistências nos comandos.
Gestor do Time	Coordenou o planejamento coletivo, controlou o tempo, organizou as falas, assegurou a participação de todos e conduziu a revisão do algoritmo após falhas.

Fonte: Autoria própria.

A Tabela 23 organiza as etapas da atividade, com a descrição de cada uma e os componentes do PC e as competências de SS trabalhados.

Tabela 23: Etapas da atividade Robô HidroLógico

Etapa	**Descrição**	**Componentes do PC e competências de SS**
1. Planejamento	O time analisou o ambiente, identificou os obstáculos e dividiu o percurso em etapas menores. O Gestor do Time coordenou a discussão e a definição coletiva da rota.	Decomposição; colaboração; criatividade.
2. Programação	O Programador converteu o plano coletivo em uma sequência de comandos em linguagem natural. O Testador revisou os comandos para eliminar ambiguidades.	Algoritmos; abstração; pensamento crítico.
3. Execução	O Robô executou as instruções. Em caso de erro ou derramamento, o time pausou, revisou e corrigiu o algoritmo.	Algoritmos; colaboração; comunicação.
4. Retorno ao ponto inicial	O time inverteu a lógica do percurso, o que exigiu a compreensão de repetição e reversão de comandos.	Reconhecimento de padrões; abstração; algoritmos.
5. Revisão e relato	O Gestor do Time conduziu uma revisão oral sobre o que funcionou e o que poderia melhorar. O Relator registrou as observações e sintetizou as estratégias adotadas.	Comunicação; pensamento crítico.

Fonte: Autoria própria.

As etapas percorreram o ciclo formativo da fase: o planejamento e a programação corresponderam a Pensar; a execução e o retorno, a Agir; e a revisão final, a Refletir. A atividade foi realizada em sala de aula, com o trajeto demarcado no chão e os obstáculos dispostos ao longo do percurso. Ao final, cada time validou sua solução pela execução do trajeto. Os registros no diário reflexivo e na Rubrica de Acompanhamento dos Times descreveram, nas duas turmas, como os times planejaram o percurso, identificaram repetições e organizaram a sequência de comandos. Em diversas situações, os erros de trajeto levaram os times a revisar e corrigir o algoritmo antes de uma nova tentativa, reforçando a relação entre a ordem dos comandos e o resultado da execução.

6.3.4.3 Atividade desplugada: Caixas de Missões Lógicas

A segunda atividade desplugada da fase foi o jogo Caixas de Missões Lógicas, voltado à retomada da lógica proposicional, conteúdo inicial comum às duas turmas no primeiro semestre. Após o trabalho com os componentes do PC, esse tema foi retomado em uma dinâmica prática centrada em proposições, conectivos lógicos e condições de verdade. Foram confeccionadas sete caixas, uma por time, cada uma contendo as mesmas quatro missões contextualizadas com situações do cotidiano: Alavanca da Porta, O Cofre Silencioso, Alerta no Laboratório e Código da Mente. As Caixas de Missões Lógicas foram desenvolvidas especificamente para esta pesquisa como recurso pedagógico do *framework* SoPensa. Cada time abria uma missão por vez e só avançava após a validação da solução

pelo professor. A atividade durou cerca de duas aulas corridas de 45 minutos, e o time que resolveu o maior número de desafios recebeu uma premiação simbólica. A Figura 34 apresenta as caixas e o percurso das missões até a abertura do cofre lógico.

Figura 34: Caixas de Missões Lógicas e percurso até o cofre lógico

Fonte: Autoria própria.

A Tabela 24 organiza as etapas da atividade, com a descrição de cada uma e os componentes do PC e as competências de SS mobilizados.

Tabela 24: Etapas da atividade Caixas de Missões Lógicas

Etapa	**Descrição**	**Componentes do PC e competências de SS**
1. Leitura e decomposição da missão	O time leu o enunciado, identificou as condições envolvidas e dividiu a missão em proposições menores.	Decomposição; colaboração; comunicação.
2. Reconhecimento e representação	O time identificou semelhanças entre as proposições e selecionou as informações essenciais para representar a missão.	Reconhecimento de padrões; abstração; criatividade.
3. Construção da tabela-verdade	O time organizou as proposições e os conectivos em uma sequência lógica e construiu a tabela-verdade.	Algoritmos; abstração; pensamento crítico.
4. Justificativa e validação	O time justificou a solução, registrou o raciocínio e submeteu a resposta à validação do professor.	Comunicação; pensamento crítico.

Fonte: Autoria própria.

Os estudantes responderam às missões em folhas de papel, nas quais registraram as proposições, construíram as tabelas-verdade e justificaram a lógica utilizada. As perguntas foram organizadas em passos sucessivos, apoiando um raciocínio progressivo. O mesmo ciclo formativo da fase também foi observado na atividade, de modo que a leitura, a decomposição e a representação das missões corresponderam a Pensar; a construção das tabelas-verdade, a Agir; e a justificativa das soluções, a Refletir. Os registros no diário reflexivo e na Rubrica de Acompanhamento dos Times indicaram que a decomposição e

o reconhecimento de padrões apareceram com mais frequência nas resoluções, enquanto a abstração já se mostrava difícil nessa etapa, pois parte dos times encontrou obstáculos para selecionar as informações essenciais antes de montar as tabelas-verdade.

Um momento da resolução em sala aparece na Figura 35.

Figura 35: Estudantes durante a atividade Caixas de Missões Lógicas

Fonte: Autoria própria.

6.3.4.4 Acompanhamento da fase

Ao longo da fase, o acompanhamento dos times apoiou-se nos três instrumentos do *framework*. A Rubrica de Acompanhamento dos Times registrou o desenvolvimento dos componentes do PC e das competências de SS durante as atividades Robô HidroLógico e Caixas de Missões Lógicas. A Ficha de Observação das Fases foi preenchida pelos professores ao final da etapa, com o registro do que funcionou, do que não funcionou e dos ajustes necessários. A Autoavaliação do Time, aplicada no encerramento, reuniu a percepção dos integrantes sobre a organização interna, a participação no time e a própria aprendizagem, oferecendo ao professor uma leitura que a observação externa, isoladamente, não alcançava. Em paralelo, o diário reflexivo do pesquisador acompanhou as duas atividades como registro de campo, com anotações sobre as interações e a mediação em sala. Os modelos dos três instrumentos constam no Apêndice E. A conclusão da Fase 2 contribuiu para consolidar uma base comum de PC e SS nas duas turmas e preparou a transição para as atividades plugadas, nas quais os times passaram a desenvolver soluções por meio da programação.

6.3.5 Fase 3: Desafiar e Mediar

Com a base comum de PC e SS fortalecida na fase anterior, a Fase 3 levou os estudantes a desafios plugados de complexidade crescente, nos quais os conceitos da ementa passaram a ser aplicados em código. Esses desafios seguiram a aprendizagem baseada em problemas, abordagem coerente com a base construcionista do *framework*, que sustenta a aprendizagem pela experimentação e pela construção de soluções (Papert, 1980). Foram trabalhados tipos de dados, variáveis e constantes, entrada e saída de dados, algoritmos e operadores aritméticos, relacionais e lógicos, estes últimos com a aplicação da lógica proposicional reforçada nas atividades desplugadas da Fase 2. A fase reuniu duas práticas plugadas em sequência, primeiro no Scratch, com programação por blocos visuais, e depois em Python, com programação textual. A passagem do visual para o textual buscou aproximar os estudantes, de forma gradual, da resolução de problemas em programação (Resnick *et al.*, 2009). Tudo ocorreu nos laboratórios de informática da instituição, com os computadores usados para acessar os ambientes de programação.

Na Fase 3, a etapa Comunicar ganhou maior visibilidade no ciclo formativo do *framework*. Diferentemente da fase anterior, em que a comunicação ocorreu internamente ao time, nesta fase os estudantes passaram a compartilhar suas soluções com a turma ao final de cada atividade, relatando o processo e justificando as escolhas técnicas. Como as práticas exigiam mais autonomia, esse fechamento coletivo tornou-se adequado ao momento da aplicação. Os times permaneceram os mesmos da Fase 1, sem troca de integrantes e com rodízio de papéis a cada aula, e a desenvoltura crescente dos estudantes nas interações sustentou as apresentações.

6.3.5.1 Preparação para as práticas plugadas

A fase começou por aulas expositivas e dialogadas sobre os conceitos de programação, acompanhadas de exercícios de familiarização. Na sequência, os professores levaram os estudantes ao laboratório, apresentaram o ambiente Scratch e resolveram exemplos práticos, de modo a aproximar a teoria do uso da linguagem visual. Tanto a prática no Scratch quanto a prática em Python percorreram o ciclo do *framework*, nas etapas Pensar, Agir, Refletir e Comunicar. A Tabela 25 mostra como cada etapa integrou os componentes do PC e as competências de SS e vale para as duas atividades.

Tabela 25: Etapas do ciclo formativo seguido nas atividades em Scratch e Python

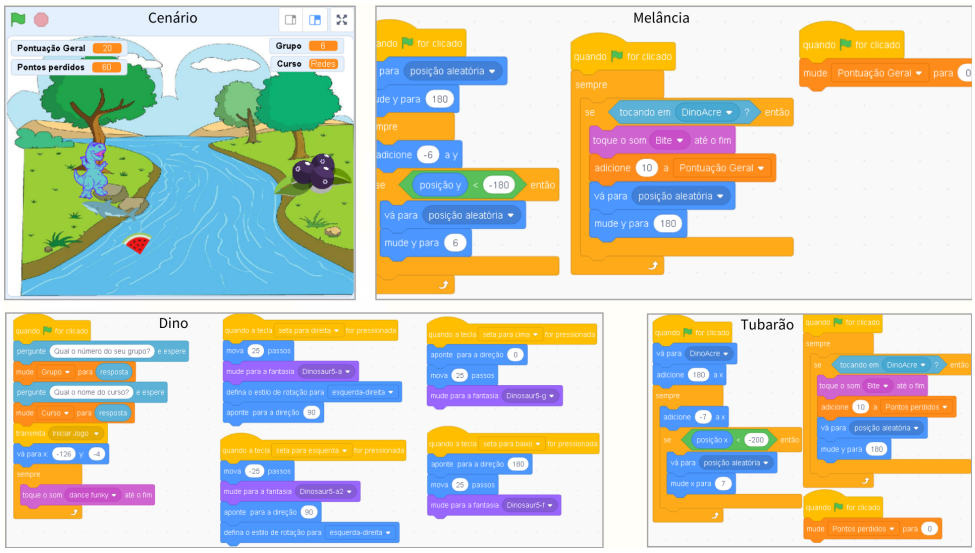
Etapa	**Descrição**	**Componentes do PC e competências de SS**
Pensar	O time analisou o desafio, identificou as entradas, as saídas, as variáveis e as regras, e decompôs o problema.	Decomposição; abstração; colaboração.
Agir	O time implementou a solução no ambiente de programação, reutilizou estruturas semelhantes e testou o funcionamento.	Algoritmos; reconhecimento de padrões; criatividade.
Refletir	O time revisou e corrigiu o código e avaliou as decisões tomadas.	Abstração; pensamento crítico.
Comunicar	O time apresentou a solução à turma, relatou o processo e justificou as escolhas técnicas.	Comunicação; pensamento crítico.

Fonte: Autoria própria.

6.3.5.2 Atividade plugada: A Travessia do Rio Açaí

A primeira prática plugada ocorreu no Scratch, com a proposta "A Travessia do Rio Açaí". O cenário foi elaborado a partir de elementos regionais, como o açaí e o rio, de modo a tornar o problema mais próximo e significativo para os estudantes. Na atividade, o personagem precisava atravessar o rio, coletar itens e desviar de obstáculos até a meta, envolvendo variáveis de pontuação, controle de tempo e condições de vitória e derrota. Ao desenvolver a solução, os times aplicaram entrada e saída de dados, variáveis e operadores lógicos e relacionais. O cenário e os blocos desenvolvidos aparecem na Figura 36.

Figura 36: Projetos desenvolvidos no Scratch: cenário da Travessia do Rio Açaí e blocos de programação



Fonte: Autoria própria.

Pelos registros no diário reflexivo e na rubrica, os algoritmos apresentaram maior organização em comparação às vivências desplugadas da Fase 2, e o reconhecimento de

padrões apareceu com frequência na reutilização de blocos semelhantes. Essa foi a atividade de maior engajamento da fase, com comunicação intensa e colaboração constante entre os integrantes. O apoio visual dos blocos, ao manter parte da lógica visível, favoreceu a elaboração de soluções mais criativas. No fechamento, cada time apresentou seu projeto à turma e explicou as decisões tomadas.

6.3.5.3 Atividade plugada: Desafios em Python

Concluída a etapa visual, os times passaram ao Python, no ambiente Online Python, acessível pelo navegador e sem instalação local. Depois de aulas introdutórias sobre operadores, variáveis e estruturas condicionais, cada time resolveu dois problemas contextualizados, que exigiam o uso conjunto de operadores aritméticos, relacionais e lógicos. A Tabela 26 descreve os dois desafios e suas regras.

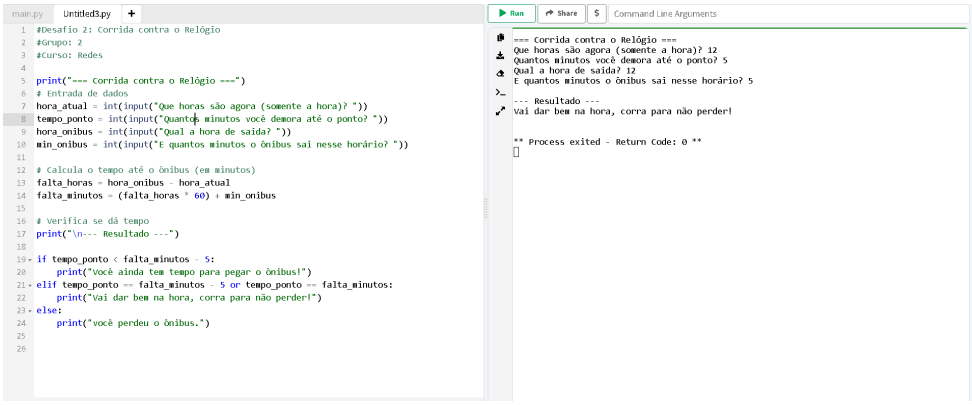
Tabela 26: Desafios propostos em Python na Fase 3

Desafio	**Descrição do problema**	**Regras de resolução**
A Porta Secreta	Criar um sistema de segurança que libere o acesso apenas se a senha for 1234 e o usuário tiver autorização (1 para sim, 0 para não).	O programa solicita a senha e o status de autorização, verifica as duas condições e exibe “Acesso liberado” quando ambas são verdadeiras ou “Acesso negado” caso contrário.
Corrida contra o Relógio	Desenvolver um programa que verifique se o estudante consegue pegar o ônibus escolar com base na hora atual, no tempo até o ponto e no horário de saída.	O programa calcula o horário de chegada, compara com o horário de saída do ônibus, com tolerância de cinco minutos, e informa se o estudante conseguiu ou perdeu o transporte.

Fonte: Autoria própria.

A Figura 37 apresenta um exemplo de programa produzido pelos estudantes na aula de laboratório de informática.

Figura 37: Exemplo de programa desenvolvido pelos estudantes em Python

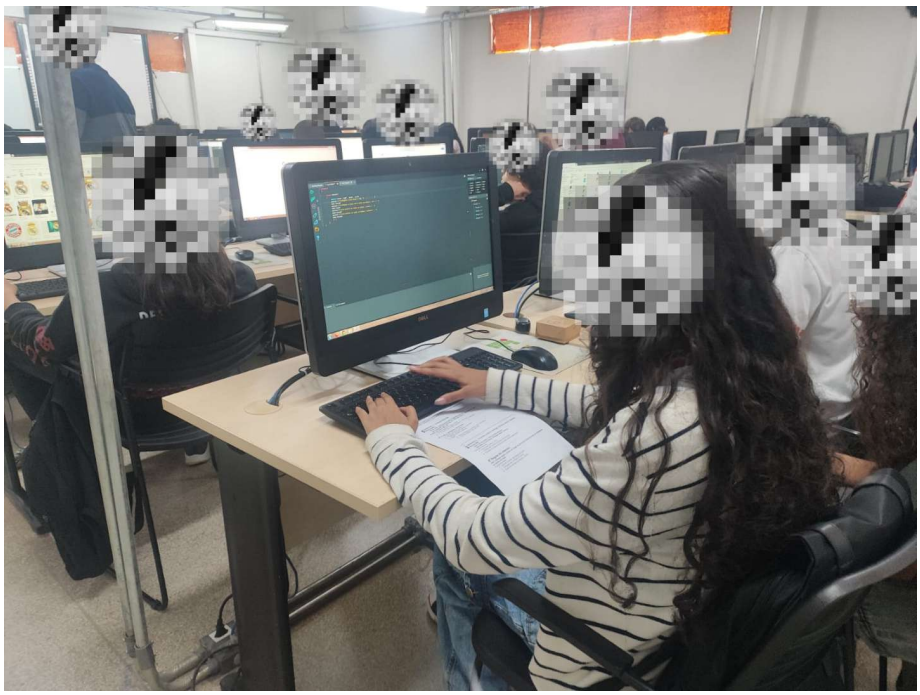


Fonte: Autoria própria.

Sem o apoio dos blocos, os estudantes tiveram de estruturar o problema com mais

autonomia antes de codificá-lo, e foi justamente nessa passagem que as dificuldades se concentraram. Vários times apresentaram códigos com funcionamento parcial, e a mediação docente foi mais requisitada. A abstração se mostrou o ponto mais sensível da fase. Em compensação, a comunicação e a colaboração estiveram mais presentes do que nas atividades desplugadas, sustentadas pela apresentação à turma ao final de cada desafio. Também apareceram diferenças de ritmo entre integrantes de um mesmo time, e alguns times não concluíram o desafio no tempo previsto, o que exigiu mediação mais individualizada. A Figura 38 apresenta os times no laboratório de informática durante a atividade em Python.

Figura 38: Estudantes no laboratório de informática durante a atividade em Python



Fonte: Autoria própria.

6.3.5.4 Acompanhamento da fase

Os três instrumentos de acompanhamento do *framework* foram novamente mobilizados nesta fase. A Rubrica de Acompanhamento dos Times registrou, agora em ambiente de programação, algoritmos mais organizados, maior presença de comunicação e colaboração e a permanência da abstração como dificuldade recorrente. A Autoavaliação do Time, recolhida ao encerramento, trouxe a leitura dos próprios times sobre as competências mobilizadas, enquanto a Ficha de Observação das Fases completou o registro docente, ainda que não tenha sido concluída em uma das turmas. Como apoio à observação, o diário

reflexivo do pesquisador reuniu anotações sobre as interações e a mediação, sustentando a leitura qualitativa da fase.

Encerradas as práticas plugadas, fechou-se a etapa de aplicação do *framework* e o percurso seguiu para a análise das produções dos times na Fase 4.

6.3.6 Fase 4: Analisar e Consolidar

A Fase 4 encerrou o percurso do *framework* e teve natureza distinta das anteriores, pois não propôs novas atividades. Encerrada a aplicação das práticas plugadas, os professores examinaram as entregas dos times e, a partir dessa leitura, conduziram um momento de *feedback* e retomada. Aqui, a mobilização recaiu sobre a comunicação e o pensamento crítico, exercidos no diálogo entre o professor e os times, sem uma nova apresentação pública como a da Fase 3.

6.3.6.1 Análise das produções e retomada

A leitura da situação de cada time apoiou-se nos três instrumentos do *framework*, agora reunidos. A Rubrica de Acompanhamento dos Times indicou em que componentes do PC e em que competências de SS cada time havia avançado ou encontrado dificuldade ao longo das Fases 2 e 3; a Ficha de Observação das Fases trouxe os registros sobre a clareza das atividades, o envolvimento dos estudantes e a mediação em sala; e a Autoavaliação do Time acrescentou a visão interna dos próprios times sobre como haviam se organizado. Esse conjunto permitiu ao professor compreender a situação de aprendizagem de cada time antes da devolução das atividades.

Com base nesse diagnóstico, os professores corrigiram as produções e apontaram as fragilidades. Os times cujas entregas precisavam de ajuste foram reconduzidos à fase correspondente, à Fase 2 quando a lacuna estava na contextualização e nos fundamentos, e à Fase 3 quando dizia respeito à programação e à construção da solução, o que permitiu retomar os componentes do PC e as competências de SS no ponto exato da dificuldade. Os times que voltaram melhoraram suas entregas, enquanto aqueles cujas produções não apresentavam fragilidades seguiram para o encerramento.

Na etapa de retomada, a abstração assumiu um papel diferente do que tivera até então. Se antes fora o componente mais difícil de mobilizar, passou a orientar a maior parte das devoluções, sobretudo nos códigos em que os times não haviam separado com clareza a estrutura essencial do problema. O ciclo de *feedback* permitiu a esses times

elaborar o que a primeira tentativa não havia consolidado.

Para além da leitura das produções, as fichas de observação, preenchidas ao final de cada fase, ajudaram a compreender a condução das atividades em sala. Na Fase 1, os dois professores notaram o entusiasmo das turmas, embora parte dos estudantes ainda se mostrasse tímida e receosa de errar no primeiro contato com as atividades do SoPensa. Na Fase 2, o professor da turma de Informática para Internet registrou que alguns estudantes interpretavam a atividade desplugada como algo separado da disciplina, exigindo maior explicitação de sua relação com a programação; na turma de Redes, os jogos lógicos despertaram maior interesse nos estudantes. Na Fase 3, a principal dificuldade foi a passagem da programação em blocos para o código textual, pois vários estudantes conseguiam montar a lógica em blocos, mas travavam ao escrever o código. Esse registro motivou a sugestão de inserir etapas intermediárias entre o visual e o textual. Na Fase 4, os registros indicaram que alguns estudantes buscavam apenas identificar acertos ou erros, sem refletir sobre os motivos, reforçando a importância do *feedback* dialogado. As fichas também revelaram a sobrecarga do preenchimento manual, a ponto de a ficha de observação da Fase 3 não ter sido concluída em uma das turmas.

6.3.6.2 Autoavaliação dos times

A autoavaliação do Time foi respondida pelos 14 times ao final das Fases 2 e 3, sete em cada curso, considerando os quatro componentes do PC e as quatro competências de SS do *framework*, com respostas em três níveis. A resolução de problemas, de caráter transversal, não integrou o quadro. A Tabela 27 reúne as respostas das duas turmas.

Tabela 27: Autoavaliação dos times por dimensão - síntese das duas turmas

Dimensão	**Fase 2**			**Fase 3**		
	Sim	**Mais ou menos**	**Não**	**Sim**	**Mais ou menos**	**Não**
Componentes do PC						
Decomposição	12	2	0	11	3	0
Reconhecimento de padrões	13	1	0	13	1	0
Abstração	6	8	0	6	5	3
Algoritmos	8	6	0	10	4	0
Competências de SS						
Colaboração	7	6	1	9	5	0
Comunicação	8	6	0	9	5	0
Criatividade	5	9	0	6	6	2
Pensamento crítico	7	6	1	5	9	0

Fonte: Autoria própria.

O reconhecimento de padrões foi o componente com maior número de respostas po-

sitivas, com treze registros em cada uma das duas coletas. Algoritmos e colaboração avançaram entre uma coleta e outra, respectivamente, de oito para dez e de sete para nove respostas afirmativas. A comunicação acompanhou esse movimento. A abstração manteve-se como o ponto mais sensível, sem aumento nas respostas afirmativas e com três respostas negativas na segunda coleta, o maior número entre as dimensões. A criatividade e o pensamento crítico concentraram respostas intermediárias. Esses resultados convergem com os dados do questionário e das rubricas, que situaram a abstração no centro das dificuldades.

A Autoavaliação do Time foi retomada no diálogo de *feedback*, quando o professor recuperou a visão interna de cada time para preparar a devolução. O instrumento revelou um aspecto que a observação docente sozinha não alcançaria. Vários times reconheceram que a participação dos integrantes nem sempre ocorreu de forma equilibrada e que a comunicação interna poderia ter sido mais clara. Esse reconhecimento é coerente com a diferença de ritmo registrada na Fase 3 e, por partir dos próprios times, configura um exercício de pensamento crítico sobre a prática coletiva. Conforme definido após a apreciação dos especialistas, o instrumento não foi reaplicado nesta fase, e a reflexão apoiou-se nas respostas produzidas ao final das Fases 2 e 3.

6.3.6.3 Ajustes ao *framework*

A fase também levou à identificação de ajustes para o próprio *framework*. Três pontos foram apontados como prioritários: a ampliação do tempo destinado às atividades, que em vários momentos se mostrou curto para a conclusão dos desafios; a simplificação das fichas e rubricas, para uso mais ágil durante a aula; e a automatização da formação dos times por meio de um software, de modo a tornar o agrupamento inicial menos trabalhoso para o professor.

Com a análise das produções, a consolidação dos componentes do PC e das competências de SS e os ajustes identificados, a Fase 4 encerrou a aplicação do *framework* SoPensa no contexto das duas turmas.

6.4 Considerações finais do capítulo

Este capítulo descreveu a aplicação do *framework* SoPensa em contexto real de ensino, desde a apreciação por especialistas até a aplicação das quatro fases em duas turmas do primeiro ano de cursos técnicos integrados ao ensino médio. A apreciação prévia permitiu

ajustar a nomenclatura, os fluxos, os instrumentos e a representação do processo antes da entrada em sala, conferindo maior clareza e viabilidade à proposta. Na sequência, a Fase 1 caracterizou o perfil das turmas e orientou a formação de times equilibrados; a Fase 2 introduziu os componentes do PC e as competências de SS por meio de atividades desplugadas; a Fase 3 conduziu os times à programação, do ambiente visual em blocos à linguagem textual; e a Fase 4 consolidou as aprendizagens por meio da análise das produções e do diálogo de *feedback*.

A aplicação mostrou que a organização do ensino em fases, conduzida pelos professores como mediadores, permitiu trabalhar os componentes do PC e as competências de SS de forma integrada e progressiva, com apoio dos instrumentos de acompanhamento previstos no *framework*. A abstração manteve-se como o componente mais difícil de mobilizar, sobretudo na transição para a programação textual, enquanto a colaboração e a comunicação se fortaleceram ao longo das atividades. O mecanismo de retroalimentação permitiu retomar, na fase correspondente, as lacunas identificadas, e a própria aplicação indicou ajustes ao *framework*, como a ampliação do tempo das atividades, a simplificação dos instrumentos e a automatização da formação dos times.

Com isso, o capítulo atende ao quarto objetivo específico da tese e contribui para responder à questão de pesquisa, ao mostrar que PC e SS, quando organizados em uma sequência de fases mediadas pelo professor, podem apoiar o ensino de programação na EPTNM de maneira sistematizada, e não apenas incidental. A análise das percepções de professores e estudantes sobre as contribuições do *framework* é apresentada no Capítulo 7.

7 PERCEPÇÕES DE PROFESSORES E ESTUDANTES SOBRE O SOPENSA

Este capítulo analisa a aplicação do *framework* SoPensa no ensino de programação, a partir das percepções dos professores que conduziram a intervenção e dos estudantes que participaram das atividades. Por se tratar de uma proposta voltada à mediação docente, a análise toma as percepções dos professores como foco principal e as dos estudantes como evidência complementar, reunidas por triangulação.

A análise foi conduzida com base nos dados coletados após a intervenção realizada nas aulas de Lógica de Programação. Para os professores, foram realizadas entrevistas individuais, com o objetivo de compreender suas percepções sobre os estudantes e sobre a aplicação do SoPensa em sala de aula. Para os estudantes, foi aplicado um questionário destinado a analisar a percepção sobre a aplicação do *framework* SoPensa.

A pesquisa adotou uma abordagem qualitativa, com uso complementar de dados quantitativos descritivos (Creswell; Creswell, 2017). A escolha dos instrumentos considerou as características de cada público investigado. Com os professores ($n = 2$), adotaram-se entrevistas semiestruturadas, considerando a profundidade analítica exigida pelo tratamento qualitativo de uma amostra reduzida (Mattar; Ramos, 2021). As entrevistas foram analisadas por meio da ATR (Braun; Clarke, 2022), procedimento que permitiu o aprofundamento da experiência de aplicação do SoPensa. Com os estudantes ($n = 64$), aplicou-se um questionário em escala *Likert*, adequado para caracterizar a distribuição das percepções em uma amostra de maior dimensão. Essa combinação de instrumentos configura a triangulação metodológica entre fontes complementares (Flick, 2013).

Os resultados apresentados expressam as percepções dos participantes e são interpretados como indicativos da experiência de aplicação do *framework* no contexto investigado, sem a pretensão de estabelecer relações causais diretas.

7.1 Percepções dos professores: análise temática reflexiva

Esta seção apresenta a análise das entrevistas realizadas com os dois professores que aplicaram o *framework* SoPensa em suas turmas de Lógica de Programação. A análise interpreta percepções construídas a partir dos relatos dos professores, considerando a natureza interpretativa do material qualitativo obtido nas entrevistas.

A análise foi orientada pela pergunta norteadora do roteiro de entrevista: *Como os professores percebem a aplicação do SoPensa em suas práticas de ensino de programação?* O roteiro completo encontra-se no Apêndice H. A discussão dos resultados em diálogo com a literatura sobre PC e SS aplicados ao ensino de programação é desenvolvida no capítulo seguinte.

A subseção 7.1.1 apresenta os procedimentos de análise adotados, incluindo o percurso de codificação, as decisões interpretativas e os critérios de qualidade observados. As subseções seguintes trazem os cinco temas construídos: O SoPensa reorganiza a prática docente; O SoPensa aproxima o aluno da disciplina; o SoPensa e a estruturação do raciocínio antes da codificação; o SoPensa e a percepção divergente sobre as *soft skills*; e o SoPensa requer condições e ajustes pedagógicos. Em seguida, é apresentado um eixo analítico complementar, denominado Potencial pedagógico, que expressa a convergência das avaliações dos professores quanto à recomendação do *framework*. A organização completa dos temas e dos códigos correspondentes encontra-se na Figura 40.

7.1.1 Procedimentos de análise

As entrevistas foram analisadas segundo a ATR de Braun e Clarke (2022). Adotou-se a versão reflexiva da abordagem, em vez das variantes orientadas à confiabilidade da codificação ou ao uso de livros de códigos previamente definidos. A escolha decorre do alinhamento entre os pressupostos da ATR e os objetivos da pesquisa. Nessa perspectiva, a subjetividade do pesquisador é compreendida como recurso analítico integrado ao processo, e não como viés a ser eliminado. A análise se apoia na coerência entre dados, códigos, temas e decisões assumidas ao longo do percurso (Braun; Clarke, 2022).

A ATR foi adotada por permitir compreender como os professores percebem a aplicação do *framework* SoPensa e os efeitos dessa aplicação no ensino de programação. Neste estudo, não se buscou estabelecer relações causais entre o uso da proposta e os efeitos relatados. As percepções construídas são compreendidas como situadas, ancoradas no contexto institucional e na trajetória profissional dos professores.

A familiarização com o *corpus* envolveu três escutas iniciais das gravações, intercaladas com a transcrição integral e com a revisão dos textos para retirada de marcadores conversacionais sem prejuízo do significado. Após a inserção das transcrições no *software* Atlas.ti (Friese, 2019), iniciou-se a codificação. A familiarização não se encerrou nesse momento, pois ao longo das fases subsequentes, sempre que dúvidas interpretativas surgiam, retornava-se às gravações e às transcrições, em escutas e releituras sucessivas. O retorno contínuo aos dados acompanhou a produção de "memos" analíticos e anotações reflexivas registradas durante todo o processo.

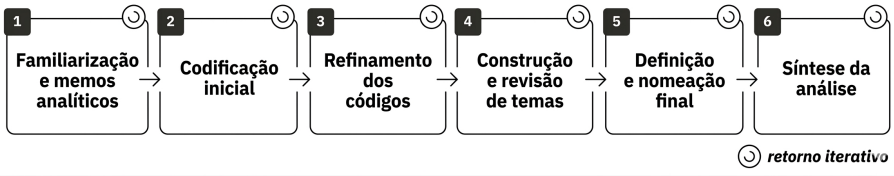
A codificação considerou trechos curtos ou segmentos mais amplos quando a preservação do sentido exigia essa manutenção contextual. Nos casos em que um mesmo recorte expressava mais de um significado analítico, foi utilizada a codificação múltipla. Essa decisão permitiu registrar diferentes leituras sobre uma mesma fala, sem fragmentação artificial do discurso.

A geração de códigos combinou orientação dedutiva e indutiva e operou em níveis semântico e latente. Em parte das falas, a codificação registrou o conteúdo manifesto dos enunciados (por exemplo: Decomposição, Tempo, Engajamento). Em outras passagens, identificou significados implícitos, como tensões, condições e racionalizações que os professores expressavam sem nomear diretamente (por exemplo, Limitação avaliativa, Sistematização pedagógica, Defasagem formativa). Parte dos códigos iniciais derivou dos eixos teóricos da pesquisa, PC e SS, em alinhamento com o desenho do estudo. Outros foram construídos a partir das próprias falas, capturando sentidos não previstos no roteiro, como Pertencimento e Lacuna curricular.

A análise foi organizada em seis fases sucessivas, percorridas com idas e vindas entre etapas adjacentes sempre que o trabalho posterior exigia revisitar decisões. A familiarização com o *corpus*, acompanhada de "memos" analíticos, abriu o processo. Seguiram-se a codificação inicial e o refinamento dos códigos por sucessivas reformulações, fusões, divisões e renomeações. A construção e a revisão dos temas e subtemas constituíram a etapa interpretativa central, que culminou na definição e na nomeação final dos temas e na síntese da análise, com a produção do quadro síntese, do mapa temático e da narrativa interpretativa. Os agrupamentos inicialmente definidos com base no roteiro de entrevista foram revistos e reorganizados em cinco temas interpretativos, cada qual estruturado por um conceito organizador central (*central organising concept*), além de um eixo analítico complementar (Braun; Clarke, 2022). A adaptação das fases buscou manter os princípios da abordagem reflexiva e evitar tratá-las como protocolo fixo.

A codificação foi percorrida em quatro rodadas sucessivas, partindo-se de 50 códigos na primeira rodada, 36 na segunda, 28 na terceira e 27 na quarta e final. Os números são apresentados como registro do refinamento analítico, sem operar como prova de rigor. Cada rodada envolveu fusões, divisões e renomeações de códigos à medida que a compreensão do *corpus* se aprofundava. A consolidação ocorreu na quarta rodada, quando novas leituras não indicaram necessidade de reorganizações analiticamente relevantes. A Figura 39 apresenta a organização do percurso, com os retornos iterativos entre fases sinalizados em cada etapa.

Figura 39: Percurso da análise temática reflexiva adotada no estudo



Fonte: Autoria própria.

Na construção dos temas, uma decisão metodológica merece destaque. O Tema 4, sobre a percepção divergente em relação às *soft skills*, foi construído tendo a divergência entre os professores como conceito central organizador. P2 relatou impacto positivo do *framework* no desenvolvimento de *soft skills*, enquanto P1 declarou não ter percebido evolução, atribuindo a percepção limitada à baixa frequência de contato com a turma. Optou-se por tratar a divergência como padrão analítico, em vez de considerá-la como inconsistência.

Na mesma linha, o eixo analítico complementar, Potencial pedagógico, não foi tratado como tema autônomo nem como tema-guarda-chuva. Trata-se de uma camada avaliativa que não organiza padrões de significado como os temas interpretativos. Registra a avaliação convergente dos participantes quanto à recomendação do *framework*, sustentada pelo código *Potencial pedagógico*, identificado em ambas as entrevistas. Sua classificação como eixo analítico complementar acompanha a orientação de Braun e Clarke (2022), que desaconselham tratar como tema abrangente (*overarching theme*) uma camada que não desempenha essa função analítica.

A organização dos dados e o registro das decisões foram apoiados pelo *software* Atlas.ti (Friese, 2019), com produção de quadros analíticos (Huberman *et al.*, 2014). Ao longo do trabalho, foram produzidos "memos" analíticos distribuídos entre as fases de familiarização, codificação, refinamento e tematização. As percepções construídas a partir das entrevistas foram postas em diálogo com os dados quantitativos descritivos do questionário

aplicado aos estudantes. O diálogo entre fontes foi utilizado para ampliar a interpretação do fenômeno, sem função de validação cruzada (Mattar; Ramos, 2021; Flick, 2013).

A análise foi conduzida pelo pesquisador, autor da tese, com a participação da orientadora como interlocutora crítica ao longo de quatro reuniões. Nesses encontros, a orientadora problematizou as decisões interpretativas e a delimitação entre códigos e temas, com a função de ampliar a reflexividade do pesquisador. A codificação e a definição dos temas permaneceram sob responsabilidade do pesquisador.

A posição do pesquisador como autor do *framework* SoPensa foi compreendida como parte constitutiva da própria análise e exigiu atenção ao risco de leituras excessivamente favoráveis. Por essa razão, foram preservados trechos que expressavam limites e dificuldades relatados pelos participantes. Em coerência com a perspectiva reflexiva adotada, as passagens em que o pesquisador assume posição interpretativa são redigidas em primeira pessoa, de modo a tornar visível a autoria das decisões analíticas. A análise observou os critérios de qualidade da ATR, com destaque para três aspectos centrais: a atenção repetida ao *corpus*; o papel ativo do pesquisador na construção interpretativa; a coerência interna dos temas. A condução por um único pesquisador é prática considerada adequada na ATR, em que a coerência interpretativa do *corpus* opera como referência de qualidade, sem exigência de concordância entre codificadores.

O *corpus* é composto pelas transcrições de duas entrevistas semiestruturadas realizadas com os dois professores responsáveis pela aplicação do *framework* SoPensa, caracterizados na Seção 6.2. Nenhum deles participou do estudo empírico apresentado no Capítulo 4, de modo que os códigos P1 e P2 utilizados neste capítulo não guardam correspondência com os códigos P1 a P13 atribuídos aos professores entrevistados naquela etapa. As entrevistas resultaram em 72 trechos literais, sendo 24 de P1 e 48 de P2. Em razão da codificação múltipla, esses 72 trechos receberam 85 aplicações de código, distribuídas em 26 em P1 e 59 em P2. A duração foi de 17 minutos e 28 segundos para P1 e 24 minutos para P2.

Os extratos apresentados nas subseções seguintes foram retirados literalmente das transcrições, sem paráfrase nas falas dos professores. Cada citação é identificada por P1 ou P2. Supressões de trechos irrelevantes ao argumento aparecem com [...]; esclarecimentos pontuais foram inseridos entre colchetes quando a fala isolada apresentava referência ambígua.

A distribuição desigual entre P1 e P2 decorre da extensão e da densidade das respostas produzidas em cada entrevista. A interpretação do *corpus* foi orientada pela relevância

analítica dos sentidos construídos e não pela frequência de ocorrência nas falas. Quando um código se sustenta em apenas um dos participantes, a origem é explicitada na narrativa interpretativa.

A Figura 40 apresenta a organização final dos temas e códigos da análise. Parte dos códigos finais do quadro temático foi desenvolvida na narrativa interpretativa; outros permanecem no quadro como registro do percurso, em coerência com a codificação múltipla e com o critério de relevância analítica. As subseções a seguir desenvolvem cada tema em sua especificidade interpretativa, seguidas do eixo analítico complementar.

Figura 40: Temas e códigos da análise temática reflexiva

Tema	Códigos (com indicação dos participantes)
T1 **O SoPensa reorganiza a prática docente**	Facilitação (P1, P2); Diagnóstico de perfil (P1, P2); • Aplicabilidade flexível (P1); • Sistematização pedagógica (P2); • Debate reflexivo (P2); • Progressão didática (P2); • Integração teoria-prática (P1)
T2 **O SoPensa aproxima o aluno da disciplina**	Engajamento (P1, P2); Melhoria no desempenho (P1, P2); • Iniciativa (P1); • Pertencimento (P1); • Preparação profissional (P2)
T3 **O SoPensa e a estruturação do raciocínio antes da codificação**	Decomposição (P1, P2); Abstração (P1, P2); Algoritmos (P1, P2); Resolução de problemas (P1, P2); • Reconhecimento de padrões (P2)
T4 **O SoPensa e a percepção divergente sobre as soft skills**	Colaboração (P1, P2); • Pensamento crítico (P2); • Limitação avaliativa (P1)
T5 **O SoPensa requer condições e ajustes pedagógicos**	**Condições modificáveis** Tempo (P1, P2); • Nivelamento inicial (P2); • Clareza metodológica (P2); • Recursos didáticos (P2) **Limites estruturais** • Defasagem formativa (P2); • Lacuna curricular (P2)
Eixo analítico complementar	*Potencial pedagógico (P1, P2)*

• *Marcador indica código individual (sustentado por apenas um dos professores).*

Fonte: Autoria própria.

7.1.2 T1. O SoPensa reorganiza a prática docente

O conceito central organizador deste tema é a reorganização da prática docente. P1 e P2 não tratam o SoPensa como acréscimo metodológico isolado. O *framework* aparece nas falas como elemento que modifica a forma de planejar e conduzir as aulas de Lógica de Programação, antes mesmo de produzir efeitos visíveis sobre o aluno.

A facilitação do processo de ensino-aprendizagem é afirmada de modo direto por P1: "O *framework* facilitou, de fato, o processo de ensino-aprendizagem." (P1)

P1 não atribui o ganho a uma técnica específica da abordagem, mas ao *framework* como um todo. P2 descreve o mesmo efeito por outra via, ao apresentar o SoPensa como sistematização institucional de práticas que, na rotina da educação técnica brasileira, costumam ocorrer de modo difuso:

> "SoPensa organiza de maneira pedagógica, institucional, essa construção, esse incentivo de você trabalhar, você construir, você debater *soft skills* e pensamento computacional. Acho que o principal ganho é esse, o principal ponto positivo." (P2)

A escolha do verbo organiza, em P2, sinaliza que o *framework* opera como estrutura e não como adição. A leitura ganha sustentação quando P2 retoma a mesma posição com outro verbo, em momento posterior da entrevista:

> "[o *framework*] formaliza algo que a gente está caminhando, ainda, a gente ainda está tropeçando na construção enquanto instituição, enquanto República." (P2)

O par 'organiza-formaliza' indica que o ponto não é pontual nem casual. Onde antes havia esforço difuso do docente para tratar PC e SS, passa a existir um arranjo institucional reconhecível. A sistematização aparece também quando P2 descreve a sequência didática proposta pelo *framework*:

> "eu acho importante seguir essa linha, começar com as desplugadas, principalmente quando você faz esse processo contextualizado com a realidade do aluno. [...] Depois você vai para atividades plugadas e gamificação, porque ele passa por um processo de amadurecimento nessa profissão." (P2)

Três elementos se conectam na passagem: a sequência didática, a contextualização com a realidade do aluno e a noção de amadurecimento profissional. A formulação descreve uma progressão que não está prevista na ementa, mas que o *framework* permite estruturar.

O alinhamento entre P1 e P2 sustenta uma leitura não explicitada nas falas, mas construída a partir delas. P2 usa a palavra organiza, P1 usa a palavra facilitou. Ambos descrevem, com vocabulários distintos, a passagem de algo difuso para algo estruturado. Tomo essa convergência como leitura implícita sobre a prática docente em Lógica de Programação que precede o *framework*. Sem uma estrutura como o SoPensa, o que é cobrado do professor é o esforço difuso, individual, não institucionalmente reconhecido.

A leitura ancora-se em escolhas terminológicas. Quando P2 afirma que o *framework* organiza institucionalmente, pressupõe-se que algo institucionalmente desorganizado existia antes. Quando P1 afirma que o *framework* facilitou, pressupõe-se que algo difícil sem ele existia antes. A função do SoPensa, nesta interpretação, é menos técnica do que institucional, ao tornar institucionalizável aquilo que antes dependia da iniciativa individual do professor.

O segundo movimento da reorganização aparece em P1 como diagnóstico do perfil dos alunos:

> "a partir do *framework* nós conseguimos ter uma análise individual do perfil de cada aluno para poder personalizar as atividades e a evolução da disciplina." (P1)

O SoPensa permite ao professor partir do aluno real, e não do aluno presumido pela ementa. P2 descreve movimento semelhante ao apontar que o *framework* também ajuda a reconhecer dificuldades estruturais dentro da própria sala de aula:

> "O SoPensa também possibilitou identificar esses problemas que existem na sala de aula e amadurecer que a gente entende com maturidade que a gente tem que lidar com isso e que a gente tem que encontrar uma forma de trabalhar para mitigar isso." (P2)

P1 fala em personalização das atividades. P2 fala em mitigação de dificuldades. Em ambos os casos, o *framework* funciona como mecanismo de leitura prévia da turma e essa leitura abre espaço para decisões pedagógicas que antes não tinham apoio metodológico.

A leitura adotada considera que P1 e P2 descrevem uma operação que se faz contra um padrão prévio da disciplina por eles ministrada, no qual a aula se organiza, em torno da ementa, com a sintaxe da linguagem ou o conteúdo curricular como ponto de partida.

Duas formulações sustentam essa interpretação. P1 usa o verbo personalizar, que pressupõe uma forma padrão de evolução da disciplina, contra a qual a personalização se faz. P2 usa a expressão problemas que existem na sala de aula, que pressupõe que esses problemas estavam invisíveis antes. Os dois pressupostos autorizam a leitura de que o *framework* opera contra uma prática anterior, na qual o aluno entrava como destinatário de algo já decidido.

A reorganização descrita pelos professores não foi automática nem ocorreu sem atritos. P2 relata que a aplicação exigiu pausas explícitas para que o *framework* pudesse ser entendido em sala:

> "como o professor definiu a equipe, os alunos a gente tinha que, principalmente o aplicador, a gente tinha que parar pra explicar alguns eventos do SoPensa, pra explicar a metodologia, como é que deveria ocorrer, e isso foi melhorando com o tempo, mas no início os alunos tinham essa dificuldade." (P2)

O trecho contrapõe a leitura otimista que o tema poderia projetar. A reorganização da prática docente, no caso aplicado, dependeu de mediação adicional do aplicador para

que os alunos compreendessem o próprio *framework*. O efeito reorganizador não foi instantâneo, mas negociado durante a aplicação, com percalços operacionais que P2 descreve em tom de aprendizado, e não de falha. O SoPensa modifica a prática docente, ainda que essa modificação envolva trabalho adicional de explicação, especialmente nos primeiros momentos.

A reorganização também se manifesta em planos cotidianos, como a flexibilidade de aplicação do *framework* em laboratório ou em sala teórica e o uso recorrente do debate reflexivo nas dinâmicas de aula relatadas por P2. Os elementos do tema, vistos em conjunto, indicam que o SoPensa desloca o ponto de partida da aula da ementa para o aluno e oferece estrutura para gestos pedagógicos que antes dependiam da sensibilidade do professor. As perspectivas dos dois professores não são equivalentes nem opostos. P1 observa a alteração em escala individual e cotidiana. P2 a observa em escala institucional e formativa.

7.1.3 T2. O SoPensa aproxima o aluno da disciplina

Se o T1 trata do efeito do *framework* sobre a prática docente, o T2 direciona o foco para a recepção dos alunos. O eixo principal deste tema é a aproximação à disciplina. Não se trata de engajamento pontual, mas de uma mudança na relação dos alunos com Lógica de Programação. A aproximação se manifesta em dois registros principais: o engajamento durante as atividades e a melhoria de desempenho percebida em comparação com outra turma da mesma disciplina ministrada por cada professor, turma que não participou da pesquisa.

O engajamento aparece com mais densidade nas falas de P1, que apresenta um indicador inesperado, a apropriação informal do vocabulário técnico pelos alunos:

> Eu acredito que até alguns jargões da lógica eles utilizam como piada, como trocadilhos na sala de aula e eu acredito que isso representa um interesse acima da média dos alunos de programação. (P1)

Quando o jargão sai do quadro e entra na conversa entre colegas, o conteúdo deixou de ser estranho. Em outra passagem, P1 explicita o vínculo entre *framework*, motivação e a posição do aluno frente ao conteúdo:

> Eles conseguem experimentar os conceitos teóricos no seu dia a dia, assim como a gamificação tornou o processo de ensino-aprendizagem muito mais interessante ao aluno, trazendo para ele o sentido de desafios e prêmios para a superação das atividades, dos trabalhos. (P1)

A gamificação não é descrita como estratégia motivacional, mas como recurso que aproxima o conteúdo do cotidiano. Desafio e prêmio aparecem como mecanismos de superação, não de competição. Em P2, o engajamento assume outra dimensão e mobiliza também o docente:

> a gente aproveitou, [...] eu aproveitei essa oportunidade do SoPensa para também intensificar o debate sobre pensamento computacional dentro da disciplina. Então, foi um elemento motivador [...] tanto para o aluno quanto para o professor. (P2)

Adoto a leitura de que o engajamento, neste caso, deixa de ser propriedade do aluno e passa a caracterizar a relação pedagógica.

O jargão técnico transformado em piada, descrito por P1, e o engajamento que motiva também o professor, descrito por P2, têm em comum um elemento que nenhum dos dois nomeia diretamente: a dissolução parcial dos limites entre quem ensina e quem aprende. A leitura ancora-se em duas escolhas dos professores. P1 usa o termo trocadilhos para descrever o que os alunos fazem com o vocabulário técnico, gesto que pressupõe domínio mínimo da linguagem técnica do professor. P2 usa a expressão tanto para o aluno quanto para o professor para descrever o efeito motivador, gesto que coloca os dois na mesma posição diante do *framework*.

Defendo a leitura de que o SoPensa, embora apresentado em registros distintos pelos dois professores, abre algum grau de mobilidade entre as posições de professor e aluno nesse contexto. Não estou afirmando que as falas dizem isso de modo explícito. Trata-se de uma leitura sustentada na proximidade entre os dois relatos e nas pressuposições de cada termo escolhido pelos participantes.

O segundo registro, melhoria de desempenho, ganha sustentação porque ambos os professores ministram a mesma disciplina em duas turmas e podem comparar a recepção do *framework*. P1 aplicou o SoPensa em uma das duas turmas, e usa a outra explicitamente como parâmetro de comparação:

> Eles conseguiram evoluir. Eu utilizo como parâmetro de comparação a outra turma, onde eu ministro a mesma disciplina com o mesmo conteúdo. Essa turma apresenta um diferencial positivo que eu acredito que é graças ao *framework*. (P1)

P2 atuou em condição equivalente e relatou percepção semelhante:

> Senti um impacto positivo com relação à turma que utilizou, principalmente em relação a *soft skills*, principalmente no processo de colaboração,

> no processo de construção crítica, no debate, no processo de pensamento computacional. (P2)

Os dois professores atribuem o diferencial ao SoPensa. A convergência importa porque os critérios de comparação são internos à prática de cada um, com o mesmo conteúdo, a mesma disciplina, o mesmo professor e turmas distintas.

Por outro lado, não se pode olhar para essa aproximação apenas de forma otimista. P2 descreve, em outro momento, uma condição inicial que contrasta com a recepção positiva relatada:

> nosso aluno ele chega com dificuldades de interação e trabalhos colaborativos. Isso é um problema muito grande, porque a partir do momento que você tem que analisar um processo de decomposição, de abstração e de construção de elementos sintáticos e semânticos, de alunos que possuem realidades diferentes e que não estão acostumados a interagir, isso dificulta bastante. (P2)

A aproximação que o tema descreve não foi imediata nem ocorreu em terreno favorável. A turma chegou com dificuldades prévias de interação, condição que P2 atribui em parte ao efeito da pandemia sobre essa geração. O contraponto desfaz a impressão de que o *framework* operou sobre alunos previamente preparados para receber sua proposta. A aproximação se constituiu apesar das dificuldades de partida, e não a partir de um estado inicial de receptividade.

O dado é importante para qualificar o que o tema descreve. O SoPensa não encontrou alunos engajados. Engajou-os progressivamente, em condições que não eram, à entrada, propícias.

Um aspecto adicional projeta o efeito do *framework* para além do presente da sala de aula. P2 aponta que o trabalho colaborativo iniciado com o SoPensa prepara o aluno para a realidade futura do trabalho. P1 lê o ganho no presente da disciplina; P2 lê o ganho em horizonte profissional. A diferença não constitui contradição. A aproximação produzida pelo *framework* opera em mais de um plano simultaneamente, e cada professor capta o plano que sua perspectiva permite ver com mais nitidez.

7.1.4 T3. O SoPensa e a estruturação do raciocínio antes da codificação

Neste tema, o conceito central organizador é a estruturação do raciocínio antes da codificação. Para os dois professores, a decomposição, a abstração e os algoritmos operam como organização do pensamento que antecede e fundamenta a escrita do código. P1 e

P2 não tratam codificar como o ato cognitivo principal. Codificar aparece como momento em que se efetiva um raciocínio previamente estruturado.

Em uma única passagem, P2 reúne três dos componentes do PC e identifica o ponto que considera mais difícil:

> E ficou claro que os alunos amadureceram no processo de decomposição, reconhecimento de padrões, no processo de abstração, no processo de transformação do processo abstrato em codificação e, posteriormente, a maturidade a nível semântico e sintático de elaboração de algoritmos. (P2)

Três elementos merecem atenção na fala. Primeiro, P2 descreve um percurso, não uma lista. Parte-se da decomposição, atravessa-se a abstração, chega-se ao algoritmo. Segundo, o algoritmo é apresentado como ponto de chegada de um processo cognitivo mais amplo, e não como início. Terceiro, a maturidade sintática vem depois, e não antes, da elaboração mental do problema.

A leitura sequencial é uma construção interpretativa minha, ancorada na estrutura sintática do extrato. A sequência "no processo de... no processo de... e, posteriormente" autoriza ler o conjunto como percurso, ainda que P2 não use o termo.

O percurso é confirmado por P1 quando o professor descreve a decomposição em ação durante uma atividade específica:

> Os alunos aplicaram nas minhas aulas alguns conceitos oriundos do *framework* SoPensa, como, por exemplo, na construção de laços de repetição e de desvio condicional, na qual o aluno precisava decompor o problema em trechos e a partir desses trechos resolver determinadas situações para uma condição específica. (P1)

A decomposição aparece ancorada em operação cognitiva específica, dividir o problema em trechos antes de codificar. Não se trata de noção abstrata, mas de gesto observado pelo professor. P2 retoma a centralidade da decomposição em outro momento, agora em chave reflexiva:

> Foram mudanças positivas no processo de raciocínio abstrato, da decomposição de problemas, da compreensão desse processo de decomposição, que eu acho que é a parte mais difícil, e posteriormente desse amadurecimento de como você partiu do abstrato para o concreto tendo um conjunto de códigos que são específicos. (P2)

A repetição da palavra decomposição em duas frases consecutivas marca onde, na percepção de P2, está o ponto crítico do percurso. Quando o mesmo professor sugere

ajustes para futuras aplicações, é à decomposição que ele retorna: "[...]a sugestão é realmente você aumentar essa carga de trabalho e intensificar principalmente no processo de decomposição" (P2).

A abstração ocupa lugar diferente na narrativa dos dois professores e é, ao mesmo tempo, o componente onde a tensão interna do tema fica mais visível. P1 é enfático: "A maior dificuldade deles hoje é a abstração" (P1).

P2 formula avaliação equivalente: "a principal dificuldade é o processo da abstração, que é onde ocorre o entrave crucial" (P2).

A leitura dessas falas exige uma ressalva. Em outro momento da entrevista, ao descrever o percurso cognitivo, P2 atribui à compreensão da decomposição a condição de "parte mais difícil". As duas formulações não são contraditórias quando lidas no percurso que o próprio professor descreve: a decomposição concentra a dificuldade de partida, no momento em que o problema precisa ser dividido, enquanto a abstração concentra o entrave de passagem, quando o raciocínio precisa converter-se em código. Adoto essa leitura porque P2 associa a decomposição ao início do percurso e a abstração ao ponto em que ele se interrompe. Registro, contudo, que se trata de uma reconciliação interpretativa e que a alternativa, ler as duas falas como oscilação do participante quanto ao ponto crítico, também é sustentável no *corpus*.

Feita a ressalva, os dois professores convergem sobre a dificuldade da abstração. P1 e P2, no momento da entrevista, afirmam que a abstração é a maior dificuldade dos alunos, e nenhum dos dois relata que o *framework* tenha resolvido essa situação. Essa permanência precisa ficar visível na narrativa, e não ser absorvida pela leitura otimista do percurso cognitivo.

O ponto ponderado vem em seguida, quando P2 também aponta ganho nesse aspecto: "houve essa melhoria, sim, porque primeiro despertou a necessidade de se pensar de maneira abstrata, entender como é que o conhecimento é construído" (P2).

O ganho descrito por P2 é qualitativamente diferente do ganho que se esperaria. Os alunos não passaram a abstrair com facilidade, passaram a perceber que precisavam abstrair. O *framework* abriu espaço para a deliberação sobre a abstração, sem garantir o domínio dela.

Algoritmos aparecem nas falas de P2 como ponto final do percurso:

> entender contexto para posteriormente, através de símbolos próprios da lógica de programação, você conseguir estruturar o pensamento que foi

> transformado, passado do abstrato para algo codificado. (P2)

Nessa fala, P2 não enfatiza a sintaxe. O código é tratado como ponto em que se efetiva o pensamento já decomposto e abstraído. A mesma posição reaparece quando P2 caracteriza o passo fundamental do processo, o registro dos elementos que compõem cada etapa até a efetivação do código.

A resolução de problemas completa o conjunto dos elementos mobilizados pela análise. P1 descreve o movimento como deslocamento do aluno, que passa de quem resolve para quem identifica alternativas. O gesto é coerente com a leitura central do tema, em que o raciocínio antecede a escrita do código.

O tema desenha uma operação cognitiva em condições mistas. Avança em decomposição e em algoritmos, mas mantém a abstração como ponto sensível persistente. P1 oferece o exemplo concreto e a percepção da dificuldade. P2 oferece a sequência dos componentes e identifica o ponto de maior tensão. Os dois professores constroem, com vocabulários distintos, uma mesma narrativa sobre o que significa raciocinar antes do código, narrativa que reconhece avanços parciais.

7.1.5 T4. O SoPensa e a percepção divergente sobre as *soft skills*

A divergência entre P1 e P2 sobre as *soft skills* é tratada nesta análise como conteúdo do tema, e não como inconsistência ou ruído. O conceito central organizador é a própria tensão entre as duas percepções, possibilidade reconhecida por Braun e Clarke (2022) quando o tema se organiza em torno de uma contradição.

Antes de avançar, faço uma pausa reflexiva necessária para o tema. A interpretação que ofereço aqui depende fortemente de uma operação que eu mesmo realizo, ler na fala de P1 não uma negação, mas uma cautela ligada a limite de observação. Outras leituras seriam possíveis. Alguém poderia interpretar a recusa de P1 como ceticismo genuíno em relação à percepção do desenvolvimento de SS no contexto da disciplina, sem que a frequência de contato fosse a explicação principal. Mantenho essa leitura porque a fala explicita o limite de observação como justificativa, mas reconheço que estou tomando posição interpretativa entre alternativas legítimas.

P1, ao ser perguntado sobre evolução nas SS, nega a percepção de mudança:

> Não, com a Soft Skills eu não percebi nenhuma evolução ou diferença, até porque eu vejo eles com pouca frequência durante a semana. Então, não houve nenhuma percepção em relação a 'sim ou não', se eles conseguiram

> trabalhar o pensamento crítico, se eles conseguiram colaborar na hora de desenvolver o código ou desenvolver alguma situação problema. (P1)

A leitura cuidadosa da fala revela algo que a recusa imediata pode ocultar. P1 não nega que as *soft skills* tenham se desenvolvido nos alunos. Afirma que não teve condições de observar. A justificativa é metodológica, a frequência semanal de contato com a turma foi reduzida. O código atribuído a esse trecho, Limitação avaliativa, captura precisamente o ponto, e essa distinção é fundamental para o tema.

Em uma única passagem, P1 chega perto de admitir comportamento colaborativo entre os alunos:

> Eles demonstraram várias situações na qual um consultava o outro durante a realização de atividades em sala de aula ou no laboratório de informática, e isso pode estar vinculado ao uso do *framework*. (P1)

A construção "pode estar vinculado" sintetiza a posição. P1 reconhece o comportamento, mas não tem condições de afirmar que o *framework* foi a causa. A cautela é coerente com a recusa anterior. Não há contradição interna na fala de P1, há limite de observação.

P2, em contraste, oferece percepção oposta e detalhada. Observou colaboração efetiva e atribui valor central a ela:

> como houve trabalho conjunto, principalmente motivado pelo SoPensa, houve o melhor desenvolvimento dessa construção que vai desde o processo de decomposição até a determinação dos algoritmos. (P2)

A colaboração não aparece como atributo isolado. P2 a descreve como condição que potencializa o PC. Quando os alunos trabalham em grupo, decomposição e abstração funcionam melhor, e a relação entre *soft skills* e PC é o que, para P2, justifica o diagnóstico positivo.

O contexto da colaboração observada por P2 é particularmente revelador. A turma chegou com dificuldade prévia de interação:

> nosso aluno ele chega com dificuldades de interação e trabalhos colaborativos. Isso é um problema muito grande, porque a partir do momento que você tem que analisar um processo de decomposição, de abstração e de construção de elementos sintáticos e semânticos, de alunos que possuem realidades diferentes e que não estão acostumados a interagir, isso dificulta bastante. (P2)

Para P2, a colaboração precisou ser construída durante a aplicação. O professor descreve uma progressão observada ao longo do tempo, atribuindo parte da dificuldade inicial aos efeitos da pandemia sobre essa geração e relatando que, com o tempo, a interação melhorou.

O tema, lido como divergência entre duas perspectivas, deixa ver algo que nenhuma das duas falas isoladamente revelaria. A percepção sobre SS é função das condições em que ela pode se constituir. P2 acompanhou a turma em frequência maior, e por isso pôde observar a progressão. P1, com frequência semanal reduzida, não pôde.

O contraste registrado pelo tema, portanto, não diz respeito ao que aconteceu com os alunos, mas ao que foi possível ver de cada posição docente. O desdobramento sustenta uma leitura específica. A percepção sobre o efeito do *framework* em SS depende das condições concretas de observação que cada professor teve em sua turma.

7.1.6 T5. O SoPensa requer condições e ajustes pedagógicos

O tema reúne o reconhecimento, pelos professores, dos fatores que precedem e acompanham a aplicação do SoPensa, e dos ajustes que cada um propõe diante desses fatores. O conceito central organizador é a relação entre condicionantes e respostas. Em geral, as condições mencionadas pelos professores aparecem relacionadas a ajustes propostos. Algumas, no entanto, são de natureza estrutural.

A condição mais evidente nas falas dos dois professores é o tempo. P1 considera o período aplicado suficiente, mas no limite mínimo:

> o tempo no qual foi aplicado é suficiente, foi o mínimo. Acredito que caso haja oportunidade de replicar em paralelo durante a carga horária da disciplina até a sua conclusão, do início até a sua conclusão, acredito que traga mais benefícios. (P1)

P2 chega ao mesmo diagnóstico por outro caminho e propõe uma organização específica:

> Essa metodologia seria importantíssima de se aplicar de maneira um pouco mais extensa, com prazo maior, e principalmente em um momento anterior ao início efetivo da disciplina de lógica de programação. (P2)

Os dois professores convergem quanto à projeção de ganhos em aplicações mais extensas, mas partem de avaliações distintas sobre o período aplicado. P1 o considera suficiente, ainda que situado no limite mínimo. P2 não emite juízo sobre a suficiência e formula sua

resposta como proposta de reorganização, ao indicar prazo maior e aplicação anterior ao início efetivo da disciplina. A convergência, portanto, está na direção do ajuste sugerido, e não na avaliação do tempo empregado. P2 acrescenta ainda uma especificação que reposiciona o *framework* no percurso formativo:

> seria importantíssimo uma metodologia como essa, ela ser aplicada, não sei se em caráter de nivelamento ou caráter preliminar, mas seria importante se você se aplicasse no momento anterior à efetiva abordagem da ementa curricular da disciplina. (P2)

Tratar o SoPensa como nivelamento prévio à ementa, na proposta de P2, não substitui a aplicação durante a disciplina. Reposiciona o *framework* no tempo formativo do aluno, em momento anterior ao trabalho com a sintaxe da linguagem. A proposta de ajuste responde diretamente a uma condição percebida, a defasagem com que os alunos chegam à disciplina, ponto que P2 retoma em vários momentos da entrevista.

Os recursos didáticos e a clareza metodológica completam o conjunto de ajustes propostos. P2 sugere o uso de IDEs em bloco, mais intuitivas, que não exijam grande conhecimento de sintaxe. Relata também que o aplicador precisou pausar, durante a aplicação, para explicar etapas do *framework* aos alunos. Os dois ajustes são respostas operacionais às dificuldades observadas, e fazem parte do mesmo movimento analítico das outras propostas, identificar o que precisa ser melhorado para que o *framework* opere com mais alcance.

A análise dos condicionantes estruturais é sustentada exclusivamente por P2 e diz respeito a condições que precedem o *framework* e o ultrapassam em escala. O primeiro condicionante estrutural é a defasagem formativa, ponto que P2 retoma em várias passagens da entrevista:

> Dentro de lógica e programação, eu entendo que o *framework* tem muito a colaborar, principalmente porque nós temos uma defasagem muito grande em relação ao ciclo natural, que é da construção, ou seja, do extrato, da interpretação do texto, fruto de uma defasagem no ensino de matemática em língua portuguesa. (P2)

> há dificuldades no processo de construção de raciocínio lógico advindo da má formação dos anos anteriores, no círculo fundamental, e há dificuldade de interpretação de texto. (P2)

Assumo uma posição interpretativa específica. Caracterizar a defasagem como condição estrutural, e não como dificuldade circunstancial, é uma leitura minha, ancorada

em duas marcas explícitas das falas de P2. A primeira é a referência repetida ao ensino fundamental e à formação anterior, que situa a origem do problema fora do controle do professor de Lógica de Programação. A segunda é a sequência de campos disciplinares afetados, matemática, língua portuguesa, raciocínio lógico, interpretação de texto, que indica condição que ultrapassa o contexto da turma específica. P2 não usa o termo estrutural, mas as duas marcas autorizam a interpretação.

O SoPensa não corrige a defasagem, e P2 é honesto quanto a isso. O que o *framework* permite, segundo o professor, é identificá-la e iniciar tratamento dela no contexto da disciplina:

> com base nesse *framework*, a gente consegue identificar inicialmente e começar a realizar um tratamento nessa defasagem de conteúdos e de outros elementos que fazem parte da formação do aluno. (P2)

Essa distinção é importante. O *framework* é instrumento de diagnóstico e mitigação, não de remediação completa. Pretender que ele resolva uma defasagem que vem do ensino fundamental seria sobrecarregar a metodologia com função que não lhe cabe.

O segundo condicionante estrutural opera em nível institucional, a lacuna curricular:

> a construção dos nossos planos de ensino, nossas ementas, elas não estão focadas no desenvolvimento do pensamento computacional e de *soft skills*. (P2)

P2 nomeia uma condição que está fora da sala de aula e fora do *framework*. A leitura de P2 sugere que, no contexto da disciplina, a aplicação do SoPensa ocorre em ambiente curricular que não prevê o desenvolvimento de PC e SS de forma estruturada.

Lido em sua totalidade, o tema organiza condicionantes em dois grupos. No grupo das condições modificáveis, fatores percebidos pelos professores (tempo, recursos didáticos, clareza metodológica) aparecem acompanhados de ajustes que cada um propõe. No grupo dos limites estruturais, P2 identifica condições que precedem o *framework*, como a defasagem formativa e a lacuna curricular, em relação às quais o *framework* opera mais como diagnóstico do que como solução. A leitura adotada é a de que a relação entre os dois grupos descreve o alcance e os limites do SoPensa no contexto da disciplina.

7.1.7 Eixo analítico complementar. Potencial pedagógico

O eixo analítico complementar não constitui tema autônomo. Trata-se de uma percepção convergente sobre o *framework*, sustentada pelo código Potencial pedagógico presente

nas falas de P1 e P2 em diferentes momentos das duas entrevistas. Optar por tratá-lo como eixo analítico complementar, e não como tema, segue Braun e Clarke (2022), que recomendam não construir temas adicionais sem necessidade analítica. O que aparece aqui não é um sexto conceito central organizador, mas uma camada avaliativa que atravessa o conjunto. P1 e P2 expressam, com vocabulários distintos, a mesma avaliação geral sobre o *framework*.

P1 oferece avaliação direta e amplia a recomendação para além da disciplina:

> Acredito que é um aspecto muito positivo e que pode e precisa ser replicado em outras disciplinas também. Principalmente nas disciplinas de exatas, principalmente, mas não apenas. Por exemplo, nas disciplinas biológicas e humanas também. (P1)

A formulação importa porque desvincula o potencial do *framework* de um conteúdo curricular único. Na percepção de P1, o que o SoPensa organiza não é específico da Lógica de Programação, mas uma forma de ensinar que pode operar em outros campos.

P2 chega à mesma conclusão geral por outro caminho e mobiliza explicitamente o termo potencial:

> A visão geral sobre o *framework* e a metodologia aplicada é que ele tem muito potencial para colaborar dentro do grupo acadêmico para a aula de programação, para as aulas de lógica. (P2)

Os dois professores recomendam o *framework*, mas as condições da recomendação diferem entre eles. P1 enfatiza a análise individual do perfil dos alunos no início da jornada acadêmica. P2 enfatiza a aplicação prévia à ementa de Lógica de Programação. A diferença no modo de recomendar corresponde às perspectivas identificadas nos temas anteriores. O que importa para o eixo é que, apesar das diferenças, ambos chegam à mesma conclusão sobre o conjunto da experiência.

A convergência não substitui os cinco temas. Complementa-os como camada avaliativa. A função analítica do eixo é permitir ler que, mesmo diante das divergências sobre SS (T4) e dos condicionantes estruturais identificados (T5), os dois professores avaliam positivamente o *framework*. A interpretação adotada é que P1 e P2 recomendam o *framework* e, ao mesmo tempo, reconhecem os condicionantes apresentados no T5. A recomendação não é descrita pelos professores como aprovação plena, mas como avaliação positiva do que foi observado na aplicação.

7.2 Percepções dos estudantes

Esta seção apresenta a percepção dos estudantes sobre o uso do SoPensa nas aulas de Lógica de Programação, com base nas respostas ao questionário aplicado.

Participaram desta etapa 64 estudantes, sendo 32 do curso de Informática para Internet e 32 de Informática em Redes, o que corresponde a 91,4% dos 70 alunos matriculados nas duas turmas. Para preservar o anonimato nas respostas abertas, os participantes foram identificados pela letra A, seguida de numeração sequencial de A1 a A64.

O questionário foi aplicado ao final da intervenção e reuniu questões objetivas organizadas em cinco aspectos: visão geral sobre o SoPensa, avaliação das atividades desenvolvidas, desenvolvimento percebido em PC, desenvolvimento percebido em SS e recomendação de uso do *framework*. Utilizou-se uma escala *Likert* de cinco pontos, composta por: 1 = Discordo totalmente, 2 = Discordo, 3 = Neutro, 4 = Concordo e 5 = Concordo totalmente. O instrumento completo utilizado na coleta de dados encontra-se no Apêndice G.

7.2.1 Procedimentos de análise

Os dados quantitativos foram tabulados e analisados por meio de estatística descritiva, com distribuição de frequências e percentuais para cada item da escala *Likert*. As respostas foram tratadas como dados ordinais, sem agregação em índices compostos ou cálculo de médias, de acordo com o caráter da escala (Boone Jr; Boone, 2012). Para a apresentação visual, adotaram-se gráficos de barras horizontais empilhadas a 100%, recurso que facilita a leitura comparativa entre os níveis de concordância, neutralidade e discordância, conforme os princípios de visualização de dados propostos por Knaflic (2019).

Para a construção do instrumento, adotaram-se as diretrizes propostas por Kitchenham e Pfleeger (2008), adaptadas ao contexto educacional. As afirmativas relacionadas ao PC e às SS foram organizadas a partir, respectivamente, dos seus quatro componentes e das suas quatro competências, cujos critérios foram definidos na proposta do *framework* (Capítulo 5). O instrumento incluiu ainda a resolução de problemas, tratada de forma transversal por expressar o objetivo geral para o qual o *framework* se orienta.

O instrumento incluiu um item formulado de maneira reversa, “Algumas atividades do SoPensa não acrescentaram muito ao meu aprendizado”. Esse item foi analisado preservando-se as categorias originais da escala, sendo a inversão tratada apenas no plano interpretativo, em que a discordância passa a representar avaliação favorável.

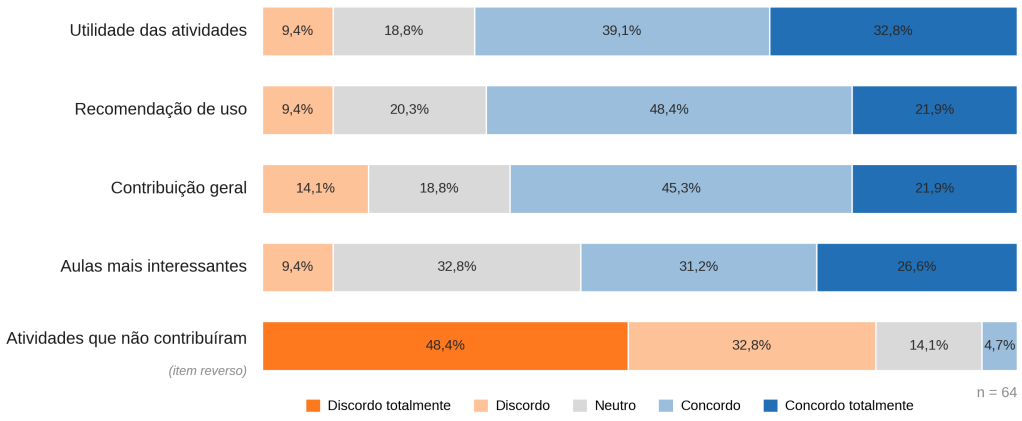
As respostas abertas do questionário foram analisadas qualitativamente e utilizadas de forma complementar aos dados quantitativos, configurando triangulação metodológica entre as duas fontes (Flick, 2013). O instrumento contemplou um único campo de comentário aberto, comum a todo o questionário, no qual os estudantes registravam observações gerais sobre a participação nas atividades.

Em cada dimensão analisada, foram selecionados trechos com conexão temática plausível ao item discutido, priorizando relatos que ajudassem a explicar percepções favoráveis, neutras ou mais moderadas em relação ao *framework*. As falas foram lidas e organizadas manualmente pelo pesquisador, sem uso de *software* de análise qualitativa, considerando o volume e o caráter pontual das respostas.

7.2.2 Aceitação do SoPensa

A aceitação do SoPensa foi examinada por meio de três conjuntos de itens: a contribuição geral do *framework*, a avaliação das atividades desenvolvidas e a recomendação de uso. A Figura 41 apresenta a distribuição das respostas.

Figura 41: Percepção dos estudantes sobre a aceitação do SoPensa



Fonte: Autoria própria.

Quanto à **contribuição geral** do SoPensa para as aulas de Lógica de Programação, dos 64 participantes, (67,2%) assinalaram concordância, sendo 45,3% em concordo e 21,9% em concordo totalmente. A neutralidade foi registrada por 18,8% e a discordância por 14,1%, sem ocorrências na categoria discordo totalmente.

Nas respostas abertas sobre a contribuição geral, A23 afirmou que "A dinâmica ajudou na compreensão dos assuntos, incentivando a aprendizagem". A17 registrou que "Me ajudou muito, aprendi de uma forma diferente". A8 apresentou uma avaliação mais moderada, ao afirmar que as atividades "ajudaram bastante", com a ressalva de que o registro

correspondia ao que ainda conseguia recordar. A22 justificou a opção neutra ao afirmar que "É neutro, não ajudou muito porém não é péssimo". Entre os que discordaram, A48 reconheceu que as explicações dos jogos e da lógica eram relativamente boas, mas relatou ainda ter encontrado dificuldade para compreender certos pontos.

As **atividades propostas** pelo SoPensa contemplaram atividades sem computador, uso do laboratório, jogos, gamificação e situações-problema. Em "As atividades foram úteis para o meu aprendizado", (71,9%) dos estudantes assinalaram concordância, sendo 39,1% em concordo e 32,8% em concordo totalmente. A neutralidade foi registrada por 18,8% e a discordância por 9,4%, sem ocorrências na categoria discordo totalmente.

No item "As atividades tornaram as aulas mais interessantes e motivadoras", a concordância somada foi de (57,8%), com 31,2% em concordo e 26,6% em concordo totalmente. A neutralidade alcançou 32,8%, e a discordância, 9,4%. No item formulado de maneira reversa, "Algumas atividades do SoPensa não acrescentaram muito ao meu aprendizado", 81,2% dos estudantes assinalaram discordo ou discordo totalmente, posicionamento que, após a inversão da escala, corresponde a avaliação favorável. Três estudantes (4,7%) concordaram com a afirmação, divergindo do padrão observado nos dois itens anteriores.

Nas respostas abertas sobre as atividades, A39 afirmou que "todas as atividades do SoPensa foram úteis e complementares, tornando o aprendizado mais claro, dinâmico e eficaz". A44 apresentou avaliação ponderada. Embora já tivesse conhecimento prévio em lógica, reconheceu que a estruturação em grupo tornou o entendimento do conteúdo mais prático e didático. Avaliações mais moderadas também foram registradas. A15 observou que as atividades foram boas e contribuíram para a integração entre os alunos, mas frisou que poderiam ter ocorrido em maior quantidade. A38 apontou o caráter prático como um aspecto positivo e observou que a explicação teórica poderia estar mais presente nos momentos de gamificação, para maior clareza na execução das tarefas. Entre os que discordaram, A45 relatou que "não entendi muito e não aprendi porque achei difícil".

Quanto à **recomendação de uso** do SoPensa em outras turmas e contextos de aprendizagem, a concordância somada foi de (70,3%), sendo 48,4% em concordo e 21,9% em concordo totalmente. A neutralidade foi registrada por 20,3% dos respondentes, e a discordância por 9,4%, sem ocorrências na categoria discordo totalmente.

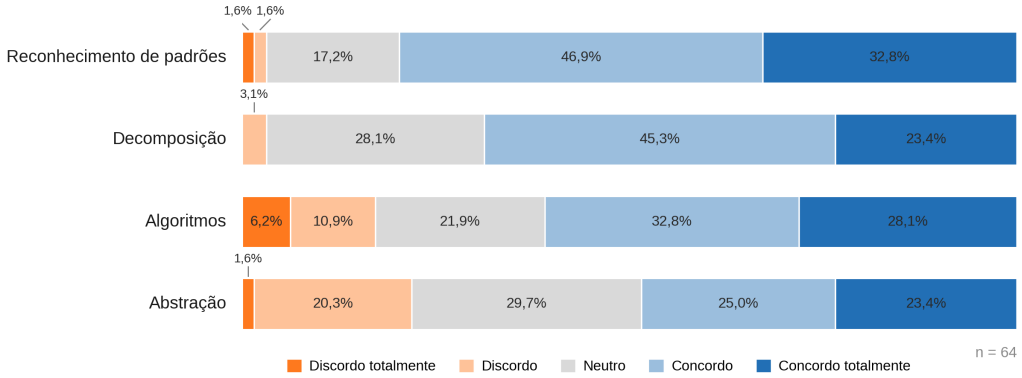
As respostas abertas associadas à recomendação não trataram diretamente do uso em outras turmas, mas da experiência vivenciada pelos respondentes. Entre os que recomendaram, A47 afirmou que "essas tecnologias de aprendizado ajudam a verificar como eu estou indo em relação ao meu desempenho na atividades propostas". A29, em posição

neutra, registrou que a experiência foi positiva, mas considerou que, no desenvolvimento computacional e na forma como aprendeu, o SoPensa ainda não alcançaria a pontuação máxima. Entre os que discordaram, A7 informou ter aprendido pouco com a proposta.

7.2.3 Desenvolvimento do PC

Em relação à percepção dos estudantes sobre o desenvolvimento do PC, a análise considerou os seus quatro componentes: decomposição, reconhecimento de padrões, abstração e algoritmo. A Figura 42 apresenta a distribuição das respostas.

Figura 42: Percepção dos estudantes sobre o desenvolvimento do PC



Fonte: Autoria Própria.

O reconhecimento de padrões registrou a maior concordância entre os quatro componentes, com 79,7%, seguido pela decomposição 68,7% e pelo algoritmo 60,9%. A abstração apresentou o menor índice, 48,4%, acompanhado de 29,7% de neutralidade e 21,9% de discordância, o maior percentual de discordância entre os componentes analisados. Nos três primeiros itens, a discordância ficou entre 3,1% e 17,2%, e a neutralidade variou entre 17,2% e 28,1%. Em quase todos os componentes, a categoria Concordo concentrou a maior parte das respostas, Concordo totalmente variou entre 23,4% e 32,8%.

Nas respostas abertas, A59 afirmou que "o primeiro passo para solucionar um problema é dividi-lo em partes menores para facilitar esse processo", trecho diretamente associado ao componente da decomposição. A19 relatou que "as atividades utilizadas ampliaram meu conhecimento e meu raciocínio".

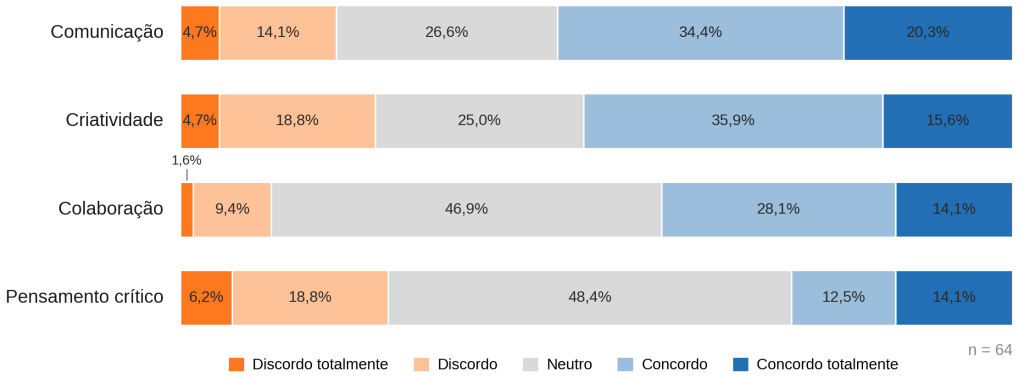
A fala de A26 trouxe dois posicionamentos: afirmou ter competência na resolução de problemas e no pensamento computacional, mas reconheceu dificuldade em "pensar fora da caixa e lançar pensamentos", referência explícita ao componente da abstração. A37 apresentou relato distinto: aprendeu a desmembrar problemas e dividi-los em partes, mas

relatou que, diante de dúvidas no grupo, os estudantes recorriam ao ChatGPT como apoio para encontrar respostas.

7.2.4 Desenvolvimento de SS

Quanto à percepção dos estudantes sobre o desenvolvimento das competências de SS, a análise considerou as quatro competências do *framework*: colaboração, comunicação, criatividade e pensamento crítico. A resolução de problemas, por seu caráter transversal, é apresentada ao final desta subseção. A Figura 43 apresenta a distribuição das respostas.

Figura 43: Percepção dos estudantes sobre o desenvolvimento de SS



Fonte: Autoria Própria.

A comunicação registrou a maior concordância entre as quatro competências, com 54,7%, seguida pela criatividade, com 51,5%. A colaboração apresentou 42,2% de concordância, acompanhada de 46,9% de neutralidade. O pensamento crítico registrou o menor índice de concordância, 26,6%, com 48,4% de neutralidade, a maior entre os itens analisados e 25,0% de discordância. Em parte dos itens, a categoria Concordo concentrou a maior proporção de respostas favoráveis, enquanto a categoria Concordo totalmente variou entre 14,1% e 20,3%. A discordância somada oscilou entre 11,0% e 25,0% ao longo das quatro competências.

Nas respostas abertas, A59 afirmou ter aprendido "a trabalhar em grupo e contribuir ao máximo". A49 relatou avanço no trabalho em equipe e na expressão de ideias ao grupo, além de maior capacidade de buscar soluções diferentes para os problemas. Enquanto A6 acrescentou que as atividades ajudaram a socializar e que foi bom trabalhar em equipe.

No que diz respeito à colaboração, duas falas ajudam a compreender o percentual de neutralidade observado nesse item, de 46,9%. A58 informou ter realizado parte das atividades individualmente, por não conseguir trabalhar adequadamente em grupo. A62

registrou que o desenvolvimento foi agradável quando o relacionamento com a turma era próximo, mas que, em outros momentos, segundo suas palavras, o grupo "travou" por falta dessa proximidade.

Quanto ao pensamento crítico, A59 foi o único respondente a abordá-lo diretamente nas respostas abertas, ao afirmar ter aprendido "a desenvolver o pensamento crítico e estratégico".

Além das quatro competências de SS, o instrumento incluiu um item transversal relativo à capacidade de enfrentar e buscar soluções para problemas mais complexos, aspecto associado ao objetivo geral do *framework*. Nesse item, (68,8%) dos estudantes assinalaram concordância, sendo 39,1% em concordo e 29,7% em concordo totalmente, com 18,8% de neutralidade e 12,5% de discordância, dos quais 1,6% em discordo totalmente. A resolução de problemas atravessa os componentes do PC e as competências de SS, e o índice observado acompanha a percepção favorável registrada nos demais aspectos. Nas respostas abertas, A51 registrou que as atividades foram dinâmicas, "sempre ajudando com resoluções de problemas".

7.3 Considerações finais do capítulo

Este capítulo analisou as percepções de estudantes e professores sobre o SoPensa após a intervenção nas duas turmas de Lógica de Programação, em conformidade com o OE5. A leitura conjunta das duas fontes, as entrevistas com os professores e o questionário aplicado aos estudantes, mostra um quadro convergente em sua orientação geral: os estudantes avaliaram de forma favorável a aceitação do *framework*, e os professores reconheceram seu potencial pedagógico e a reorganização que ele introduz na condução das aulas.

A convergência mais nítida entre as duas fontes diz respeito à abstração. Entre os estudantes, foi o componente do PC com menor concordância; entre os professores, foi apontada pelos dois como a principal dificuldade no desenvolvimento do PC. O encontro entre o dado quantitativo e a leitura qualitativa situa a abstração como o ponto sensível da aplicação, em que os avanços percebidos foram parciais. Movimento semelhante aparece nas SS, dimensão de percepção mais contida entre os estudantes e de avaliação divergente entre os professores, o que torna o desenvolvimento de SS o aspecto de leitura menos firme do capítulo, ainda que por razões distintas em cada fonte.

O capítulo também trouxe à tona condições que cercaram a aplicação, como o tempo no limite mínimo, as defasagens formativas anteriores e as limitações curriculares. Consi-

deradas em conjunto, as percepções sustentam a interpretação de que o *framework* operou, no cenário investigado, como estrutura de apoio à prática docente e à condução das atividades, sem que disso se afirme ganho de aprendizagem medido, porém, leitura coerente com o caráter qualitativo deste capítulo.

Os resultados das duas seções permitiram analisar as percepções de professores e estudantes sobre as contribuições do SoPensa para o ensino de programação, atendendo ao OE5, e, na seção 7.1, responderam também à questão que orientou as entrevistas. A discussão desses resultados em diálogo com a literatura sobre ensino de programação, PC e SS é desenvolvida no capítulo seguinte e o retorno ao problema de pesquisa é apresentado nas considerações finais da tese.

8 DISCUSSÃO DOS RESULTADOS

Este capítulo discute os resultados apresentados nos Capítulos 6 e 7, em diálogo com a literatura sobre ensino de programação, PC e SS, bem como o MSL do Capítulo 3 e o estudo empírico com professores do Capítulo 4. A discussão está organizada em torno dos principais achados, e não dos instrumentos de coleta, de modo que cada resultado seja interpretado a partir do cruzamento entre a literatura, o questionário aplicado aos estudantes, as entrevistas com os professores e os instrumentos observacionais da aplicação, incluindo as rubricas, a ficha de observação das fases e a autoavaliação dos times.

A triangulação adotada amplia os ângulos de interpretação sobre a mesma experiência, conforme Flick (2013), sem assumir a função de validação cruzada entre as fontes. Tal escolha mantém coerência com a ATR (Braun; Clarke, 2022), na qual divergências entre materiais são compreendidas como elementos analíticos relevantes. Assim, os resultados expressam percepções e registros situados no contexto investigado, sendo interpretados sem a pretensão de estabelecer relações causais.

8.1 A abstração como dificuldade persistente

A abstração concentrou a maior dificuldade percebida no desenvolvimento do PC ao longo da pesquisa. No diagnóstico inicial, foi o componente de menor adequação nas duas turmas, com 25,7% em Informática para Internet e 28,6% em Informática em Redes, enquanto o reconhecimento de padrões alcançou os maiores índices. Essa tendência reapareceu na Fase 3: na autoavaliação dos times, a abstração reuniu o maior número de respostas negativas, com três times, e, nas rubricas, foi registrada como dificuldade recorrente, associada à dificuldade de separar a estrutura essencial do problema. No questionário, obteve a menor concordância entre os quatro componentes, com 48,4%, e a maior discordância, com 21,9%. Nas entrevistas, os dois professores também a identificaram como o entrave central do percurso cognitivo. A recorrência em cinco registros de coleta reduz a possibilidade de resultado incidental.

O achado acompanha um padrão já descrito na literatura, tendo em vista que Wing (2014) e Montuori *et al.* (2024) situam a abstração no nível mais alto do PC, definindo-a como o processo responsável por lidar com a complexidade ao selecionar os aspectos relevantes do problema e generalizar soluções. A contrapartida desse papel central é a dificuldade de aprendizagem, recorrente entre iniciantes (Qian; Lehman, 2017; Cheah, 2020; Sim; Lau, 2022). Para Sim e Lau (2022), abstrair o problema a ser resolvido é a etapa mais desafiadora do percurso. Cheah (2020), por sua vez, localiza essa dificuldade ainda antes da codificação, quando o estudante não consegue dividir o problema em partes menores. Assim, ao registrar a abstração como ponto sensível, esta pesquisa aproxima-se de um resultado já descrito na literatura, em vez de contrariá-lo.

No contexto da EPTNM, a dificuldade ganha peso adicional. Os professores não interpretam o obstáculo apenas como traço do desenvolvimento cognitivo, mas como efeito associado às defasagens trazidas da formação básica, sobretudo em matemática, língua portuguesa e interpretação de texto. Essa relação é descrita na literatura (Medeiros; Falcão; Ramalho, 2021; Silva; Moreira, 2021), que trata a deficiência dos conhecimentos de base como traço característico do cenário brasileiro e relaciona os problemas de abstração ao raciocínio lógico pouco desenvolvido. No ensino técnico integrado, portanto, abstrair não ocorre de forma isolada, mas se combina a lacunas anteriores, ajudando a explicar os avanços parciais percebidos.

As falas dos estudantes acompanham essa leitura de modo direto. Um respondente reconheceu competência na resolução de problemas, mas relatou dificuldade em pensar "fora da caixa", em referência direta à abstração. Na autoavaliação aberta, um time de Informática para Internet anotou dificuldade para abstrair e registrou dúvida sobre esse ponto. Esses registros mostram estudantes cientes do próprio limite, e não alheios a ele, condição que torna o ponto passível de trabalho pedagógico.

Dois pontos ajudam a refinar essa interpretação. O primeiro é a concentração das respostas negativas sobre abstração na segunda coleta, ausentes na primeira. A dificuldade tornou-se explícita com a programação textual, sugerindo que as etapas desplugada e visual mantiveram a lacuna latente, revelada apenas diante do código em Python. Outra leitura possível é que os times passaram a avaliar suas limitações de forma mais crítica ao longo das fases. As duas interpretações são compatíveis com os dados, mas a coleta não permite indicar qual prevalece. O segundo ponto aparece no relato de A37, que registrou o uso do ChatGPT pelo grupo para buscar respostas. Embora pontual, o dado introduz uma ressalva ao argumento de precedência do raciocínio sobre a codificação, pois indica

possível delegação de parte da resolução antes da elaboração própria. A pesquisa não acompanhou esse uso, mas o registro aponta uma questão relevante para a autonomia cognitiva no ensino de programação.

O que o *framework* SoPensa produziu nesse aspecto foi distinto de uma solução imediata para o problema. Os estudantes não passaram a abstrair com facilidade. Passaram a perceber a necessidade de abstrair, movimento descrito por P2 ao registrar que a proposta despertou a necessidade de pensar de maneira abstrata. O ciclo de retomada da Fase 4 fez da abstração o eixo de grande parte das devoluções, sobretudo nos códigos em que os times não haviam separado a estrutura essencial do problema. Assim, a sequência das atividades desplugadas para as de programação ofereceu apoio à construção progressiva do conceito, em coerência com a aprendizagem significativa de Ausubel, Novak e Hanesian (1980) e com a base construcionista de Papert (1980) e Resnick (1996), que orientam o SoPensa. A contribuição do *framework*, nesse ponto, é de diagnóstico e mediação, não de superação plena da dificuldade.

Os dados das cinco fontes indicam que, em disciplinas introdutórias da educação técnica, a abstração demanda mediação planejada e explícita, e não apenas a exposição dos estudantes às atividades. O SoPensa cria condições para essa mediação ao tornar a dificuldade visível e ao devolvê-la ao percurso no ponto em que ela interrompe a passagem do raciocínio para o código.

8.2 Os componentes do PC são mais reconhecidos do que as competências de SS

Na percepção dos estudantes, os componentes do PC foram mais reconhecidos do que as competências de SS. Entre os componentes, reconhecimento de padrões, decomposição e algoritmos registraram os maiores índices de concordância, variando de 60,9% a 79,7%, enquanto a abstração permaneceu no menor patamar. Nas SS, os percentuais foram mais baixos, variando de 26,6% no pensamento crítico a 54,7% na comunicação. A diferença aparece com mais força na neutralidade: colaboração, com 46,9%, e pensamento crítico, com 48,4%, reuniram os maiores percentuais neutros do questionário. A autoavaliação dos times confirmou essa tendência, com mais respostas intermediárias nas competências de SS do que nos componentes do PC.

Essa diferença também é descrita na literatura, pois Tomić *et al.* (2019) observaram que os estudantes se mostram, em geral, menos engajados na aquisição de SS do que na

de habilidades técnicas e que os cursos de programação concentram atenção na dimensão técnica. Embora o PC não se confunda com habilidades técnicas, por se tratar de um processo cognitivo, seus componentes tendem a se manifestar de forma mais perceptível nas disciplinas de programação porque se vinculam diretamente à resolução de problemas, à estruturação de algoritmos e à passagem para o código. Ciancarini, Missiroli e Russo (2020) acrescentam que o PC foi historicamente tratado como habilidade individual, com menor espaço para o trabalho em equipe e para as SS em disciplinas introdutórias. O menor reconhecimento das SS pelos estudantes, portanto, é compatível com um campo que tende a dar menos visibilidade às SS.

A leitura da neutralidade exige cautela, pois o fato de a colaboração e o pensamento crítico terem reunido mais respostas neutras do que discordantes sugere dificuldade de autopercepção, e não rejeição às competências. Wang e Herzog (2023) caracterizam as SS como habilidades pouco quantificáveis e a literatura aponta fragilidades persistentes na sua avaliação e mensuração (Dadelo, 2025; Villegas, 2024). Essa característica torna previsível que os estudantes hesitem em afirmar o próprio desenvolvimento quando solicitados a avaliá-lo. A neutralidade indica competências difíceis de perceber e de medir, mas não competências ausentes.

As entrevistas reforçam esse quadro, embora apresentem percepções diferentes entre os professores: P2 relatou desenvolvimento das SS, sobretudo na colaboração, enquanto P1 não percebeu mudança, atribuindo essa ausência de percepção à baixa frequência de contato com a turma. A divergência, tratada como parte da análise e não como inconsistência, mostra que a percepção sobre SS depende das condições de observação de cada professor.

Esses dados indicam uma implicação para o desenho pedagógico: inserir as SS no planejamento não basta para que os estudantes as reconheçam. Por serem menos observáveis do que a identificação de padrões ou a escrita de algoritmos, essas competências demandam mediação intencional durante a atividade. A SBC (2025) registra que muitos professores da área não receberam, na formação inicial, subsídios para trabalhar as SS ou empregar metodologias ativas, o que ajuda a explicar seu baixo reconhecimento. A BNCC e documentos orientadores do MEC preveem o desenvolvimento dessas competências de forma transversal, interdisciplinar e intencional (Brasil, 2017b, 2021), reforçando que esse reconhecimento depende de condução planejada, e não apenas da presença das SS nas atividades. O período de um semestre, indicado como tempo mínimo na avaliação de P1, também limita o que pode ser percebido. No construcionismo de Papert, a colaboração

aparece vinculada à produção de artefatos. O *framework* atua nesse ponto, ao criar fases em que as SS deixam de aparecer apenas de modo implícito e passam a ser trabalhadas explicitamente pelo professor.

8.3 A progressão das atividades desplugadas para as plugadas

A sequência proposta pelo *framework* parte de atividades desplugadas e avança para atividades plugadas, primeiro em ambiente visual e depois em linguagem textual. A autoavaliação dos times registra essa progressão entre as Fases 2 e 3. As respostas afirmativas em algoritmos passaram de oito para dez, e as de colaboração, de sete para nove. As rubricas também indicam esse avanço, com algoritmos mais organizados na etapa plugada do que na desplugada.

O início sem computador encontra apoio na literatura sobre computação desplugada (Brackmann, 2018; Sim; Teow; Lau, 2021). Brackmann (2018) descreve essa abordagem como caminho para introduzir conceitos de PC sem a barreira da sintaxe, recurso adequado ao perfil identificado no diagnóstico, em que poucos estudantes tinham experiência prévia em programação. A construção de uma base comum antes do contato com o código permitiu trabalhar decomposição e reconhecimento de padrões em situações concretas, preparando a transição para a programação.

A passagem para o ambiente visual foi a etapa em que se observou maior envolvimento dos estudantes. As rubricas registraram, no ambiente de programação, algoritmos mais organizados e maior presença de comunicação e colaboração, ainda que a abstração permanecesse como dificuldade persistente. A construção do próprio jogo no Scratch, mostrada na Figura 36, abriu espaço para a criatividade dos times. Resnick *et al.* (2009) descrevem a programação visual em blocos como meio de aproximar iniciantes da lógica de programação ao reduzir a carga sintática, recurso também explorado por Bodaker e Rosenberg-Kima (2023) em atividades com crianças em duplas.

Schwartz *et al.* (2024) acrescentam que abordagens tangíveis e visuais introduzem a programação e permitem desenvolver, ao mesmo tempo, componentes do PC e competências de SS, como comunicação e colaboração, relação também observada na atividade em Scratch. Ribeiro *et al.* (2020) apontam que a abordagem lúdica e o sentido de progressão funcionam como recursos de engajamento, em contraste com a desmotivação que Queiroz, Rodrigues e Coutinho (2018) associam aos erros contínuos e à falta de domínio

nas atividades iniciais. A etapa em Python exigiu maior organização lógica e atenção à sequência, e vários times produziram códigos parcialmente funcionais, retomando os desafios de abstração discutidos na seção anterior. Attard e Busuttil (2020) apontam que a passagem do ambiente em blocos para a linguagem textual é um momento sensível e deve ocorrer gradualmente, para que o estudante não perca o apoio da programação visual. Na aplicação, a redução da carga sintática favoreceu o envolvimento e a organização dos algoritmos, mas apenas adiou a dificuldade de abstração para a escrita do código. Esse resultado restringe o alcance da vantagem descrita por Resnick *et al.* (2009). Assim, o *framework* deve tratar a passagem para a linguagem textual como uma etapa que exige mediação específica, e não como um prolongamento automático do ambiente visual.

A aproximação dos estudantes à disciplina, relatada pelos professores, não ocorreu em uma turma previamente engajada. P2 descreveu dificuldades prévias de interação na turma, atribuindo-as, em parte, ao efeito da pandemia sobre essa geração. O envolvimento observado, nesse contraste, foi construído ao longo das fases e não estava necessariamente presente na entrada. P1 situou esse envolvimento no presente da disciplina, ao notar o uso informal do vocabulário técnico entre os estudantes, enquanto P2 o relacionou ao horizonte profissional. As duas leituras descrevem o engajamento em momentos diferentes.

A progressão das atividades indica que a organização do percurso do concreto para o abstrato favoreceu a construção dos conceitos. Essa organização se aproxima da AS, ao constituir uma base prévia antes do contato com o novo, e do construcionismo, ao deslocar a aprendizagem para a construção de artefatos no ambiente visual e na linguagem textual. O ganho observado em algoritmos e colaboração na passagem para a etapa plugada acompanha essa interpretação, embora a abstração tenha permanecido como ponto de atenção em todas as etapas. A literatura, contudo, não é unânime quanto ao peso do método. Montuori *et al.* (2024), em revisão sistemática, observam que os efeitos sobre o PC não variaram de forma consistente entre abordagens desplugadas, virtuais ou baseadas em robótica. Tal resultado não contradiz o que se defende aqui, pois o *framework* não pressupõe a superioridade de um método, mas oferece ao professor um percurso gradual para introduzir a programação em turmas com pouca ou nenhuma experiência prévia. A sequência do concreto ao abstrato orienta a mediação docente, o que difere de uma alegação de eficácia de método, e evita a exposição imediata ao código.

8.4 Colaboração e comunicação como competências que demandam mediação intencional

Ao final das Fases 2 e 3, vários times registraram, nas respostas abertas da autoavaliação, que a participação dos integrantes nem sempre ocorreu de forma equilibrada e que a comunicação interna poderia ter sido mais clara. Esse reconhecimento, feito pelos próprios times, antecipa o argumento desta seção, pois mostra a colaboração como prática que não se realiza pela simples formação do grupo.

Os dados do questionário acompanham essa percepção. A comunicação obteve concordância mais alta, com 54,7%, enquanto a colaboração ficou em 42,2%, com neutralidade elevada, de 46,9%. Duas falas ajudam a compreender essa percepção. O estudante A58 relatou ter realizado parte das atividades sozinho, por não conseguir trabalhar adequadamente em grupo, e o estudante A62 descreveu que o time avançava quando havia maior interação entre os integrantes, mas travava na ausência dela. O trabalho em equipe apareceu, assim, como experiência desigual na turma.

A literatura também indica que a colaboração depende de mediação e não apenas da formação de grupos. Younis *et al.* (2019) e Apeanti e Essel (2021) relatam ganhos em comunicação, liderança e resolução de conflitos quando a aprendizagem baseada em projetos é conduzida de forma colaborativa, o que pressupõe orientação intencional.

Ciancarini, Missiroli e Russo (2020) enfatizam que o trabalho em equipe costuma ser pouco considerado em cursos introdutórios de programação e, por vezes, até desencorajado. Hsu *et al.* (2022) observam que reunir estudantes em grupos não garante o desenvolvimento das competências esperadas, pois esse desenvolvimento depende de um desenho cuidadoso e de uma abordagem apropriada. Cruz *et al.* (2024) acrescentam que ainda há poucos estudos sobre como os componentes do PC auxiliam no desenvolvimento do trabalho em equipe. A centralidade da comunicação observada no questionário converge com Espina-Romero *et al.* (2023), que definem as SS como competências sociais associadas à capacidade de comunicar. Considerando a finalidade profissional da formação técnica, Coelho *et al.* (2024) constatam que líderes de projeto consideram a comunicação e o trabalho em equipe entre as competências mais críticas, o que reforça a pertinência de desenvolvê-las já na disciplina.

A identificação, pelos próprios times, de suas falhas internas é em si um dado relevante para a análise. Quando um grupo percebe que sua comunicação interna falhou ou que a condução se concentrou em poucos integrantes, realiza uma análise crítica sobre a própria

prática coletiva. Esse registro, feito na Fase 4, indica exercício de pensamento crítico, ainda que sinalize uma colaboração incompleta.

A composição de times equilibrados na Fase 1, a partir do diagnóstico, organizou o ponto de partida, mas não garantiu, por si, a distribuição efetiva do trabalho. A colaboração e a comunicação se mostraram competências que demandam mediação sustentada ao longo das atividades. As fases do *framework* oferecem ao professor uma estrutura para exercer essa mediação de forma contínua, e não pontual. Diferentemente do tratamento historicamente individualizado do PC, já mencionado, o SoPensa organiza o desenvolvimento dos componentes do PC por meio do trabalho conjunto. P2 percebeu que essa organização potencializou o processo cognitivo, da decomposição aos algoritmos, funcionando como contraponto a esse tratamento, ainda que no plano da percepção docente.

8.5 O *framework* na prática docente e os condicionantes da educação profissional técnica

Os dois professores descreveram o SoPensa como apoio à organização da prática docente, ao sistematizar o que antes ocorria de forma difusa. Essa percepção encontra respaldo em SBC (2025), que registra a distância entre o conhecimento técnico e o domínio pedagógico como obstáculo à integração de abordagens voltadas à colaboração, à empatia e à comunicação, dificuldade que Hudin (2023) também associa ao preparo limitado do professor para conduzir o ensino de programação. Diante dessa lacuna formativa, cada docente mobilizava repertório próprio e a sistematização atribuída ao *framework* oferece um método comum de trabalho.

Parte dessa reorganização se apoia na leitura prévia das turmas. A Fase 1 permitiu ao professor partir do perfil real dos estudantes, e não apenas dos conhecimentos previstos na ementa, ao identificar características e dificuldades antes do trabalho com a sintaxe. O diagnóstico das duas turmas tornou essa leitura mais concreta. O levantamento apontou o reconhecimento de padrões como ponto forte inicial e a abstração como fragilidade comum, além de registrar poucos estudantes com experiência prévia em programação, cinco em Informática para Internet e quatro em Informática em Redes. Entre as competências de SS, sobressaíram o pensamento crítico em Informática para Internet e a colaboração em Informática em Redes, o que orientou a composição equilibrada dos times.

Morais, Mendes Neto e Osório (2020) observam que turmas numerosas e heterogêneas dificultam a tarefa do professor e que identificar as dificuldades dos alunos contribui para

orientar a condução da disciplina, ponto reforçado por Moraes, Costa e Scholz (2022) em mapeamento do ensino introdutório de programação nos níveis técnico e superior no Brasil. O diagnóstico inicial do *framework* opera nessa direção, ao oferecer base para decisões pedagógicas que antes dependiam apenas da sensibilidade do docente. Essa leitura prévia se aproxima da aprendizagem significativa, ao buscar os conhecimentos prévios, ou subsunçores, sobre os quais o novo conteúdo pode se ancorar (Moreira, 2006a; Ausubel; Novak; Hanesian, 1980).

A especificidade do ensino técnico integrado ajuda a compreender esse resultado. Ramos (2008) e Moura (2008) caracterizam o ensino médio integrado como projeto de articulação entre formação geral e formação técnica, em tensão com a dualidade histórica que separa educação e trabalho na escola básica brasileira. O SoPensa atua sobre essa separação no âmbito da disciplina, ao articular componentes do PC, diretamente associados à programação, com competências de SS, vinculadas à formação humana e profissional.

Os condicionantes estruturais identificados por P2 são anteriores ao *framework* e vão além do seu alcance. O primeiro é a defasagem formativa anterior, que P2 associa a deficiências em matemática, língua portuguesa e interpretação de texto. A literatura nacional aponta essa deficiência dos conhecimentos de base como característica do cenário brasileiro e mostra como tais lacunas comprometem o raciocínio lógico necessário à programação (Medeiros; Falcão; Ramalho, 2021; Silva; Moreira, 2021). O segundo condicionante aparece no currículo, na lacuna que P2 nomeia ao observar que os planos de ensino e as ementas não estão voltados ao desenvolvimento do PC e das SS, condição que contrasta com documentos normativos e orientadores sobre computação e competências de SS (Brasil, 2017b, 2022; SBC, 2025).

Diante desses condicionantes, o alcance do *framework* precisa ser delimitado com cautela. O SoPensa não corrige defasagens originadas na formação básica nem reescreve a ementa da disciplina. O que a aplicação mostrou é que o *framework* torna essas defasagens visíveis e pedagogicamente tratáveis no contexto da disciplina, ao identificá-las no diagnóstico e ao retomá-las nas fases correspondentes. A contribuição ocorre no plano da prática docente, e não no plano da política educacional.

Mesmo diante dessas condições, a avaliação dos dois professores foi convergente quanto à utilidade do *framework* para a prática docente, ainda que com recomendações distintas: P1 sugeriu a análise individual do perfil dos alunos, enquanto P2 indicou a aplicação prévia à ementa. Esse resultado se alinha à literatura que aponta a escassez de propostas estruturadas para o contexto. Maitz, Paleczek e Danielowitz (2022) registram a raridade

de intervenções que combinam programação, PC e SS, e Ciancarini, Missiroli e Russo (2020) observam que sistemas centrados em habilidades técnicas encontram dificuldade em promover as SS. Nouri *et al.* (2020) acrescentam que essas competências tendem a ser desenvolvidas de forma implícita, sem orientação que apoie o professor. Rege, Salgado e Viterbo (2025) apontam a ausência de estudos voltados especificamente ao ensino técnico de nível médio, lacuna que esta pesquisa procurou ocupar. As avaliações sobre o tempo também reforçam essa delimitação: P1 considerou a aplicação suficiente, ainda que no limite mínimo, e P2 propôs ampliá-la, ao sugerir prazo maior e aplicação anterior ao início da disciplina. A projeção de ganhos em aplicações mais extensas, comum aos dois, remete à carga de conteúdos da educação técnica, em que o conteúdo operacional disputa espaço com o desenvolvimento das competências aqui discutidas.

8.6 Considerações finais do capítulo

Este capítulo discutiu os resultados dos Capítulos 6 e 7 em diálogo com a literatura sobre ensino de programação, PC e SS, organizando a interpretação por achados. Em conjunto, os resultados indicam a abstração como a principal dificuldade do percurso, presente em todas as fontes e ainda dependente de mediação planejada. Também mostram que o PC foi mais reconhecido pelos estudantes do que as SS, cuja percepção é dificultada pela menor observabilidade dessas competências. A progressão das atividades desplugadas para as plugadas favoreceu a construção dos conceitos e o engajamento, enquanto a colaboração e a comunicação se confirmaram como competências que demandam condução planejada, e não apenas agrupamento. Na EPTNM, esses achados indicam que o *framework* apoiou a prática docente, com avanços parciais e condicionantes que ele torna visíveis, sem ter alcance para corrigi-los sozinho.

As duas teorias da proposta ajudam a interpretar esses resultados. A AS sustenta que novos conhecimentos se ancoram quando encontram conhecimentos prévios adequados, pressuposto mobilizado na leitura inicial do perfil da turma e limitado pelas defasagens formativas, especialmente nos pontos em que a abstração se mostrou mais difícil. O construcionismo sustenta a aprendizagem como construção ativa por meio de artefatos, ideia observada no engajamento durante a passagem para a programação e na colaboração favorecida pelo trabalho conjunto. O retorno ao problema de pesquisa e o exame de como cada objetivo específico foi atendido são apresentados no capítulo seguinte.

9 CONSIDERAÇÕES FINAIS

Esta tese investigou como integrar os componentes do PC e as competências de SS no ensino de programação na educação profissional técnica de nível médio. A pesquisa, desde a revisão da literatura até a aplicação em sala de aula, indica que essa integração não garante ganhos de aprendizagem, mas organiza a prática docente. Os componentes do PC e as competências de SS cumprem esse papel quando deixam de ser aspectos pouco explicitados e passam a objetos explícitos do trabalho do professor. Esse movimento é realizado pelo *framework* SoPensa ao diagnosticar o perfil da turma, sequenciar as atividades desplugadas e plugadas e tornar visíveis as dificuldades, em especial a abstração, no ponto em que elas interrompem o raciocínio.

9.1 Retomada do problema e dos objetivos

A questão que orientou a pesquisa, como integrar os componentes do PC e as competências de SS no ensino de programação na educação profissional técnica de nível médio, foi respondida ao longo dos capítulos, a partir do objetivo geral de analisar a integração entre o PC e as SS no ensino de programação e de propor e analisar as contribuições de um *framework* conceitual-operacional.

Quanto ao objetivo geral, os resultados indicaram que a integração entre PC e SS pode ser realizada por meio da mobilização conjunta desses elementos nas atividades de programação. Essa integração é promovida pelo ciclo formativo do SoPensa e pela mediação intencional do professor, de modo que componentes do PC, como decomposição, abstração e algoritmos, sejam trabalhados juntamente com competências de SS, como colaboração, criatividade e pensamento crítico.

O percurso da pesquisa atendeu aos cinco objetivos específicos. O MSL do Capítulo 3 identificou as estratégias, as lacunas e os conceitos da aplicação integrada do PC e das SS, ao mostrar a concentração dos estudos na educação básica e a ausência de investigações voltadas ao ensino técnico de nível médio. O estudo empírico do Capítulo 4 investigou as

percepções e práticas dos professores dos Institutos Federais, revelou o desenvolvimento pouco sistematizado desses elementos e originou as diretrizes da proposta. Com base no referencial, no mapeamento e nessas percepções, o Capítulo 5 propôs e estruturou o *framework* SoPensa em quatro fases, do diagnóstico à retomada. O Capítulo 6 implementou a proposta em duas turmas de um Instituto Federal e a aplicação mostrou sua viabilidade e a progressão das atividades desplugadas para as plugadas. Os Capítulos 7 e 8 examinaram as percepções de professores e estudantes e apontaram o reconhecimento maior do PC do que das SS, a abstração como dificuldade persistente e a reorganização da prática docente como principal ganho percebido.

Em conjunto, essas respostas sustentam a tese de que, na educação profissional técnica de nível médio, a integração entre o PC e as SS apoia o ensino de programação primeiro pela reorganização da mediação docente e, em seguida, pelo desenvolvimento percebido do estudante. O *framework* não garante a aprendizagem, mas permite trabalhar de forma explícita aspectos do PC e das SS que antes permaneciam implícitos. Já a abstração se confirma como a principal dificuldade cognitiva identificada na aplicação, ponto em que a mediação planejada se faz mais necessária. Desse achado decorre a recomendação de tratar a abstração com mediação explícita e com etapas intermediárias entre a programação visual em blocos e o código textual, em vez de supô-la adquirida pela simples exposição às atividades.

9.2 Contribuições

No plano teórico, o *framework* SoPensa constitui uma estrutura conceitual-operacional que integra os componentes do PC e as competências de SS com foco na mediação docente, fundamentada na AS e no Construcionismo. Com isso, contribui para preencher uma lacuna apontada pelo mapeamento, relativa à ausência de propostas que integrem esses dois eixos no ensino técnico, e não apenas em níveis anteriores ou com foco no estudante. Ao situar a integração entre PC e SS no trabalho de mediação docente e documentar uma aplicação em contexto real ainda pouco explorado na literatura, a tese apresenta uma forma de compreender como esses elementos podem orientar a prática docente no ensino de programação na EPTNM.

No plano prático, o SoPensa oferece ao professor um modo organizado de conduzir o ensino de programação, do diagnóstico do perfil da turma à retomada das produções, convertendo em procedimento organizado práticas que antes ocorriam de modo pouco

sistematizado. Ao reunir os componentes do PC e as competências de SS em uma mesma proposta de ensino, o *framework* aproxima a dimensão técnica e a dimensão interpessoal da formação, em linha com o perfil de egresso da EPTNM.

Os instrumentos de acompanhamento, como diagnóstico, rubrica, ficha de observação e autoavaliação dos times, ficam disponíveis para reuso, e as diretrizes derivadas do estudo com professores dos IFs oferecem às instituições da Rede Federal uma referência aplicável às disciplinas iniciais de programação, considerando as condições próprias da educação técnica. Como forma de ampliar o acesso à proposta, o *framework* e seus instrumentos foram reunidos em um sítio eletrônico de acesso aberto, desenvolvido no âmbito desta pesquisa e concebido para facilitar a consulta e o reuso por professores.[1] Uma síntese das telas encontra-se no Apêndice J. Trata-se de uma versão inicial, cuja avaliação de usabilidade não integrou o escopo desta pesquisa. A contribuição tende a ser mais perceptível nas disciplinas introdutórias e para professores sem formação pedagógica específica, embora os docentes que conduziram a aplicação fossem experientes.

No contexto da Educação Profissional e Tecnológica, essas contribuições ampliam a perspectiva de formação no ensino de programação ao considerar que a preparação para o trabalho não se restringe ao domínio de conhecimentos técnicos. Na EPTNM, essa perspectiva é particularmente relevante, pois a formação dos estudantes envolve tanto a aquisição de conhecimentos específicos quanto o desenvolvimento de competências necessárias à resolução de problemas, à colaboração, à comunicação e à adaptação a diferentes situações profissionais. Nos IFs, essa perspectiva reforça o desafio de desenvolver práticas de ensino de programação que considerem, de forma intencional, essas diferentes dimensões da formação. Nesse contexto, o SoPensa oferece uma possibilidade de integração entre PC e SS no ensino de programação, mas sua aplicação também evidencia que esse processo não se resolve pela adoção isolada de um *framework*. A consolidação dessa perspectiva requer continuidade das ações, formação docente e condições pedagógicas que favoreçam sua incorporação às práticas de ensino. Assim, a contribuição da tese não está em propor uma solução definitiva para a formação em programação, mas em oferecer uma referência sistematizada para avançar nessa direção no contexto da EPTNM.

[1] Produto técnico derivado desta tese. Disponível em: https://sopensa.netlify.app. Acesso em: 30 jun. 2026.

9.3 Limitações da pesquisa

As limitações desta pesquisa delimitam o alcance dos resultados e devem ser consideradas na interpretação dos achados. A primeira refere-se ao contexto de aplicação do SoPensa, realizado em duas turmas, com dois professores, em um único campus de um Instituto Federal e ao longo de um semestre letivo. Esse recorte permitiu analisar a proposta em contexto real de ensino, mas restringe a possibilidade de generalização dos resultados para outros contextos da educação profissional técnica de nível médio.

Outra limitação refere-se ao caráter cíclico da pesquisa-ação. Embora a pesquisa tenha contemplado as etapas de diagnóstico, planejamento, intervenção e avaliação, foi realizado um único ciclo de aplicação, sem uma nova intervenção destinada a incorporar os resultados da avaliação e retroalimentar o processo. Assim, não foi possível verificar, em um ciclo posterior, como os ajustes decorrentes da experiência inicial poderiam modificar a aplicação do *framework.*

Também deve ser considerada a posição do pesquisador no estudo, uma vez que ele participou da elaboração do SoPensa e acompanhou o processo de aplicação, oferecendo preparação e suporte aos professores durante a aplicação. Essa proximidade constitui uma condição a ser considerada na interpretação dos dados, especialmente nos registros e análises qualitativas. Além disso, a presença e o suporte do pesquisador durante a aplicação não permitem avaliar, neste estudo, em que medida os professores conseguiriam utilizar o *framework* de forma autônoma sem esse acompanhamento.

Quanto aos dados, os resultados foram baseados predominantemente em percepções dos participantes e em registros descritivos, não tendo sido utilizados instrumentos específicos para mensurar objetivamente a aprendizagem dos estudantes. Dessa forma, os resultados relacionados ao desenvolvimento do PC e das SS devem ser compreendidos como percepções e evidências observadas no contexto da aplicação, e não como medidas objetivas de ganho de desempenho. Além disso, os dados autorrelatados podem estar sujeitos a vieses de autopercepção e desejabilidade social.

As condições de acompanhamento também constituíram uma limitação. Um dos professores teve menor tempo de contato com a turma durante a intervenção, o que pode ter influenciado sua percepção sobre o desenvolvimento das SS e contribuído para diferenças entre os relatos dos participantes. Somam-se a isso as dificuldades formativas anteriores dos estudantes e as limitações curriculares do contexto investigado, aspectos que ultrapassam o alcance imediato do *framework* e não podem ser atribuídos exclusivamente

à proposta.

Por fim, o uso de assistentes de inteligência artificial pelos estudantes foi observado de forma pontual, mas não acompanhado sistematicamente. Assim, não foi possível determinar sua influência sobre o desenvolvimento do raciocínio computacional, especialmente sobre a abstração e a passagem da programação em blocos para o código textual. Essas limitações não invalidam os resultados obtidos, mas delimitam sua interpretação e indicam as condições em que as contribuições do SoPensa foram analisadas.

9.4 Trabalhos futuros

A partir dos resultados obtidos e das limitações identificadas, algumas possibilidades de continuidade são propostas. A primeira consiste na realização de novos ciclos de aplicação do SoPensa, incorporando os resultados e ajustes identificados em cada ciclo, de modo a fortalecer a retroalimentação característica de uma pesquisa-ação. A extensão do estudo a diferentes turmas, professores, campi e instituições permitirá analisar a utilização do *framework* em outros contextos da educação profissional técnica de nível médio e verificar a consistência de seus resultados em condições distintas.

Como a abstração se manteve como a principal dificuldade, sobretudo na passagem da programação em blocos para o código textual, propõe-se desenhar e testar atividades intermediárias para essa transição, voltadas a apoiar o estudante no momento em que o raciocínio ainda não se converte em código. Trata-se de atuar sobre o ponto em que a dificuldade se manifesta, e não apenas sobre seu registro nos instrumentos do *framework*.

Outra possibilidade consiste na validação empírica do instrumento de formação de times, verificando em que medida a composição por médias equiparadas de PC e de SS favorece uma distribuição mais justa da participação entre os integrantes. Essa proposta decorre da experiência de aplicação, na qual a composição inicial dos times não foi suficiente para assegurar o envolvimento equilibrado de todos.

Também poderá ser investigada a flexibilização das formas de acompanhamento previstas no SoPensa, mantendo o registro por time como estratégia principal, mas permitindo, de forma opcional, o olhar individual sobre o estudante em situações nas quais o professor identifique a necessidade de uma análise mais específica de seu desenvolvimento.

A observação da aplicação por períodos mais longos permitirá identificar mudanças que não se manifestam em um único semestre letivo. Em outra direção, estudos futuros

podem incorporar instrumentos específicos para avaliar objetivamente o desenvolvimento do PC e das SS, complementando as percepções dos participantes.

Considerando ainda a presença crescente da inteligência artificial no ensino de programação, propõe-se investigar de forma sistemática como o uso dessas ferramentas interfere no desenvolvimento do PC, especialmente quando o estudante recorre a respostas prontas em vez de construir a solução, aspecto que aproxima o tema da autonomia cognitiva.

Por fim, o escopo do SoPensa pode ser ampliado para contemplar de forma mais sistemática diferentes habilidades técnicas e cognitivas relacionadas ao ensino de programação, bem como as competências de SS que não integraram o recorte deste estudo, conforme registrado na Tabela 9.

9.5 Produções acadêmicas

As produções acadêmicas desenvolvidas ao longo do percurso da pesquisa estão organizadas em três grupos: artigos publicados, artigos aceitos para publicação e outras publicações.

Artigos publicados

- Rege, Salgado, Viterbo 2025. "Pensamento Computacional e Soft Skills no Contexto do Ensino de Programação: Um Mapeamento Sistemático." In Anais do XXXVI SBIE-CBIE 2025.
- Rege, Salgado, Viterbo 2024. "Três Anos de Código: Uma Análise das Percepções, Desafios e Expectativas no Ensino de Programação." In Anais do XXX WIE-CBIE 2024.
- Rege, Salgado, Viterbo 2023. "Pensamento Computacional no Contexto da Educação Brasileira: um Mapeamento Sistemático da Literatura nos Diferentes Níveis de Ensino." Revista RENOTE, 2023.
- Rege et al. 2023. "O perfil dos ingressantes nos cursos técnicos integrados ao ensino médio em Informática para Internet." In Anais do XXIX WIE-CBIE 2023.

Artigos aceitos para publicação

- Rege, Salgado, Viterbo 2026. "SoPensa: um framework de pensamento computacional e soft skills na percepção docente." XXXVII SBIE-CBIE 2026.

- Rege et al. 2026. “Ensino de programação no técnico integrado e no superior: um survey nacional com docentes dos Institutos Federais.” XXXVII do SBIE-CBIE 2026.

- Rege et al. 2026. “Framework SoPensa para o ensino de lógica de programação: um relato de experiência no ensino técnico integrado.” XXXII WIE-CBIE 2026.

Outras Publicações

- Rege et al. 2023. “Uma Abordagem Baseada no Scrum: Relato de Experiência no Acompanhamento das Atividades de ensino em Disciplinas de Computação.” In Anais do XXIX WIE-CBIE 2023.

REFERÊNCIAS

ALVES, Havana Diogo. Análise da aplicação de metodologias ativas em disciplina de lógica de programação. *In:* SBC. ENCONTRO Nacional de Computação dos Institutos Federais (ENCompIF). [*S. l.: s. n.*], 2023. p. 85–92.

APEANTI, Wilson Osafo; ESSEL, D. Learning computer programming using project-based collaborative learning: Students' experiences, challenges and outcomes. **International Journal for Innovation Education and Research**, v. 9, n. 8, p. 191–207, 2021.

ASTOLFI, Gilberto; JUNIOR, Dejahyr Lopes. Ensino de Linguagem de Programação com Ênfase na Aprendizagem Significativa. *In:* ANAIS do XXIV Workshop sobre Educação em Computação. Porto Alegre: SBC, 2016. p. 2106–2115. DOI: 10.5753/wei.2016.9654. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/9654.

ATTARD, Lara; BUSUTTIL, Leonard. Teacher perspectives on introducing programming constructs through coding mobile-based games to secondary school students. **Informatics in Education**, Vilnius University Institute of Data Science e Digital Technologies, v. 19, n. 4, p. 543–568, 2020.

AUSUBEL, David P. The psychology of meaningful verbal learning. Grune & Stratton, 1963.

AUSUBEL, David P. **Aquisição e retenção de conhecimentos: uma perspectiva cognitiva**. Lisboa: Plátano, 2003.

AUSUBEL, David Paul; NOVAK, Joseph D; HANESIAN, Helen. **Psicologia educacional**. [*S. l.*]: Interamericana, 1980.

AVILA, Christiano Martino Otero; COSTA CAVALHEIRO, Simone André da. Rubrica construcionista para avaliação de atividades didáticas. *In:* SBC. ANAIS do XXX Workshop sobre Educação em Computação. [*S. l.: s. n.*], 2022. p. 310–321.

AZEVEDO DIAS, Helder Daniel de; SOUZA ARAÚJO, Josivaldo de. Um Relato de Experiência do Uso de Metodologias Ativas na Construção do Pensamento Computacional Paralelo. **International Journal of Computer Architecture Education**, v. 13, n. 1, p. 7–16, 2024.

BARR, David; HARRISON, John; CONERY, Leslie. Computational thinking: A digital age skill for everyone. **Learning & Leading with Technology**, ERIC, v. 38, n. 6, p. 20–23, 2011.

BARR, Valerie; STEPHENSON, Chris. **Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads, 2 (1), 48–54**. [*S. l.: s. n.*], 2011.

BBC. **Abstraction: What is abstraction?** 2024. Disponível em: https://www.bbc.co.uk/bitesize/guides/zttrcdm/revision/1. Acesso em: 2 jul. 2024.

BERSSANETTE, J.H.; FRANCISCO, Antonio. Proposta de Abordagem Prática para o Ensino de Programação Baseada em Ausubel. *In:* p. 398. DOI: 10.5753/cbie.sbie.2018.398.

BERSSANETTE, João Henrique; FRANCISCO, Antonio Carlos de. Active learning in the context of the teaching/learning of computer programming: A systematic review. **Journal of Information Technology Education: Research**, v. 20, p. 201–220, 2021.

BODAKER, Liat; ROSENBERG-KIMA, Rinat B. Online pair-programming: Elementary school children learning scratch together online. **Journal of Research on Technology in Education**, Taylor & Francis, v. 55, n. 5, p. 799–816, 2023.

BOONE JR, Harry N; BOONE, Deborah A. Analyzing likert data. **The Journal of extension**, v. 50, n. 2, p. 48, 2012.

BRACKMANN, Christian Puhlmann. Pensamento computacional desplugado: Ensino e avaliacao na educacao primaria espanhola. **Journal on Computational Thinking (JCThink)**, v. 2, n. 1, p. 36–36, 2018.

BRASIL. **Catálogo Nacional de Cursos Técnicos**. 4. ed. Brasília, 2020. Atualizado em 22 de outubro de 2025. Aprovado pela Resolução CNE/CEB nº 2, de 15 de dezembro de 2020. Disponível em: https://www.gov.br/mec/pt-br/assuntos/ept/catalogo-nacional-de-cursos-tecnicos.

BRASIL. **Educação Profissional e Tecnológica (EPT)**. [*S. l.: s. n.*], 2024. Portal do Ministério da Educação, Assuntos: EPT. Disponível em: https://www.gov.br/mec/pt-br/assuntos/ept.

BRASIL. **Educação Profissional Técnica de Nível Médio Integrada ao Ensino Médio: Documento Base**. Brasília, dez. 2007. Coordenação editorial de Dante Henrique Moura. Texto de Dante Henrique Moura, Sandra Regina de Oliveira Garcia e

Marise Nogueira Ramos. Disponível em:
http://portal.mec.gov.br/setec/arquivos/pdf/documento_base.pdf.

BRASIL. Lei nº 11.892, de 29 de dezembro de 2008. Institui a Rede Federal de Educação Profissional, Científica e Tecnológica, cria os Institutos Federais de Educação, Ciência e Tecnologia, e dá outras providências. **Diário Oficial da União**, Poder Legislativo Brasília, DF, v. 145, n. 252, p. 1–1, 2008.

BRASIL. **Lei nº 13.415, de 16 de fevereiro de 2017**. [*S. l.: s. n.*], 2017. https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2017/lei/l13415.htm. Altera as diretrizes e bases da educação nacional. Acessado em 03/04/2025.

BRASIL. **Manual de Implementação das Competências Socioemocionais**. [*S. l.*], 2021. Disponível em: Portal do MEC. Acesso em: 13 nov. 2024.

BRASIL. **Plataforma Nilo Peçanha. 2024**. [*S. l.: s. n.*], 2024. https://www.gov.br/mec/pt-br/pnp. Acesso em: 22 de abril de 2024.

BRASIL. **Resolução CNE/CEB nº 1, de 4 de outubro de 2022**. [*S. l.: s. n.*], 2022. https://www.gov.br/mec/pt-br/cne/pdf/resolucoes-do-cne/ceb/2022/rceb001_22.pdf. Institui normas sobre Computação na Educação Básica – Complemento à Base Nacional Comum Curricular (BNCC). Acesso em: 18 maio 2024.

BRASIL. **Resolução CNE/CP nº 2, de 22 de dezembro de 2017: institui e orienta a implantação da Base Nacional Comum Curricular**. [*S. l.: s. n.*], 2017. Ministério da Educação. Conselho Nacional de Educação. Disponível em: Portal da Base Nacional Comum Curricular. Acesso em: 7 abr. 2024.

BRAUN, Virginia; CLARKE, Victoria. **Thematic Analysis: A Practical Guide**. London: SAGE Publications, 2022. ISBN 9781473953246.

BRENNAN, Karen; RESNICK, Mitchel *et al.* Using artifact-based interviews to study the development of computational thinking in interactive media design. *In:* ANNUAL American Educational Research Association meeting, Vancouver, BC, Canada. [*S. l.: s. n.*], 2012. p. 1–25.

BURBEKOVA, Saule. Soft skills as the most in-demand skills of future IT specialists. *In:* IEEE. 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST). [*S. l.: s. n.*], 2021. p. 1–5.

CALDERON, Ivanilse; ORAN, Ana Carolina; FEITOSA, Eduardo; SILVA, Williamson. Um Survey sobre o Uso de Metodologias Ativas no Ensino de Programação em Universidades Brasileiras. *In:* SBC. SIMPÓSIO Brasileiro de Informática na Educação (SBIE). [*S. l.: s. n.*], 2024. p. 2163–2177.

CALDERON, Ivanilse; SILVA, Williamson; FEITOSA, Eduardo. Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study. **Informatics in Education**, Vilnius University Institute of Data Science e Digital Technologies, v. 23, n. 2, p. 279–322, 2024.

CALDERON, Ivanilse; SILVA, Williamson; FEITOSA, Eduardo. Um Mapeamento Sistemático da Literatura sobre o uso de Metodologias Ativas durante o Ensino de Programação no Brasil. **Simpósio Brasileiro de Informática na Educação (SBIE)**, SBC, p. 1152–1161, 2021.

CARVALHO, Franklhes; SOARES, Carlos. Avaliação por Rubricas seguindo a Taxonomia de Bloom no Diagnóstico de Aprendizagem em Algoritmos. *In:* ANAIS do XXXVI Simpósio Brasileiro de Informática na Educação. Curitiba/PR: SBC, 2025. p. 1109–1121. DOI: 10.5753/sbie.2025.12787. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/38490.

CHAIBATE, Hind; HADEK, Amine; AJANA, Souad; BAKKALI, Soumia; FARAJ, Kenza. A Comparative Study of the Engineering Soft Skills Required by Moroccan Job Market. **International Journal of Higher Education**, ERIC, v. 9, n. 1, p. 142–152, 2020.

CHANG, Lung-Chun; LIN, Hon-Ren; LIN, Jian-Wei. Learning motivation, outcomes, and anxiety in programming courses—A computational thinking–centered method. **Education and Information Technologies**, Springer, v. 29, n. 1, p. 545–569, 2024.

CHEAH, Chin Soon. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. **Contemporary Educational Technology**, Bastas, v. 12, n. 2, ep272, 2020.

CHENG, Michelle WT; LEE, Katherine KW; CHAN, Cecilia KY. Generic skills development in discipline-specific courses in higher education: A systematic literature review. **Curriculum and teaching**, James Nicholas Publishers, v. 33, n. 2, p. 47–65, 2018.

CHÉVEZ-CORONEL, Kevin; SARAGURO-BRAVO, Rodrigo; MAWYIN-KORIAKOVA, Margarita; RUGEL-DIAZ, Pamela; JARAMILLO-SALTOS, José. A comparative analysis of hard and soft skills required by Software Development Companies in Ecuador. *In:* IEEE. 2023 11th International Conference in Software Engineering Research and Innovation (CONISOFT). [*S. l.: s. n.*], 2023. p. 75–81.

CIANCARINI, Paolo; MISSIROLI, Marcello; RUSSO, Daniel. Exploiting agile practices to teach computational thinking. *In:* SPRINGER. SOFTWARE Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment:

Second International Workshop, DEVOPS 2019, Château de Villebrumier, France, May 6–8, 2019, Revised Selected Papers 2. [*S. l.: s. n.*], 2020. p. 63–83.

COELHO, Murilo; ARAÚJO, Allysson Allex; FREIRE, Sávio; PAIXAO, Matheus. Um estudo comparativo entre a visão de líderes e liderados sobre a importância de soft skills em desenvolvimento de software. *In:* SBC. WORKSHOP sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES). [*S. l.: s. n.*], 2024. p. 130–140.

CRESWELL, John W; CRESWELL, J David. **Research design: Qualitative, quantitative, and mixed methods approaches**. [*S. l.*]: Sage publications, 2017.

CRUZ, ...; SANTANA, ...; BITTENCOURT, ...; BARRETO, ...; GOMES, ...; SANTOS, ... [título completo do artigo Cruz et al. 2024]. **Revista Brasileira de Informática na Educação**, 2024.

CULCASI, Irene; VENEGAS, Rocío Paz Fontana. Service-Learning and soft skills in higher education: A systematic literature review. **Form@ re-Open Journal per la formazione in rete**, v. 23, n. 2, p. 24–43, 2023.

DADELO, Stanislav. Possibilities for evaluation to foster the soft skills of critical thinking, creativity, and communication in higher education. **Creativity Studies**, v. 18, n. 1, p. 13–29, 2025.

DELZANNO, Giorgio; GELATI, Luca; GUERRINI, Giovanna; SUGLIANO, Angela; TRAVERSARO, Daniele. Experience-based training in computer science education via online multiplayer games on computational thinking. *In:* SPRINGER. INTERNATIONAL Workshop on Higher Education Learning Methodologies and Technologies Online. [*S. l.: s. n.*], 2022. p. 459–470.

DENG, Wenbo; PI, Zhongling; LEI, Weina; ZHOU, Qingguo; ZHANG, Wenlan. Pencil Code improves learners' computational thinking and computer learning attitude. **Computer applications in engineering education**, Wiley Online Library, v. 28, n. 1, p. 90–104, 2020.

DESIDÉRIO, Sofia Bento; LELIS, Maria Rebecca L; RODRIGUES, Maria Elanne M; SANTOS, Francisca Kelen F dos; MARQUES, Anna Beatriz. Investigando o desenvolvimento de soft skills em projetos parceiros do Programa Meninas Digitais: um estudo exploratório. *In:* SBC. ANAIS do XVII Women in Information Technology. [*S. l.: s. n.*], 2023. p. 171–181.

DESROCHERS, Marcie; ZHANG, Jie; CARON, Stacey; STEINMILLER, Jenna. An experimental comparison of the effect of teacher versus self-evaluation/self-reflection feedback on college students' behavioral observation skills. **Journal of Behavioral Education**, Springer, v. 28, n. 2, p. 258–274, 2019.

EGGERT, Katia Monica Verdim; ASQUINO, Monica Aparecida; CRUZ, Dulce Márcia. Prática Pedagógica Construcionista com a Linguagem de Programação Scratch em uma abordagem STEAM. *In:* SBC. ANAIS do XXIX Workshop de Informática na Escola. [*S. l.: s. n.*], 2023. p. 158–168.

ESPINA-ROMERO, Lorena C; FRANCO, Sandra Lucia Aguirre; CONDE, Helga Ofelia Dworaczek; GUERRERO-ALCEDO, Jesús M; PARRA, Doile Enrique Ríos; RAMÍREZ, Juan Carlos Rave. Soft skills in personnel training: Report of publications in scopus, topics explored and future research agenda. **Heliyon**, Elsevier, v. 9, n. 4, 2023.

FANG, Jian-Wen; SHAO, Dan; HWANG, Gwo-Jen; CHANG, Shao-Chen. From critique to computational thinking: A peer-assessment-supported problem identification, flow definition, coding, and testing approach for computer programming instruction. **Journal of Educational Computing Research**, SAGE Publications Sage CA: Los Angeles, CA, v. 60, n. 5, p. 1301–1324, 2022.

FARIAS, Adriano Fiad; BARONE, Dante Augusto Couto. Computational thinking through an online game to develop soft and hard skills. *In:* IEEE. 2023 32nd Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE). [*S. l.: s. n.*], 2023. p. 1–6.

FERNÁNDEZ, E. A.; ROA MARTÍN, N. C. Educational Policy Framework to promote Computational Thinking towards STEAM in Public Schools in Boyacá - Colombia. *In:* 20TH LACCEI International Multi-Conference for Engineering, Education, and Technology: "Education, Research and Leadership in Post-pandemic Engineering: Resilient, Inclusive and Sustainable Actions". Boca Raton, Florida, USA: [*s. n.*], 2022.

FIGUEIREDO, José; GARCÍA-PEÑALVO, Francisco José. Design science research applied to difficulties of teaching and learning initial programming. **Universal Access in the Information Society**, Springer Science e Business Media Deutschland GmbH, 2022. ISSN 16155297. DOI: 10.1007/s10209-022-00941-4.

FIGUEIREDO, José; GARCÍA-PEÑALVO, Francisco José. Design science research applied to difficulties of teaching and learning initial programming. **Universal access in the information society**, Springer, v. 23, n. 3, p. 1151–1161, 2024.

FLICK, Uwe. **The SAGE handbook of qualitative data analysis**. [*S. l.*]: sage, 2013.

FOJCIK, Marcin; FOJCIK, Martyna Katarzyna; ZIEBINSKI, Adam; POLLEN, Bjarte; KAMPEN, Anne-Lena. The role of soft skills in engineering education. *In:* IEEE. 2023 IEEE Frontiers in Education Conference (FIE). [*S. l.: s. n.*], 2023. p. 1–9.

FRIESE, Susanne. Qualitative data analysis with ATLAS. ti. SAGE Publications Ltd, 2019.

GALSTER, Matthias; MITROVIC, Antonija; MALINEN, Sanna; HOLLAND, Jay. What soft skills does the software industry* really* want? An exploratory study of software positions in New Zealand. *In:* PROCEEDINGS of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. [*S. l.: s. n.*], 2022. p. 272–282.

GERALDES, Wendell Bento; AFONSECA, Ulisses Rodrigues. Estratégia pedagógica para permanência e êxito na disciplina de algoritmos e técnicas de programação do curso Técnico em Informática para Internet. *In:* SBC. SIMPÓSIO Brasileiro de Informática na Educação (SBIE). [*S. l.: s. n.*], 2024. p. 3097–3105.

GOLDONI, Diógines; REIS, Helena Macedo; JAQUES, Patricia. Computational tools to teach and develop socio-emotional skills: a systematic mapping. **International Journal of Learning Technology**, 2023. DOI: 10.1504/IJLT.2023.10058191.

GROVER, Shuchi; PEA, Roy. 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.

HECKMAN, James J; KAUTZ, Tim. Hard evidence on soft skills. **Labour economics**, Elsevier, v. 19, n. 4, p. 451–464, 2012.

HOLANDA, Wallace Duarte de; PAIVA FREIRE, Laís de; SILVA COUTINHO, Jarbele Cássia da. Estratégias de ensino-aprendizagem de programação introdutória no ensino superior: uma Revisão Sistemática da Literatura. **Revista Novas Tecnologias na Educação**, v. 17, n. 1, p. 527–536, 2019.

HSU, Ting-Chia; CHANG, Ching; WU, Long-Kai; LOOI, Chee-Kit. Effects of a pair programming educational robot-based approach on students' interdisciplinary learning of computational thinking and language learning. **Frontiers in psychology**, Frontiers Media SA, v. 13, p. 888215, 2022.

HU, Linlin. Programming and 21st century skill development in K-12 schools: A multidimensional meta-analysis. **Journal of Computer Assisted Learning**, Wiley Online Library, v. 40, n. 2, p. 610–636, 2024.

HUANG, Wendy; LOOI, Chee-Kit. A critical review of literature on "unplugged" pedagogies in K-12 computer science and computational thinking education. **Computer Science Education**, Taylor & Francis, v. 31, n. 1, p. 83–111, 2021.

HUBERMAN, A *et al.* Qualitative data analysis a methods sourcebook. Thousand Oaks, Califorinia SAGE Publications, Inc., 2014.

HUDIN, Salmiah Salleh. A Systematic Review of the Challenges in Teaching Programming for Primary Schools' Students. **Online Journal for TVET Practitioners**, v. 8, n. 1, p. 75–88, 2023.

HYURY, Andercley; MERCÊS, Sara; COQUEIRO, Thiago; RUIZ, Igor; CARVALHO, Tássio; JAILTON, José. Principais aplicações da computação desplugada no ensino fundamental ii: Um mapeamento sistemático da literatura. *In:* SBC. WORKSHOP de Informática na Escola (WIE). [*S. l.: s. n.*], 2024. p. 320–330.

ISMAIL, Dingot Hamonangan; NUGROHO, Joko; ROHAYATI, Teti. Literature Review: Soft Skill Needed by Gen Z in the Era RI 4.0 and Society 5.0. **Majalah Ilmiah Bijak**, v. 20, n. 1, p. 119–131, 2023.

JALINUS, Nizwardi; PUTRA, Rusnardi Rahmat *et al.* Implementation of Project-Based Learning Computational Thinking Models in Mobile Programming Courses. **International Journal of Interactive Mobile Technologies**, v. 18, n. 11, 2024.

KALLURI, Balaji; PRASAD, Prajish; SHARMA, Prakrati; CHIPPA, Divyaansh. Developing future computational thinking in foundational cs education: a case study from a liberal education University in India. **IEEE Transactions on Education**, IEEE, v. 67, n. 6, p. 944–953, 2024.

KAPUSTINA, Valeriia A; MATYUSHINA, Marina A. Theoretical Review of Digital Tools Aimed to Development Soft Skills during a Learning Process. *In:* IEEE. 2023 IEEE 24th International Conference of Young Professionals in Electron Devices and Materials (EDM). [*S. l.: s. n.*], 2023. p. 1920–1923.

KIM, Jiyoung; LEFTWICH, Anne; CASTNER, Daniel. Beyond teaching computational thinking: Exploring kindergarten teachers' computational thinking and computer science curriculum design considerations. **Education and Information Technologies**, Springer, p. 1–37, 2024.

KITCHENHAM, Barbara; CHARTERS, Stuart. Guidelines for performing systematic literature reviews in software engineering (EBSE 2007-001). **Keele University and Durham University Joint Report**, p. 1–53, 2007.

KITCHENHAM, Barbara A; PFLEEGER, Shari L. Personal opinion surveys. *In:* GUIDE to advanced empirical software engineering. [*S. l.*]: Springer, 2008. p. 63–92.

KNAFLIC, Cole Nussbaumer. **Storytelling com dados: um guia sobre visualização de dados para profissionais de negócios**. [*S. l.*]: Alta Books, 2019.

KUKUL, Volkan; ÇAKIR, Recep. Exploring the development of primary school students' computational thinking and 21st century skills through scaffolding: Voices

from the stakeholders. **International Journal of Computer Science Education in Schools**, v. 4, n. 2, p. 36–57, 2020.

LEE, Sang Joon; FRANCOM, Gregory M.; NUATOMUE, Jeremiah. Computer science education and K-12 students' computational thinking: A systematic review. **International Journal of Educational Research**, v. 114, p. 102008, 2022. ISSN 0883-0355. DOI: https://doi.org/10.1016/j.ijer.2022.102008. Disponível em: https://www.sciencedirect.com/science/article/pii/S0883035522000866.

LI, Feng; WANG, Xi; HE, Xiaona; CHENG, Liang; WANG, Yiyu. The effectiveness of unplugged activities and programming exercises in computational thinking education: A Meta-analysis. **Education and Information Technologies**, Springer, v. 27, n. 6, p. 7993–8013, 2022.

LIMA, Maurício; FERREIRA, Deller; DIAS, Elisângela. Uso de Rubricas em Disciplinas de Programação Introdutória: Uma Revisão Sistemática da Literatura. *In:* ANAIS do XXXV Simpósio Brasileiro de Informática na Educação. Rio de Janeiro/RJ: SBC, 2024. p. 1–14. DOI: 10.5753/sbie.2024.240991. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/31226.

LOKSA, Dastyni; MARGULIEUX, Lauren; BECKER, Brett A; CRAIG, Michelle; DENNY, Paul; PETTIT, Raymond; PRATHER, James. Metacognition and self-regulation in programming education: Theories and exemplars of use. **ACM Transactions on Computing Education (TOCE)**, ACM New York, NY, v. 22, n. 4, p. 1–31, 2022.

LÓPEZ-PERNAS, Sonsoles; SAQR, Mohammed; VIBERG, Olga. Putting it all together: Combining learning analytics methods and data sources to understand students' approaches to learning programming. **Sustainability**, MDPI, v. 13, n. 9, p. 4825, 2021.

MACRIDES, Elena; MILIOU, Ourania; ANGELI, Charoula. Programming in early childhood education: A systematic review. **International Journal of Child-Computer Interaction**, Elsevier, v. 32, p. 100396, 2022.

MADARIAGA, Leonardo; ALLENDES, Carolina; NUSSBAUM, Miguel; BARRIOS, Gustavo; ACEVEDO, Nicolás. Offline and online user experience of gamified robotics for introducing computational thinking: Comparing engagement, game mechanics and coding motivation. **Computers & Education**, Elsevier, v. 193, p. 104664, 2023.

MAITZ, Katharina; PALECZEK, Lisa; DANIELOWITZ, Claudia. Simultaneously Fostering Computational Thinking and Social-Emotional Competencies in 4th Graders Using Scratch: A Feasibility Study. *In:* PROCEEDINGS of Mensch und Computer 2022. [*S. l.: s. n.*], 2022. p. 399–403.

MANIKUTTY, Gayathri. My Robot can tell stories: Introducing robotics and physical computing to children using dynamic dioaramas. *In:* IEEE. 2021 IEEE Frontiers in Education Conference (FIE). [*S. l.: s. n.*], 2021. p. 1–9.

MATOS, [primeiro nome]; SILVA, [primeiro nome]; FARIAS, [primeiro nome]; ARAÚJO, [primeiro nome]; ARAÚJO, [primeiro nome]. Ensino do Pensamento Computacional como Estratégia para a Regulação Emocional. *In:* ANAIS do XXXII Simpósio Brasileiro de Informática na Educação (SBIE 2021). [cidade do evento]: Sociedade Brasileira de Computação (SBC), 2021. [páginas]. DOI: [DOIdaSBCOpenLib].

MATTAR, João; RAMOS, Daniela Karine. **Metodologia da pesquisa em educação: abordagens qualitativas, quantitativas e mistas**. [*S. l.*]: Edições 70, 2021.

MATTURRO, Gerardo; RASCHETTI, Florencia; FONTÁN, Carina. A Systematic mapping study on soft skills in software engineering. **J. Univers. Comput. Sci.**, v. 25, n. 1, p. 16–41, 2019.

MEDEIROS, Rodrigo; FALCÃO, Taciana; RAMALHO, Geber. Comparação entre o panorama internacional e nacional sobre o Ensino e a Aprendizagem de Introdução à Programação no Ensino Superior. *In:* ANAIS do XXIX Workshop sobre Educação em Computação. Evento Online: SBC, 2021. p. 478–487. DOI: 10.5753/wei.2021.15939.

MEDEIROS, Rodrigo; FALCÃO, Taciana; RAMALHO, Geber. Ensino e Aprendizagem de Introdução à Programação no Ensino Superior Brasileiro: Revisão Sistemática da Literatura. *In:* ANAIS do XXVIII Workshop sobre Educação em Computação. Porto Alegre, RS, Brasil: SBC, 2020. p. 186–190. DOI: 10.5753/wei.2020.11155.

MEDEIROS, Rodrigo Pessoa; RAMALHO, Geber Lisboa; FALCÃO, Taciana Pontual. A systematic literature review on teaching and learning introductory programming in higher education. **IEEE Transactions on Education**, IEEE, v. 62, n. 2, p. 77–90, 2018.

MIOTO, Fernanda; PETRI, Giani; WANGENHEIM, Christiane Gresse von; BORGATTO, Adriano Ferreti; PACHECO, Lúcia Helena Martins. bases21-um modelo para a autoavaliaçao de habilidades do século xxi no contexto do ensino de computaçao na educaçao básica. **Revista Brasileira de Informática na Educação**, v. 27, n. 01, p. 26, 2019.

MOHAMMED, FS; OZDAMLI, F. **A systematic literature review of soft skills in information technology education. Behavioral Sciences, 14 (10), 894**. [*S. l.: s. n.*], 2024.

MONTUORI, Chiara; GAMBAROTA, Filippo; ALTOÉ, Gianmarco; ARFÉ, Barbara. The cognitive effects of computational thinking: A systematic review and meta-analytic study. **Computers & Education**, Elsevier, v. 210, p. 104961, 2024.

MORAES, Rafael Peixoto de; COSTA, Valéria Franklin da; SCHOLZ, Ricardo EP. Mapeamento Sistemático do Ensino Introdutório de Programação nos Ensinos Técnico e Superior no Brasil. **Revista Brasileira de Informática na Educação**, v. 30, p. 628–647, 2022.

MORAIS, Ceres Germanna Braga; MENDES NETO, Francisco Milton; OSÓRIO, António José Meneses. Difficulties and challenges in the learning process of algorithms and programming in higher education: a systematic literature review. **Research, Society and Development**, v. 9, n. 10, e9429109287, 2020. DOI: 10.33448/rsd-v9i10.9287. Disponível em: https://rsdjournal.org/index.php/rsd/article/view/9287.

MOREIRA, Marco Antonio. Aprendizagem Significativa: da visão clássica à visão crítica (Meaningful learning: from the classical to the critical view). *In:* SN. CONFERÊNCIA de encerramento do V Encontro Internacional sobre Aprendizagem Significativa, Madrid, Espanha, setembro de. [*S. l.: s. n.*], 2006.

MOREIRA, Marco Antonio. Aprendizagem significativa: da visão clássica à visão crítica (Meaningful learning: from the classical to the critical view). *In:* CONFERÊNCIA de encerramento do V Encontro Internacional sobre Aprendizagem Significativa. Madrid, Espanha: [*s. n.*], set. 2006.

MOREIRA, Marco Antonio. O que é afinal aprendizagem significativa? **Revista cultural La Laguna Espanha**, 2012. Disponível em: urlhttp://moreira.if.ufrgs.br/oqueeafinal.pdf/ . Acesssado em 10/06/2019.

MOURA, Dante Henrique. A formação de docentes para a educação profissional e tecnológica. **Revista Brasileira da educação profissional e tecnológica**, v. 1, n. 1, p. 23–38, 2008.

NOURI, Jalal; ZHANG, Lechen; MANNILA, Linda; NORÉN, Eva. Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. **Education Inquiry**, Taylor & Francis, v. 11, n. 1, p. 1–17, 2020.

NOVAK, JD; GOWIN, DB. **Aprender a aprender. Lisboa: Plátano Edições Técnicas. Tradução de Learning how to learn.(1984)**. [*S. l.*]: Ithaca, NY: Cornell University Press, 1996.

NOVAK, Joseph D; RABAÇA, Ana; VALADARES, Jorge. **Aprender criar e utilizar o conhecimento: Mapas conceptuais TM como ferramentas de facilitação nas escolas e empresas**. [*S. l.: s. n.*], 2000.

NUDIN, Salamun Rohman; WARSITO, Budi; WIBOWO, Adi. Impact of soft skills competencies to predict graduates getting jobs using random forest algorithm. *In:*

IEEE. 2022 1st International Conference on Information System & Information Technology (ICISIT). [*S. l.: s. n.*], 2022. p. 49–54.

PACHECO, Eliezer. Perspectivas da educação profissional técnica de nível médio. **Proposta de diretrizes curriculares nacionais. São Paulo: Moderna**, 2012.

PAPERT, Seymour. Children, computers and powerful ideas. **New York: Basic Books**, v. 10, n. 1990, p. 1095592, 1990.

PAPERT, Seymour A. **Mindstorms: Children, computers, and powerful ideas**. [*S. l.*]: Basic books, 1980.

PARTNERSHIP FOR 21ST CENTURY LEARNING. **Framework for 21st Century Learning Definitions**. [*S. l.*]: Battelle for Kids, 2019. Acesso em: 8 jun. 2025. Disponível em: http://static.battelleforkids.org/documents/p21/P21_Framework_DefinitionsBFK.pdf.

PELIZZARI, Adriana; KRIEGL, Maria de Lurdes; BARON, Márcia Pirih; FINCK, Nelcy Teresinha Lubi; DOROCINSKI, Solange Inês. Teoria da aprendizagem significativa segundo Ausubel. **revista PEC**, Curitiba, v. 2, n. 1, p. 37–42, 2002.

PETERSEN, Kai; FELDT, Robert; MUJTABA, Shahid; MATTSSON, Michael. Systematic mapping studies in software engineering. *In:* BCS LEARNING & DEVELOPMENT. 12TH international conference on evaluation and assessment in software engineering (EASE). [*S. l.: s. n.*], 2008.

PRABAWATI, Putu Lely Somya; AGUSTIKA, Gusti Ngurah Sastra. Project-based learning based on STEM (Science, Technology, Engineering, And Mathematics) enhancing students science knowledge competence. **Jurnal Ilmiah Sekolah Dasar**, v. 4, n. 4, p. 621–629, 2020.

QIAN, Yizhou; LEHMAN, James. Students' misconceptions and other difficulties in introductory programming: A literature review. **ACM Transactions on Computing Education (TOCE)**, ACM New York, NY, USA, v. 18, n. 1, p. 1–24, 2017.

QUEIROZ, João Victor; RODRIGUES, Larissa Milena; COUTINHO, Jarbele C. Um relato dos fatores motivacionais na aprendizagem de programação na perspectiva de alunos iniciantes em programação da universidade federal rural do semi-Árido campus pau dos ferros-rn. **Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574)**, v. 1, n. 2, 2018.

RAMOS, Marise. Concepção do ensino médio integrado. **Texto apresentado em seminário promovido pela Secretaria de Educação do Estado do Pará nos dias**, v. 8, p. 1–26, 2008.

REGE, Antonio; SALGADO, Luciana; VITERBO, José. Pensamento Computacional e Soft Skills no Contexto do Ensino de Programação: Um Mapeamento Sistemático. *In:* ANAIS do XXXVI Simpósio Brasileiro de Informática na Educação. Curitiba/PR: SBC, 2025. p. 1390–1401. DOI: 10.5753/sbie.2025.12907. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/38510.

REHDER, Mads Middelboe; JENSEN, Jesper Juellund; MØLLER, Thilde Emilie; SCHRØDER, Vibeke. Exploring the potentials of unplugged activities: developing computational thinking in teacher education. **Nordic Journal of Comparative and International Education (NJCIE)**, v. 8, n. 4, 2024.

RESNICK, Mitchel. Distributed constructionism. Association for the Advancement of Computing in Education (AACE), 1996.

RESNICK, Mitchel; MALONEY, John; MONROY-HERNÁNDEZ, Andrés; RUSK, Natalie; EASTMOND, Evelyn; BRENNAN, Karen; MILLNER, Amon; ROSENBAUM, Eric; SILVER, Jay; SILVERMAN, Brian; KAFAI, Yasmin. Scratch: Programação para todos. **Communications of the ACM**, ACM, v. 52, n. 11, p. 60–67, 2009. DOI: 10.1145/1592761.1592779.

RIBEIRO, Natália; MARTINS, Lavínia; BERSSANETTE, João. Panorama Brasileiro do Ensino e Aprendizagem de Programação de Computadores na Educação Básica. *In:* ANAIS do XXX Workshop sobre Educação em Computação. Niterói: SBC, 2022. p. 346–356. DOI: 10.5753/wei.2022.222551. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/20843.

RIBEIRO, Ralph B. S.; CARVALHO, Leandro S. G.; OLIVEIRA, David B. F.; OLIVEIRA, Elaine H. T. de; PESSOA, Marcela. Investigação Empírica sobre os Efeitos da Gamificação de um Juiz Online em uma Disciplina de Introdução à Programação. **Revista Brasileira de Informática na Educação**, v. 28, p. 461–490, 2020. DOI: 10.5753/rbie.2020.28.0.461. Disponível em: https://journals-sol.sbc.org.br/index.php/rbie/article/view/3952.

ROCCO, Tonette S; PLAKHOTNIK, Maria S. Literature reviews, conceptual frameworks, and theoretical frameworks: Terms, functions, and distinctions. **Human resource development review**, SAGE Publications Sage CA: Los Angeles, CA, v. 8, n. 1, p. 120–130, 2009.

ROCHA, Filipa; CORREIA, Filipa; NETO, Isabel; PIRES, Ana Cristina; GUERREIRO, João; GUERREIRO, Tiago; NICOLAU, Hugo. Coding Together: On Co-located and Remote Collaboration between Children with Mixed-Visual Abilities. *In:* PROCEEDINGS of the 2023 CHI Conference on Human Factors in Computing Systems. [*S. l.: s. n.*], 2023. p. 1–14.

ROMÁN-GONZÁLEZ, Marcos; PÉREZ-GONZÁLEZ, Juan-Carlos; JIMÉNEZ-FERNÁNDEZ, Carmen. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. **Computers in human behavior**, Elsevier, v. 72, p. 678–691, 2017.

SANG JOON LEE GREGORY M. FRANCOM, Jeremiah Nuatomue. Play with coding toys in early childhood education and Care: Teachers' pedagogical Strategies, views and impact on children's development a systematic literature review. **Entertainment Computing**, Elsevier B.V., v. 50, 2024. DOI: 10.1016/j.entcom.2024.100637.

SANTANA, Bianca Leite; CHAVEZ, Christina von Flach Garcia; BITTENCOURT, Roberto Almeida. Uma definição operacional para pensamento computacional. *In:* SBC. SIMPÓSIO Brasileiro de Educação em Computação (EDUCOMP). [*S. l.: s. n.*], 2021. p. 93–103.

SANTOS, Jarles Gomes; SANTOS, Jaian. Primeiro contato com a programação através do Software Scratch: experiência no ensino técnico. *In:* 1. ANAIS do Workshop de Informática na Escola. [*S. l.: s. n.*], 2017. v. 23, p. 362–371.

SBC. **Grandes Desafios da Educação em Computação 2025-2035: Resumo Executivo**. Porto Alegre: Sociedade Brasileira de Computação (SBC), 2025. Disponível em: https://books-sol.sbc.org.br/index.php/sbc/catalog/book/163.

SCHWARTZ, Jeffrey L. Preparing high school students for college while training engineering students in "soft skills". *In:* IEEE. 2016 IEEE Integrated STEM Education Conference (ISEC). [*S. l.: s. n.*], 2016. p. 112–115.

SCHWARTZ, Lou; MAQUIL, Valérie; JOHANNSEN, Laurence; MOLL, Christian; HERMEN, Johannes. Teaching computational thinking with a tangible development platform: An exploratory field study at school with Kniwwelino. **Education and Information Technologies**, Springer, v. 29, n. 4, p. 4935–4967, 2024.

SHEKH-ABED, Aziz; BARAKAT, Nael. Exploring the correlation between systems thinking and soft skills for improved effectiveness of project based learning. *In:* IEEE. 2022 IEEE Frontiers in Education Conference (FIE). [*S. l.: s. n.*], 2022. p. 1–4.

SHUTE, Valerie J; SUN, Chen; ASBELL-CLARKE, Jodi. Demystifying computational thinking. **Educational research review**, Elsevier, v. 22, p. 142–158, 2017.

SILVA, Francisco Leocassio da; MOREIRA, Irlan Arley Targino. Análise das dificuldades na aprendizagem de programação no curso de análise e desenvolvimento de sistemas do IFRN/Pau dos Ferros. *In:* SBC. ENCONTRO Unificado de Computação do Piauí (ENUCOMPI). [*S. l.: s. n.*], 2021. p. 41–48.

SILVA, Leonardo S. Análise do aprendizado em programação de estudantes do ensino técnico integrado do Instituto Federal de Pernambuco. *In:* ANAIS do V Encontro Nacional de Computação dos Institutos Federais. Natal: SBC, 2018. DOI: 10.5753/encompif.2018.3559. Disponível em: https://sol.sbc.org.br/index.php/encompif/article/view/3559.

SILVA, Luciano Leite da; ALBUQUERQUE, Danyllo W. Identificaçao de Soft Skills a Partir da Avaliaçao de Anuncios de Vagas em Tecnologia da Informaçao. *In:* SBC. ANAIS do X Encontro Nacional de Computação dos Institutos Federais. [*S. l.: s. n.*], 2023. p. 5–8.

SILVA DEVINCENZI, Sam da; KWECKO, Viviani Rios; COSTA, Aléssio Almada da; LIMA BICHO, Alessandro de. Desenvolvimento de Soft Skills: Um Programa de Formação Universitária na Era da Capacitação 4.0. *In:* SBC. ANAIS do XXXIV Simpósio Brasileiro de Informática na Educação. [*S. l.: s. n.*], 2023. p. 1074–1085.

SIM, Tze Ying; LAU, Sian Lun. Review on challenges and solutions in novice programming education. *In:* IEEE. 2022 IEEE International Conference on Computing (ICOCO). [*S. l.: s. n.*], 2022. p. 55–61.

SIM, Tze Ying; TEOW, Matthew; LAU, Sian Lun. Unplugged computational thinking activities framework development for novice programmer. *In:* IEEE. 2021 IEEE International Conference on Computing (ICOCO). [*S. l.: s. n.*], 2021. p. 297–302.

TAYLOR, Sandra; MIN, Wookhee; MOTT, Bradford; EMERSON, Andrew; SMITH, Andy; WIEBE, Eric; LESTER, James. Position: IntelliBlox: A toolkit for integrating block-based programming into game-based learning environments. *In:* IEEE. 2019 IEEE blocks and beyond workshop (B&B). [*S. l.: s. n.*], 2019. p. 55–58.

THIOLLENT, Michel. **Metodologia da pesquisa-ação**. [*S. l.*]: Cortez editora, 2022.

TOMIĆ, Bojan; JOVANOVIĆ, Jelena; MILIKIĆ, Nikola; DEVEDŽIĆ, Vladan; DIMITRIJEVIĆ, Sonja; ĐURIĆ, Dragan; ŠEVARAC, Zoran. Grading students' programming and soft skills with open badges: A case study. **British Journal of Educational Technology**, Wiley Online Library, v. 50, n. 2, p. 518–530, 2019.

TRIPP, David. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, SciELO Brasil, v. 31, p. 443–466, 2005.

TSORTANIDOU, Xanthippi; DARADOUMIS, Thanasis; BARBERÁ, Elena. A K-6 computational thinking curricular framework: pedagogical implications for teaching practice. **Interactive Learning Environments**, Taylor & Francis, v. 31, n. 8, p. 4903–4923, 2023.

TSORTANIDOU, Xanthippi; DARADOUMIS, Thanasis; BARBERÁ, Elena. Connecting moments of creativity, computational thinking, collaboration and new media literacy skills. **Information and Learning Sciences**, Emerald Publishing Limited, v. 120, n. 11-12, p. 704–722, 2019.

VALENTE, José Armando. Por que o computador na educação. **Computadores e conhecimento: repensando a educação. Campinas: Unicamp/Nied**, p. 24–44, 1993.

VALLS POU, Albert; CANALETA, Xavi; FONSECA, David. Computational thinking and educational robotics integrated into project-based learning. **Sensors**, MDPI, v. 22, n. 10, p. 3746, 2022.

VASCONCELOS, Verónica; ALMEIDA, Ricardo; MARQUES, Luís; BIGOTTE, Emília. Scratch4All Project-Educate for an All-inclusive Digital Society. *In:* IEEE. 2023 32nd Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE). [*S. l.: s. n.*], 2023. p. 1–5.

VERSUTI, Fabiana Maris; MULLE, Rafael Lima Dalle; GUERREIRO, Carlos Antônio Rodrigues; MARTINS, Flavio Pinheiro; PERALTA, Deise Aparecida. Habilidades Socioemocionais e Tecnologias Educacionais: Revisão Sistemática de Literatura. **Revista Brasileira de Informática na Educação**, 2020.

VIEIRA, Alboni Marisa Dudeque Pianovski; SOUZA JÚNIOR, Antônio de. A educação profissional no Brasil. **Revista Interacções**, v. 12, n. 40, 2016.

VILLEGAS, C. **A systematic review of research on soft skills for employability. Advanced Education, 2024 (25), 200–212**. [*S. l.: s. n.*], 2024.

WANG, Xining; HERZOG, Samuel. A conceptual design for a learning platform enhancing children and youth's soft skills education. *In:* IEEE. 2023 3rd International Conference on Educational Technology (ICET). [*S. l.: s. n.*], 2023. p. 149–153.

WEF. **The Future of Jobs Report 2023**. [*S. l.*], 2023. Disponível em: https://www.weforum.org/reports/the-future-of-jobs-report-2023. Acessado em 13/11/2024.

WILSON, Thato Thamsanqa; MARNEWICK, AL. A comparative study of soft skills amongst the Washington accord engineering degree graduates with industry expectations. *In:* IEEE. 2018 IEEE international conference on engineering, technology and innovation (ICE/ITMC). [*S. l.: s. n.*], 2018. p. 1–6.

WING, Jeanette. Research notebook: Computational thinking—What and why. **The link magazine**, v. 6, p. 20–23, 2011.

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

WING, Jeannette M. **Computational Thinking Benefits Society**. [*S. l.: s. n.*], 2014. Disponível em: url http://socialissues.cs.toronto.edu/2014/01/computational-thinking/ . Acessado em 04/04/2024.

WONG, Gary Ka-Wai; CHEUNG, Ho-Yin. Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. **Interactive Learning Environments**, Taylor & Francis, v. 28, n. 4, p. 438–450, 2020.

WU, Ting-Ting; ASMARA, Andik; HUANG, Yueh-Min; PERMATA HAPSARI, Intan. Identification of Problem-Solving Techniques in Computational Thinking Studies: Systematic Literature Review. **SAGE Open**, SAGE Publications Sage CA: Los Angeles, CA, v. 14, n. 2, p. 21582440241249897, 2024.

YANG, Weipeng. Coding With Robots or Tablets? Effects of Technology-Enhanced Embodied Learning on Preschoolers' Computational Thinking and Social-Emotional Competence. **Journal of Educational Computing Research**, SAGE Publications Sage CA: Los Angeles, CA, p. 07356331241226459, 2024.

YONG, Su Ting; TIONG, Kung Ming. A blended Learning Approach: Motivation and difficulties in learning programming. **International Journal of Information and Communication Technology Education (IJICTE)**, IGI Global Scientific Publishing, v. 18, n. 1, p. 1–16, 2022.

YOUNIS, Awad A; SUNDERRAMAN, Rajshekhar; METZLER, Mike; BOURGEOIS, Anu G. Case study: Using project based learning to develop parallel programing and soft skills. *In:* IEEE. 2019 IEEE international parallel and distributed processing symposium workshops (IPDPSW). [*S. l.: s. n.*], 2019. p. 304–311.

YURCHENKO, A; DRUSHLYAK, M; KHVOROSTINA, Y; OSTROHA, M; PONOMARENKO, V; SEMENIKHINA, O. **The Impact of Team Competitions on the Development of Soft Skills in Youth. 2024 47th MIPRO ICT and Electronics Convention (MIPRO), 323–328**. [*S. l.: s. n.*], 2024.

YUSOFF, Karimah Mohd; ASHAARI, Noraidah Sahari; WOOK, TSMT; ALI, Noorazean Mohd. Analysis on the requirements of computational thinking skills to overcome the difficulties in learning programming. **International Journal of Advanced Computer Science and Applications**, Science e Information (SAI) Organization Limited, v. 11, n. 3, p. 244–253, 2020.

ZAMORA-HERNANDEZ, Israel; RODRIGUEZ-PAZ, Miguel X; GONZALEZ-MENDIVIL, Jorge A. An educational model for teaching PLC programming incorporating soft skills. *In:* PROCEEDINGS of the 15th International Conference on Education Technology and Computers. [*S. l.: s. n.*], 2023. p. 280–285.

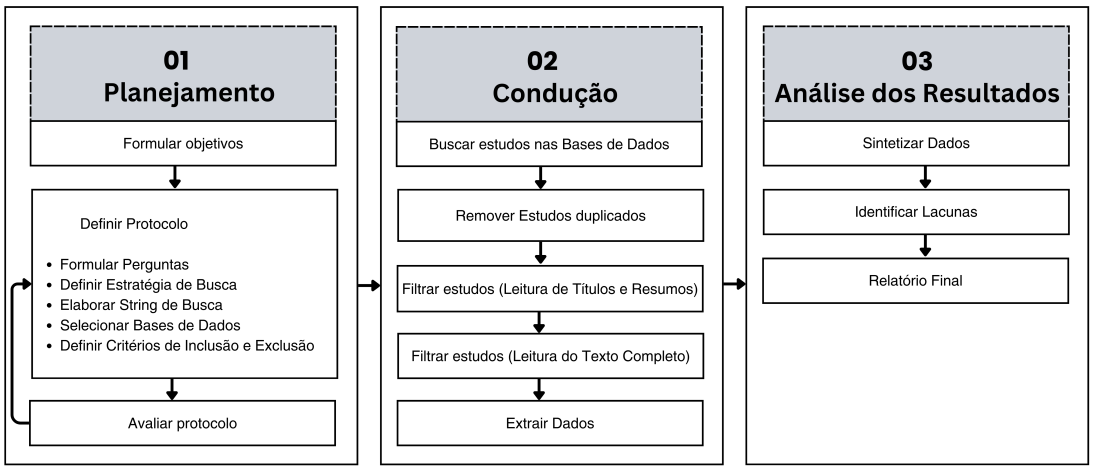
ZANETTI, Humberto; BORGES, Marcos; RICARTE, Ivan. A Teoria de Aprendizagem Significativa no Ensino de Programação: um Mapeamento Sistemático da Literatura. *In:* ANAIS do XXXIII Simpósio Brasileiro de Informática na Educação. Manaus: SBC, 2022. p. 01–14. DOI: 10.5753/sbie.2022.224579. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/22391.

APÊNDICE A – Protocolo do Mapeamento Sistemático da Literatura

Este apêndice apresenta o protocolo detalhado do MSL que fundamentou a construção do estado da arte do Capítulo 3. Parte do conteúdo metodológico anteriormente incluído nesse capítulo foi transferida para este apêndice, com o objetivo de tornar a discussão principal mais voltada à análise dos resultados.

A estratégia metodológica adotada para o MSL fundamenta-se nas diretrizes de Petersen *et al.* (2008) e Kitchenham e Charters (2007), amplamente utilizadas em pesquisas na área de computação. O MSL foi conduzido em três etapas principais: planejamento, condução e análise dos resultados, conforme mostra a Figura 44.

Figura 44: Etapas do MSL



Fonte: Autoria própria

A.1 Planejamento

No planejamento, definiram-se os objetivos e o protocolo da pesquisa, incluindo a formulação das perguntas, a estratégia de busca, o desenvolvimento da *string*, a seleção das

bases de dados e os critérios de inclusão e exclusão dos estudos. O *software* Parsifal [1] foi utilizado como suporte ao MSL por auxiliar na organização dos dados, na aplicação dos critérios definidos e na visualização dos resultados.

A.1.1 Questões de pesquisa

O MSL foi orientado por uma questão principal (QP), que define o escopo geral da revisão, e por cinco questões secundárias (PS1 a PS5), que detalham aspectos específicos da análise. A QP busca compreender como a literatura trata o desenvolvimento de PC e SS no ensino de programação, enquanto as questões secundárias investigam a distribuição temporal das publicações, os contextos educacionais, as abordagens e ferramentas utilizadas, as competências mais citadas e os desafios reportados pelos professores e estudantes. A Tabela 28 apresenta as questões e seus objetivos.

Tabela 28: Perguntas e objetivos do MSL

Perguntas	**Objetivos**
PP. De que maneira a literatura aborda o desenvolvimento do PC e das SS no ensino de programação?	Analisar como a literatura aborda o desenvolvimento do PC e das SS no ensino de programação, identificando tendências, enfoques teóricos e práticos.
PS1. Como se distribuem temporalmente as publicações sobre PC e SS no ensino de programação?	Caracterizar o volume das pesquisas no período.
PS2. Quais contextos educacionais (níveis de ensino) têm sido investigados?	Mapear os contextos em que ocorre o desenvolvimento de PC e SS.
PS3. Quais abordagens pedagógicas, ferramentas e fundamentos teóricos têm sido empregados?	Identificar as estratégias didáticas e recursos tecnológicos adotados.
PS4. Quais competências de PC e SS são mais frequentemente abordadas?	Mapear as competências centrais investigadas pelos estudos.
PS5. Quais desafios professores e estudantes enfrentam no desenvolvimento de PC e SS?	Identificar barreiras pedagógicas, técnicas e estruturais.

Fonte: Autoria própria.

A.1.2 Estratégia de busca

A construção da *string* de busca foi conduzida em duas etapas. Na primeira, definiram-se os blocos conceituais a partir da estrutura PICO (População, Intervenção, Comparação e Resultados), conforme mostra a Tabela 29.

[1] https://parsif.al/

Tabela 29: Estratégia PICO

PICO	Descrição
População	Professores e estudantes.
Intervenção	Abordagens, ferramentas e estratégias voltadas ao desenvolvimento de PC e SS no ensino de programação.
Comparação	Não aplicável (estudo de mapeamento sistemático, não comparativo).
Resultados	Resultados sobre o desenvolvimento de competências de PC e SS no ensino de programação.

Fonte: Autoria própria.

Na segunda, realizou-se um levantamento exploratório de revisões sistemáticas e mapeamentos publicados sobre os temas centrais da pesquisa (PC, ensino de programação e habilidades não técnicas correlatas às SS), com o objetivo de identificar termos recorrentes na literatura e suas variações morfológicas. A Tabela 30 apresenta os estudos consultados nessa fase exploratória, que serviram de base para o refinamento do vocabulário da *string*.

Tabela 30: Estudos de referência consultados na construção da *string*

Autor(es) e Ano	Local de publicação	Contribuição para o refinamento da string
(Cruz *et al.*, 2024)	RBIE	Revisão sobre o efeito da promoção do PC nas Habilidades do Século XXI, contemplando os eixos *computational thinking*, *programming/coding* e *21st century skills/social skills*.
(Espina-Romero *et al.*, 2023)	*Heliyon*	Revisão sobre publicações em *soft skills* indexadas em Scopus, contemplando a denominação *soft skills* em contextos amplos de formação.
(Goldoni; Reis; Jaques, 2023)	*International Journal of Learning Technology*	Mapeamento sistemático sobre ferramentas computacionais para ensinar e desenvolver *socio-emotional skills*, contemplando o eixo *social-emotional competence/skills*.
(Versuti *et al.*, 2020)	RBIE	Revisão sobre habilidades socioemocionais e tecnologias educacionais, contemplando a aproximação entre tecnologias e o leque de habilidades não técnicas em contexto brasileiro.

Fonte: Autoria própria.

A.1.3 String de busca

Com base nas diretrizes estabelecidas, a *string* de busca foi construída com operadores *booleanos* AND e OR e o uso de curingas como "*". A base Scopus foi utilizada nos ajustes iniciais, realizados por meio de testes-piloto e refinamentos sucessivos para melhorar a precisão e a relevância dos estudos recuperados. Durante esse processo, termos sem resultados relevantes foram removidos. Após sucessivas revisões e validação por especialista de educação em computação, definiu-se a *string* final apresentada na Tabela 31.

Tabela 31: *String* de busca

("computational thinking"AND ("programming"OR "coding")) AND ("soft skill"OR "social skill*"OR "non-cognitive skill*"OR "personal skill*"OR "essential skill*"OR "interpersonal skill*"OR "social-emotional competenc*"OR "21st century skill*"OR "21st century competenc*")*

Fonte: Autoria própria.

A.1.4 Base de dados

Foram consultadas seis bases de dados, incluindo cinco internacionais e uma nacional, conforme a Tabela 32.

Tabela 32: Bases de dados utilizadas no mapeamento

Base de Dados	**Abrangência**	**Justificativa**
ACM Digital Library	Internacional	Cobertura em ciência da computação e educação em computação.
IEEE Digital Library	Internacional	Referência em engenharia e tecnologia, com presença em educação em computação.
ERIC	Internacional	Base especializada em educação, mantida pelo Departamento de Educação dos EUA.
ISI Web of Science	Internacional	Indexação multidisciplinar de alta qualidade.
Scopus	Internacional	Cobertura abrangente e multidisciplinar.
SOL (SBC)	Nacional	Cobertura da produção brasileira em Informática na Educação.

Fonte: Autoria própria.

A.1.5 Critérios de inclusão e exclusão

Os critérios de inclusão (CI) e exclusão (CE) foram definidos para garantir a aderência dos estudos selecionados aos objetivos do MSL.

Critérios de Exclusão

1. Estudos com 3 páginas ou menos (short papers).
2. Estudos com textos completos indisponíveis para *download*.
3. Estudos secundários.
4. Estudos que não estejam escritos em inglês ou português.
5. Estudos duplicados (somente uma cópia de cada estudo foi incluída).
6. Estudos que não sejam relevantes para as questões de pesquisa.

7. Estudos de literatura cinza, resumos de conferência, relatórios técnicos, livros ou capítulos de livro.

A.1.6 Justificativa do recorte temporal (2020 a 2024)

A delimitação temporal de 2020 a 2024 foi definida com base em marcos normativos brasileiros que reorganizaram a Educação Básica. A Resolução CNE/CP nº 2/2017 instituiu a BNCC e estabeleceu 2020 como prazo para adequação curricular das redes de ensino. A BNCC incorpora competências cognitivas e socioemocionais e, posteriormente, o Parecer CNE/CEB nº 2/2022 e a Resolução CNE/CEB nº 1/2022 institucionalizaram a Computação na Educação Básica brasileira. Assim, o recorte acompanha o período de vigência obrigatória da BNCC e da consolidação da Computação na Educação Básica no país.

O intervalo de cinco anos também é recorrente em revisões e mapeamentos sobre PC e ensino de programação, como em (Lee; Francom; Nuatomue, 2022; Sang Joon Lee Gregory M. Francom, 2024; Matos *et al.*, 2021). Metodologicamente, esse recorte permitiu concentrar a análise em produções recentes sem comprometer a viabilidade do MSL no contexto da tese.

Reconhece-se que estudos anteriores ao período delimitado permanecem como referências teóricas relevantes e foram utilizados no Capítulo 2 como base conceitual da pesquisa. O recorte temporal aplica-se exclusivamente aos estudos primários analisados no MSL.

A.2 Condução

Na etapa de condução, foram realizadas buscas automatizadas nas bases selecionadas utilizando a *string* previamente definida. As consultas foram adaptadas às características de cada base, considerando busca em texto completo ou em título, resumo e palavras-chave. A Tabela 33 apresenta os detalhes da condução das buscas em cada base.

Tabela 33: Aplicação da *string* de busca nas bases de dados

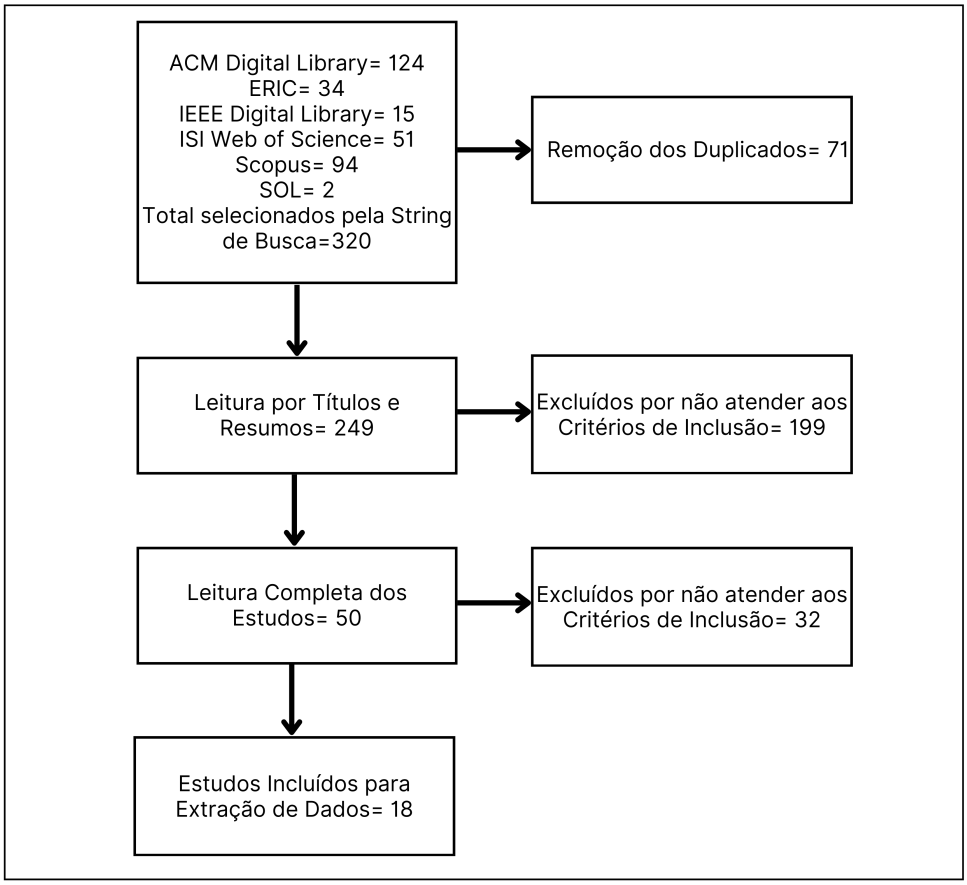
Base de Dados	**Modo de Aplicação da String**	**Data da última Busca**
ACM Digital Library	Texto Completo	12 de outubro de 2024
IEEE Digital Library	Texto Completo	13 de outubro de 2024
ERIC	Título, Resumo e Palavras-Chave	14 de outubro
ISI Web of Science	Título, Resumo e Palavras-Chave	13 de outubro de 2024
Scopus	Título, Resumo e Palavras-Chave	14 de outubro
SOL	Texto Completo	12 de outubro de 2024

Fonte: Autoria própria.

A.2.1 Resultados das buscas

Após a obtenção dos 320 estudos, foi executado o processo de seleção em três etapas: (1) remoção de duplicatas e triagem inicial, restando 249 estudos; (2) leitura de título e resumo, resultando em 50 estudos para leitura completa; e (3) leitura integral, com a seleção final de 18 estudos primários para extração de dados. A Figura 45 apresenta esse processo.

Figura 45: Processo de seleção dos estudos



Fonte: Autoria Própria.

A.3 Síntese dos resultados

O MSL foi conduzido em três etapas, planejamento, condução e análise dos resultados. As buscas realizadas entre 12 e 14 de outubro de 2024 nas bases ACM Digital Library, IEEE Digital Library, ERIC, ISI Web of Science, Scopus e SOL resultaram em 320 estudos, dos quais 18 foram selecionados após as etapas de triagem. A extração dos dados mapeou contextos educacionais, abordagens pedagógicas, ferramentas, competências de PC e SS e desafios reportados. A análise desses resultados é apresentada no Capítulo 3 desta tese.

APÊNDICE B – Matriz de Extração dos Estudos

Este apêndice apresenta a matriz de extração dos dados dos 18 estudos selecionados no MSL descrito no Apêndice A. A matriz organiza, para cada estudo, as informações relativas às questões de pesquisa: nível de ensino investigado (PS2), abordagens pedagógicas e ferramentas tecnológicas adotadas (PS3), componentes do PC e SS mais abordados (PS4) e principais desafios reportados (PS5). A primeira questão (PS1), relativa à distribuição temporal das publicações, é tratada de forma agregada no Capítulo 3 e, por isso, não compõe esta matriz. Os dados consolidados foram obtidos a partir do processo de extração registrado no *software* Parsifal. A Tabela 34 apresenta a matriz de extração dos dados dos 18 estudos selecionados no MSL.

Tabela 34: Matriz de extração dos 18 estudos

ID	Nível (PS2)	Abordagem (PS3)	Ferramenta (PS3)	PC (PS4)	SS (PS4)	Desafios (PS5)
E01	Fundamental	Scaffolding	KoduGameLab	Decomposição, abstração	Criatividade, colaboração, resolução de problemas	Falta de interesse dos alunos, dificuldade de abstração e influência da experiência do professor
E02	Fundamental	Robótica educacional	Scratch, robótica	Reconhecimento de padrões, decomposição, algoritmos	Colaboração, comunicação, criatividade, persistência, paciência	Falta de conhecimento dos professores e suporte limitado a alunos com dificuldades
E03	Fundamental	Jogos, Construcionismo	Scratch, KoduGameLab	Algoritmos, decomposição	Colaboração, comunicação, resolução de problemas	Transferência limitada do PC e SS para outros contextos
E04	Médio, Superior	Aprendizagem autodirigida, programação em pares, metodologias ágeis	Programação individual e em pares	Decomposição, abstração	Trabalho em equipe, comunicação, colaboração	Equilibrar habilidades técnicas e sociais, retenção limitada e avaliação individual em grupo
E05	Fundamental	Storytelling, Construcionismo	Robótica, Scratch	Algoritmos, abstração	Colaboração, criatividade, comunicação	Falta de infraestrutura tecnológica adequada
E06	Fundamental	Aprendizagem baseada em projetos	Scratch, robótica	Paralelismo, abstração, algoritmos	Trabalho em equipe, resolução de problemas, comunicação	Complexidade da programação e adaptação à dinâmica de cooperação em grupo
E07	Fundamental	Programação em pares, jogos	Robótica educacional	Decomposição, algoritmos	Cooperação, comunicação	Criar ambiente que incentive a interação e reduza ansiedade no aprendizado
E08	Fundamental	PBL, ABP, jogos	Scratch, Python	Decomposição, abstração, design de algoritmos, avaliação	Comunicação, colaboração, flexibilidade, pensamento crítico	Falta de formação docente, escassez de recursos e dificuldade de integração curricular
E09	Fundamental	Programação em pares	Scratch	Decomposição, abstração, reconhecimento de padrões	Colaboração, empatia, expressão de sentimentos	Dificuldades técnicas iniciais no uso do Scratch
E10	Fundamental	Aprendizagem experiencial, jogos	Jogo multiplayer online	Algoritmos	Trabalho em equipe, colaboração	Elaboração de desafios apropriados e adaptação ao ambiente online
E11	Superior	Aprendizagem baseada em projetos	Robótica educacional	Decomposição, abstração, algoritmos	Criatividade, colaboração	Engajamento em robótica e compreensão de conceitos complexos por iniciantes
E12	Fundamental	Robótica, jogos	Robótica educacional	Decomposição, algoritmos, depuração	Comunicação, colaboração, resolução de conflitos	Falta de ferramentas acessíveis e coordenação em ambientes remotos

Continua na próxima página.

Tabela 34: Matriz consolidada de extração dos 18 estudos (continuação)

ID	Nível (PS2)	Abordagem (PS3)	Ferramenta (PS3)	PC (PS4)	SS (PS4)	Desafios (PS5)
E13	Fundamental	Programação em pares	Scratch	Abstração, algoritmos, decomposição	Trabalho em equipe, comunicação	Diferenças de habilidade entre pares e desengajamento em tarefas individuais
E14	Fundamental, Médio	Plataforma tangível	Kniwwelino	Abstração, decomposição, algoritmos, depuração, generalização	Comunicação, colaboração	Frustração inicial pela falta de feedback imediato e adaptação dos materiais
E15	Fundamental	Aprendizagem baseada em projetos	Scratch, robótica	Algoritmos	Criatividade, colaboração, comunicação	Dificuldade no trabalho colaborativo e falta de recursos humanos e materiais
E16	Infantil	Robótica educacional	Robótica, ScratchJr	Decomposição, reconhecimento de padrões, depuração, algoritmos	Comunicação, colaboração, regulação emocional	Regulação emocional inicial e adaptação a diferentes modalidades de interação
E17	Superior	Aprendizagem baseada em projetos	Programação móvel	Abstração, algoritmos, decomposição	Criatividade, comunicação, colaboração, pensamento crítico	Integração de projetos e PC pelos professores e manutenção do engajamento
E18	Superior	Aprendizagem experiencial, programming em pares	Programação específica	Abstração, decomposição, algoritmos	Criatividade, responsabilidade, colaboração	Não reportado pelo estudo

Fonte: Autoria própria.

A organização dos dados em formato matricial favorece a leitura comparativa entre os estudos e torna mais visíveis os padrões e lacunas da produção científica analisada. Observa-se concentração de componentes do PC, como decomposição, algoritmos e abstração, e de competências interpessoais, como colaboração, comunicação e trabalho em equipe. Esses elementos aparecem com frequência nos estudos analisados. Entre os desafios, destacam-se as limitações relacionadas à formação docente, à integração curricular e à combinação entre componentes do PC e SS, aspectos relevantes para a fundamentação do *framework* proposto neste trabalho.

APÊNDICE C – Instrumentos do Estudo Empírico com os Professores dos Institutos Federais

Questionário – Percepção dos Professores sobre os Fatores que Influenciam o Ensino de Programação

Parte 1: Perfil do Respondente

1. Qual é o seu sexo?
2. Qual é a modalidade do seu curso de graduação?

   ( ) Licenciatura

   ( ) Bacharelado

   ( ) Tecnólogo

3. Qual é a sua formação acadêmica mais alta?

   ( ) Graduação

   ( ) Especialização

   ( ) Mestrado

   ( ) Doutorado

   ( ) Pós-doutorado

4. Em qual estado está localizado o Instituto Federal onde você trabalha? (Lista dos estados brasileiros)
5. Há quanto tempo você é professor no Instituto Federal?

( ) Menos de 3 anos
( ) 3 a 6 anos
( ) 7 a 10 anos
( ) Mais de 10 anos

6. Em qual modalidade de ensino você atua como docente em curso de computação?

   ( ) Técnico Integrado
   ( ) Ensino Superior
   ( ) Ambos

7. Em quais áreas você ministra disciplinas no IF? (Marque todas as opções que se aplicam.)

   [ ] Desenvolvimento de Software
   [ ] Redes de Computadores
   [ ] Inteligência Artificial
   [ ] Design e Experiência do Usuário
   [ ] Banco de Dados
   [ ] Engenharia de Software
   [ ] Robótica
   [ ] Segurança da Informação
   [ ] Informática Básica
   [ ] Lógica e Algoritmos
   [ ] Outras: _____

Parte 2: Percepções e Desafios no Ensino de Programação

Para cada afirmativa abaixo, marque um "X" na opção que melhor reflete sua resposta. Os valores variam de discordo totalmente (1) a concordo totalmente (5).

8. O laboratório de informática e a internet são adequados para o ensino de programação.
9. Os alunos conseguem desenvolver abstração, algoritmos, reconhecimento de padrões e decomposição de problemas de forma satisfatória.
10. Os alunos fazem perguntas e participam de discussões durante as aulas.
11. A matemática básica não influencia o aprendizado em programação.
12. Os alunos demonstram engajamento nas atividades de programação.
13. O tempo disponível para ensinar programação é suficiente.
14. Metodologias ativas podem favorecer o ensino e a aprendizagem de programação.

15. As atividades das disciplinas de programação causam sobrecarga nos alunos.

16. A colaboração, comunicação e outras habilidades interpessoais entre os alunos têm influência na aprendizagem de programação.

17. Os alunos demonstram confiança em seguir carreira acadêmica ou profissional na área de programação.

Parte 3: Considerações Finais

18. Gostaria de adicionar alguma outra observação ou sugestão que considere relevante para este estudo?

Roteiro da Entrevista – Percepção dos Professores sobre o Uso de Pensamento Computacional e Soft Skills no Ensino de Programação

Percepção sobre Soft Skills no Ensino de Programação

1. Você conhece o conceito de *soft skills*? Se sim, como você o definiria?

2. Na sua opinião, as *soft skills* têm algum impacto no ensino de programação? Por quê?

3. Quais *soft skills* você considera mais relacionadas ao ensino de programação? Selecione até cinco habilidades na lista abaixo e, se desejar, justifique sua escolha com exemplos práticos.

   - Comunicação
   - Trabalho em Equipe
   - Colaboração
   - Adaptabilidade
   - Resolução de Problemas
   - Pensamento Crítico
   - Criatividade
   - Gestão do Tempo
   - Empatia
   - Proatividade

- Inteligência Emocional
- Liderança
- Persistência
- Curiosidade
- Organização
- Habilidade de Pesquisa
- Responsabilidade
- Outras

4. Você utiliza estratégias específicas para desenvolver *soft skills* em suas aulas? Caso utilize, poderia descrever se essas estratégias são aplicadas de forma mais informal (ad hoc) ou planejada (sistematizada)?

5. Você encontra dificuldades ao incorporar *soft skills* no ensino de programação? Caso sim, quais? Caso não, pode explicar por quê?

Percepção sobre Componentes de Pensamento Computacional

6. Você conhece o conceito de pensamento computacional? Caso sim, como você o definiria no contexto do ensino de programação?

7. Você já utilizou componentes de pensamento computacional, como decomposição, abstração, algoritmos ou reconhecimento de padrões, no ensino de programação? Caso sim, esse uso foi de forma *ad hoc* (informal) ou sistematizada (planejada)? Poderia compartilhar brevemente sua experiência?

8. Como você percebe o uso dos componentes de pensamento computacional no ensino de programação? Algum deles se destaca como mais desafiador ou mais presente? Por quê?

Sugestões para o Framework

9. Você percebe alguma relação entre o desenvolvimento de pensamento computacional e *soft skills* no ensino de programação? Caso sim, como essas dimensões podem se complementar?

10. Se você fosse aplicar um *framework* no ensino de programação, como organizaria sua implementação? Pode descrever sua preferência em termos de tempo e estrutura?

11. Um treinamento prático ou uma instrução teórica sobre o funcionamento do *framework* seria importante para sua aplicação no ensino de programação? Ambos seriam necessários? Por quê?

12. Caso uma instrução teórica ou treinamento sejam considerados importantes, qual formato você acredita ser mais adequado para oferecê-los?

13. Quais recursos pedagógicos, abordagens metodológicas e ferramentas você considera mais relevantes para apoiar a aplicação de um *framework* no ensino de programação?

Considerações Finais

14. Gostaria de adicionar alguma outra observação ou sugestão que considere relevante para este estudo?

APÊNDICE D – Instrumento de apreciação do SoPensa por especialistas

Identificação do especialista

Código: _____ Formação: _____

Tempo de experiência em Lógica de Programação: _____ Data: _____

Instruções: para cada critério, indique o grau de atendimento (A = Atendido; PA = Parcialmente atendido; NA = Não atendido) e registre as observações e recomendações pertinentes.

Tabela 35: Instrumento de apreciação do *framework* SoPensa por especialistas

ID	Dimensão	Critério apreciado	A	PA	NA	Observações / Recomendações
1	Nomenclatura	Consistência dos termos utilizados no framework				
2	Mediação docente	Adequação do nome e da função atribuída à Fase 3				
3	Retroalimentação	Clareza do retorno à fase correspondente à lacuna identificada				
4	Organização das atividades	Adequação das atividades à função de cada fase				
5	Representação do processo	Clareza e sequência lógica do fluxo de implementação				
6	Decisão final	Clareza do ponto de decisão após a Fase 4				
7	Autoavaliação	Adequação dos momentos destinados à autoavaliação				
8	Progressão pedagógica	Organização gradual das atividades				
9	Instrumentos de acompanhamento	Quantidade, pertinência e viabilidade dos instrumentos				
10	Alinhamento curricular	Correspondência das atividades com a ementa da disciplina				

Fonte: Autoria própria.

Comentários gerais (clareza conceitual, coerência pedagógica e viabilidade prática):

_____

_____

_____

APÊNDICE E – Instrumentos de Diagnóstico e Acompanhamento Docente do SoPensa

Questionário diagnóstico sobre os conhecimentos prévios de PC e SS

Objetivo: identificar os perfis prévios de Pensamento Computacional e *Soft Skills* dos alunos antes da aplicação do *framework*.

Orientações de uso: leia cada pergunta e marque a opção que mais se aproxima da sua forma de agir. Não há resposta certa ou errada.

1. Qual o seu curso?

( ) Informática para Internet ( ) Redes de Computadores

2. Já teve contato com programação?

( ) Sim ( ) Não

Se sim, quais linguagens ou ferramentas utilizou? _____

Sobre sua abordagem para resolver problemas

3. Como você costuma começar uma tarefa mais complexa, como criar um jogo ou aplicativo?

( ) Faço um plano com as etapas principais antes de começar.

( ) Começo direto e vou ajustando ao longo do processo.

( ) Procuro exemplos prontos para copiar e adaptar.

( ) Outro. Explique: _____

4. Quando encontra uma tarefa parecida com algo que já fez antes, o que costuma fazer?

( ) Uso a ideia anterior e faço as adaptações necessárias.
( ) Faço do zero, com uma nova abordagem.
( ) Testo soluções diferentes até funcionar.
( ) Outro. Explique: _____

5. Se precisasse criar um sistema simples para uma biblioteca, por onde começaria?

( ) Focaria nas funcionalidades principais, como cadastro de livros e empréstimos, e deixaria de lado os detalhes extras.
( ) Tentaria incluir todas as funcionalidades possíveis logo de início.
( ) Escreveria um código simples e iria ajustando enquanto testasse.
( ) Outro. Explique: _____

6. Se tivesse que ensinar alguém a fazer algo (por exemplo, fazer *login* no computador), como faria?

( ) Criaria um passo a passo bem organizado e explicativo.
( ) Daria instruções gerais e deixaria a pessoa tentar.
( ) Mostraria como se faz e deixaria a pessoa copiar.
( ) Outro. Explique: _____

7. Quando participa de atividades em grupo, você costuma...

( ) Colaborar nas decisões e nas tarefas.
( ) Trabalhar sozinho, mas aceitar ajuda quando preciso.
( ) Resolver as coisas sozinho, pois não gosto de trabalho em grupo.
( ) Outro. Explique: _____

8. Quando precisa explicar algo para outras pessoas, como se sente?

( ) Consigo explicar minhas ideias de forma clara.
( ) Tenho um pouco de dificuldade, mas tento explicar.
( ) Evito falar em público e prefiro apenas executar a tarefa.
( ) Outro. Explique: _____

9. Quando surge uma tarefa nova, como costuma lidar com a solução?

( ) Gosto de pensar em ideias diferentes para resolver.
( ) Prefiro usar exemplos que já funcionaram antes.
( ) Costumo seguir as instruções que me foram dadas.
( ) Outro. Explique: _____

10. Quando alguém apresenta uma solução pronta para você...

( ) Penso se a solução faz sentido antes de usar.

( ) Sigo a sugestão, mas reflito se poderia melhorar.

( ) Uso a solução diretamente, se estiver clara.

( ) Outro. Explique: _____

11. Caso queira, deixe seu comentário ou sugestão sobre suas expectativas para o curso.

_____

_____

Rubrica de Acompanhamento dos Times

Objetivo: apoiar o professor na observação das competências de Pensamento Computacional e Soft Skills de cada time durante a resolução colaborativa de problemas de programação.

Orientações de uso: assinale com um X o parêntese do nível observado em cada critério. Os níveis funcionam como referência descritiva, e o desenvolvimento do time é acompanhado pela comparação dos níveis entre as fases, sem soma de pontos.

(4) Excelente: domínio consistente da competência.

(3) Bom: desempenho satisfatório, com pequenas limitações.

(2) Regular: desempenho mediano, com lacunas perceptíveis.

(1) Precisa Melhorar: dificuldades ou realizações incompletas.

Identificação:

Curso: _____ Time: _____ Data: _____/_____/_____

Tabela 36: Rubrica de acompanhamento de PC e SS

Critério	**Excelente (4)**	**Bom (3)**	**Regular (2)**	**Precisa Melhorar (1)**
PC: Decomposição	( ) Identificou claramente entradas, regras e saídas.	( ) Identificou a maior parte, com pequenas imprecisões.	( ) Identificou parcialmente.	( ) Identificação incompleta ou incorreta.
PC: Reconhecimento de Padrões	( ) Relacionou com problemas anteriores e aplicou padrões.	( ) Relacionou e aplicou padrões, com pequenos apoios.	( ) Relacionou parcialmente.	( ) Não identificou padrões ou conexões.
PC: Abstração	( ) Focou no essencial e descartou o que não era relevante.	( ) Concentrou-se no essencial, com poucos desvios.	( ) Abstração parcial ou confusa.	( ) Incapaz de diferenciar o essencial do irrelevante.
PC: Algoritmos	( ) Passo a passo claro, funcional e bem estruturado.	( ) Passo a passo funcional, com pequenos ajustes.	( ) Lógica básica ou incompleta.	( ) Algoritmo confuso ou ausente.
SS: Colaboração	( ) Trabalho bem dividido, todos contribuíram.	( ) Boa divisão, com pequena diferença de participação.	( ) Contribuição parcial de alguns membros.	( ) Apenas um ou dois fizeram a maior parte.
SS: Comunicação	( ) Troca de ideias clara, ativa e respeitosa.	( ) Boa comunicação, com pontos a melhorar.	( ) Comunicação parcial ou desigual entre os membros.	( ) Faltou diálogo ou houve confusão constante.
SS: Criatividade	( ) Propôs soluções diferentes e inovadoras.	( ) Propôs soluções próprias, com variação moderada.	( ) Soluções básicas, com pouca variação.	( ) Não tentou alternativas criativas.
SS: Pensamento Crítico	( ) Questionou e testou hipóteses de forma eficaz.	( ) Questionou e testou, com apoio pontual.	( ) Alguma análise crítica, ainda superficial.	( ) Aceitou respostas sem testar ou questionar.

Fonte: Autoria própria.

Observações do professor(a):

_____

_____

_____

Autoavaliação do Time

Objetivo: ajudar o time a refletir sobre como trabalhou nas atividades, reconhecendo o que foi feito bem e o que pode melhorar.

Orientações de uso: marque um X na opção que melhor representa o trabalho do time em cada item.

Identificação:

Time: _____ Curso: _____ Data: _____/_____/_____
Tipo de atividade: [ ] Desplugada _____ [ ] Plugada _____

1. Pensamento Computacional

Decomposição: conseguimos identificar todas as partes do problema (entradas, regras, saídas)?
( ) Sim ( ) Mais ou menos ( ) Não

Reconhecimento de padrões: conseguimos ver algo parecido com o que já fizemos?
( ) Sim ( ) Mais ou menos ( ) Não

Abstração: ignoramos as partes que não eram úteis para o código?
( ) Sim ( ) Mais ou menos ( ) Não

Algoritmos: o passo a passo do nosso programa estava claro e organizado?
( ) Sim ( ) Mais ou menos ( ) Não

2. Soft Skills

Colaboração: todos participaram de forma equilibrada?
( ) Sim ( ) Mais ou menos ( ) Não

Comunicação: conseguimos explicar e escutar bem as ideias do time?
( ) Sim ( ) Mais ou menos ( ) Não

Criatividade: tivemos ideias diferentes ou saímos do óbvio?
( ) Sim ( ) Mais ou menos ( ) Não

Pensamento crítico: testamos a lógica e questionamos se funcionaria em outras situações?
( ) Sim ( ) Mais ou menos ( ) Não

3. O que fizemos bem?

_____

4. O que podemos melhorar na próxima vez?

_____

Ficha de Observação das Fases do SoPensa

Objetivo: registrar, fase a fase, como ocorreu a aplicação das atividades do SoPensa, destacando o que funcionou, o que não funcionou e o que precisa ser ajustado.

Orientações de uso: preencher um registro ao final de cada fase aplicada, usando uma cópia deste formulário por fase.

Identificação:

Curso: _____ Turma: _____ Data: _____/_____/_____

Fase: ( ) 1. Diagnosticar e Organizar ( ) 2. Contextualizar e Ambientar ( ) 3. Desafiar e Mediar ( ) 4. Analisar e Consolidar

1. Como a aula desta fase foi conduzida

( ) Desplugada ( ) Plugada ( ) Jogos/gamificação ( ) Situação-problema

Quando houver programação: ( ) em blocos/visual ( ) textual ( ) transição de blocos para texto

2. Instrumentos aplicados nesta fase

( ) Questionário diagnóstico ( ) Rubrica de acompanhamento ( ) Autoavaliação do time

3. O que funcionou bem nesta fase?

_____

_____

4. O que não funcionou ou gerou dificuldade?

_____

_____

5. Que ajustes você faria na próxima aplicação desta fase?

_____

_____

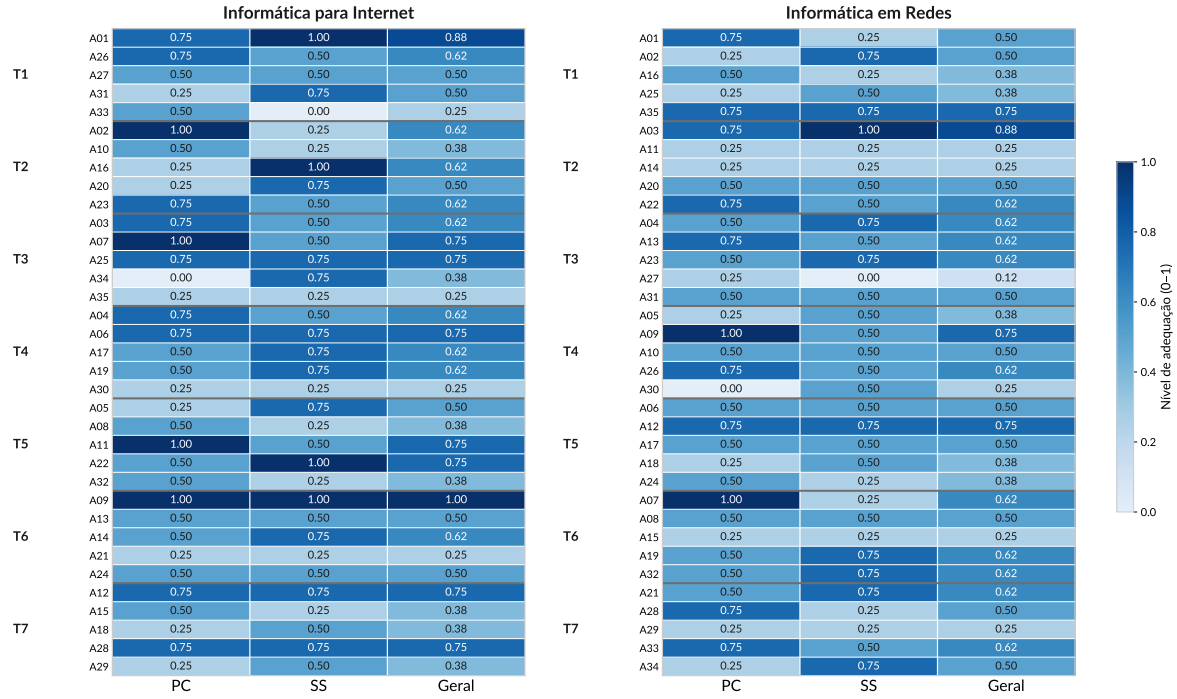
6. Observações sobre o envolvimento da turma e a mediação

_____

_____

APÊNDICE F – Mapa de calor por estudante, agrupado por time, segundo as dimensões de (PC, SS e Geral), nos cursos de Informática para Internet e Informática em Redes

Figura 46: Nível de adequação das dimensões (PC, SS e Geral) por estudante e por time, em cada curso



Fonte: Autoria própria

APÊNDICE G – Questionário de percepção dos estudantes sobre o *framework* SoPensa

Instrumento aplicado aos estudantes ao final da intervenção, de forma anônima e voluntária. Os itens fechados foram respondidos em escala *Likert* de cinco pontos: 1 = Discordo totalmente; 2 = Discordo; 3 = Neutro; 4 = Concordo; 5 = Concordo totalmente. O instrumento contemplou ainda um campo aberto, comum a todo o questionário, para observações gerais. As afirmativas relacionadas ao PC correspondem aos quatro componentes (decomposição, reconhecimento de padrões, abstração e algoritmos); as relacionadas às SS, às quatro competências do *framework* (colaboração, comunicação, criatividade e pensamento crítico); e a afirmativa de resolução de problemas foi incluída de forma transversal.

Identificação

Qual é o seu curso?
( ) Informática para Internet
( ) Redes de Computadores

Visão geral

1. De modo geral, considero que o SoPensa contribuiu para as aulas de Lógica de Programação.

Atividades

Avalie sua percepção sobre as afirmações abaixo:

2. As atividades utilizadas no SoPensa (atividades sem computador, atividades no

computador, jogos, *scratch* e situações-problema) foram úteis para o meu aprendizado.

3. Algumas atividades do SoPensa não acrescentaram muito ao meu aprendizado. *(item reverso)*

4. As atividades do SoPensa tornaram as aulas de Lógica de Programação mais interessantes e motivadoras.

Desenvolvimento do Pensamento Computacional

Avalie sua percepção sobre as afirmações abaixo:

5. Percebo melhora na minha capacidade de dividir problemas em partes menores.

6. Percebo melhora na minha capacidade de identificar padrões em problemas parecidos.

7. Percebo melhora na minha capacidade de focar no que é mais importante e deixar de lado detalhes menos relevantes para resolver um problema.

8. Percebo melhora na minha capacidade de criar um passo a passo para resolver problemas.

Resolução de problemas (transversal)

Avalie sua percepção sobre a afirmação abaixo:

9. Percebo melhora na minha capacidade de enfrentar e buscar soluções para problemas mais complexos.

Desenvolvimento de *Soft Skills*

Avalie sua percepção sobre as afirmações abaixo:

10. Percebo melhora no meu trabalho em equipe durante as atividades.

11. Percebo melhora na minha capacidade de comunicar e explicar ideias aos colegas.

12. Percebo melhora na minha criatividade ao buscar soluções para problemas.

13. Percebo melhora no meu pensamento crítico diante de problemas difíceis.

Recomendação de uso

14. Eu recomendaria que o SoPensa fosse utilizado em outras turmas.

Comentário aberto

Você tem alguma sugestão ou comentário sobre sua participação nas atividades com o SoPensa?

APÊNDICE H – Roteiro da Entrevista com os Professores após a Aplicação do Framework SoPensa

Objetivo: compreender a percepção dos professores sobre a aplicação do *framework* SoPensa em suas aulas de programação, identificando contribuições, desafios, efeitos sobre os alunos e sugestões de melhoria.

Pergunta de pesquisa: como os professores percebem a aplicação do SoPensa em suas práticas de ensino de programação?

Bloco 1 – Contribuições para o ensino-aprendizagem de programação

1. Como você percebeu o papel do SoPensa em suas aulas de programação?
2. O *framework* facilitou ou dificultou a condução do processo de ensino-aprendizagem? Poderia explicar?

Bloco 2 – Funcionalidades e estrutura

3. Como você avalia as práticas pedagógicas propostas pelo SoPensa (atividades desplugadas, plugadas, jogos e gamificação, situações-problema e consolidação), considerando aspectos positivos e pontos a melhorar?
4. Na sua percepção, o SoPensa apresentou algum diferencial em comparação com abordagens tradicionais? Se sim, poderia compartilhar exemplos?

Bloco 3 – Desenvolvimento dos componentes do PC e das competências de SS

5. Na sua percepção, houve mudanças, positivas ou negativas, no desenvolvimento do Pensamento Computacional dos alunos (decomposição, reconhecimento de padrões, abstração e algoritmos)? Pode dar exemplos?

6. Em relação às *Soft Skills* (colaboração, comunicação, criatividade e pensamento crítico), houve evolução ou dificuldades em seu desenvolvimento?

7. Na sua percepção, como o SoPensa influenciou o engajamento e a motivação dos alunos nas aulas de programação?

Bloco 4 – Condições de implementação do SoPensa

8. Quais foram os principais desafios ao aplicar o SoPensa em sala de aula e como você lidou com eles?

9. Como você avalia as condições de tempo, recursos e suporte para a aplicação do SoPensa em sala de aula?

Bloco 5 – Sugestões e melhorias

10. Que ajustes você faria no SoPensa para futuras aplicações e que práticas ou recursos adicionais poderiam enriquecer o *framework*?

Bloco 6 – Considerações finais

11. No geral, quais mudanças positivas ou negativas o SoPensa trouxe para o ensino de programação em sua turma?

12. Você recomendaria o uso do *framework* a outros professores? Por quê?

13. Há algo mais que gostaria de acrescentar sobre sua experiência?

APÊNDICE I – Termos de Consentimento e de Assentimento

I.1 Termo de Consentimento Livre e Esclarecido (TCLE) – Professor

Projeto de pesquisa: SoPensa: Um *Framework* para o Ensino de Programação com Base no Pensamento Computacional e *Soft Skills*.

Convite. Convidamos você, professor(a) das disciplinas de cursos de computação do Instituto Federal, a participar da pesquisa intitulada "SoPensa: Um *Framework* para o Ensino de Programação com Base no Pensamento Computacional e Soft Skills", conduzida pelo pesquisador responsável, Antonio Rege Lopes dos Santos, que pode ser contatado pelo telefone (68) 99911-1042 e pelo*e-mail antonio.rsantos@ifac.edu.br*. Solicitamos que leia este Termo de Consentimento Livre e Esclarecido (TCLE) com atenção e esclareça todas as suas dúvidas sobre a pesquisa e a sua participação. Caso se sinta esclarecido e aceite o convite para participar da pesquisa, pedimos que assine a última página e rubrique as demais páginas das duas vias deste Termo.

Informações sobre a pesquisa. A pesquisa tem como objetivo principal propor um *framework* que integre o Pensamento Computacional e Soft Skills no ensino de programação, visando melhorar o processo de ensino-aprendizagem e alinhá-lo com as demandas contemporâneas do mercado de trabalho. Este estudo caracteriza-se como uma pesquisa aplicada e descritiva, com abordagem qualitativa, que possibilita uma análise aprofundada dos fenômenos educacionais. Para a coleta de dados, serão utilizadas entrevistas semiestruturadas e questionários aplicados aos professores. As entrevistas serão realizadas de forma individual e gravadas em áudio, com duração estimada de 15 a 20 minutos, permitindo uma análise mais detalhada das percepções, estratégias adotadas e dos resultados observados pelos professores durante o processo de aplicação do *framework*. A sua

participação não é obrigatória, e você poderá desistir da pesquisa a qualquer momento, sem prejuízo.

Riscos, benefícios, providências e acompanhamento. A participação na pesquisa pode acarretar os seguintes riscos: Desconforto emocional - reflexões sobre desafios no ensino de programação durante as entrevistas podem gerar desconforto; como medida, o participante poderá pular perguntas ou desistir da pesquisa a qualquer momento, sem prejuízos. Desconforto físico - durante a aplicação do *framework*, o uso prolongado do computador pode gerar fadiga ocular ou desconforto postural; como medida, serão recomendadas pausas regulares, além de orientações sobre postura adequada.

Como benefícios, a pesquisa oferece a oportunidade de refletir sobre os desafios e oportunidades no ensino de programação, identificando pontos fortes e áreas de melhoria; a contribuição para um estudo que subsidie estratégias mais eficazes no ensino de programação nos Institutos Federais e o impacto educacional dos resultados, que poderão contribuir para a melhoria do ensino de programação, beneficiando professores e estudantes no âmbito técnico-profissional.

Confidencialidade. Todas as informações coletadas durante a pesquisa serão tratadas de forma sigilosa e confidencial, respeitando as diretrizes da Resolução Nº 466/2012 do Conselho Nacional de Saúde (CNS). Os áudios e demais registros serão armazenados em um HD protegido por senha, de acesso exclusivo ao pesquisador, por um período de 5 a 10 anos, sendo possível a destruição imediata dos dados após a transcrição. Após a transcrição, os áudios serão permanentemente deletados, assegurando o anonimato e a privacidade dos participantes. O nome ou qualquer outra identificação pessoal não será divulgado em publicações, relatórios ou apresentações de resultados. O participante terá o direito de se recusar a responder perguntas ou desistir da pesquisa a qualquer momento, sem prejuízos de qualquer natureza.

A qualquer tempo, você poderá solicitar outras informações sobre esta pesquisa ao pesquisador responsável, Antonio Rege Lopes dos Santos, pelo telefone (68) 99911-1042 e pelo *e-mail antonio.rsantos@ifac.edu.br*. Você também poderá entrar em contato com o Comitê de Ética em Pesquisa do Instituto Federal do Acre (CEP-IFAC), de segunda a sexta-feira, em horário de expediente: Via Chico Mendes, 3.084, Bairro Areal, CEP 69.906-302, Sala 20, Rio Branco-AC. Fone (68) 98101-8246. *E-mail cep@ifac.edu.br*. Poderá, ainda, contatar a Comissão Nacional de Ética em Pesquisa (CONEP) pelo telefone (61) 3315-5877 ou pelo *e-mail conep@saude.gov.br*.

Declaração do Pesquisador Responsável. Eu, Antonio Rege Lopes dos Santos,

RG 363184 e CPF 689.423.462-00, declaro cumprir todas as exigências éticas contidas nos itens IV.3 e IV.4 da Resolução CNS Nº 466/2012, durante e após a realização da pesquisa.

Consentimento do professor participante. Eu, _____, declaro que fui informado(a) sobre os objetivos, procedimentos, riscos e benefícios da pesquisa "SoPensa: Um Framework para o Ensino de Programação com Base no Pensamento Computacional e *Soft Skills*". Entendi que a minha participação é voluntária e que posso desistir a qualquer momento, sem prejuízo. Concordo com a gravação da entrevista e estou ciente de que as informações serão mantidas em sigilo. Recebi uma via deste Termo de Consentimento Livre e Esclarecido e concordo em participar da pesquisa.

Rio Branco-Acre, _____ de _____ de 202___.

_____

Assinatura do Professor Participante

_____

Assinatura do Pesquisador

I.2 Termo de Consentimento Livre e Esclarecido (TCLE) – Responsáveis

Projeto de pesquisa: SoPensa: Um *Framework* para o Ensino de Programação com Base no Pensamento Computacional e *Soft Skills*.

Convite. Convidamos seu/sua filho/a menor ou legalmente incapaz, sob sua responsabilidade, a participar da pesquisa intitulada "SoPensa: Um *Framework* para o Ensino de Programação com Base no Pensamento Computacional e *Soft Skills*", que tem como pesquisador responsável Antonio Rege Lopes dos Santos, contatável pelo telefone (68) 99911-1042 e pelo *e-mail antonio.rsantos@ifac.edu.br.* Solicitamos que leia este Termo com atenção e esclareça todas as dúvidas. Caso aceite o convite, pedimos que assine a última página e rubrique as demais páginas das duas vias.

Informações sobre a pesquisa. A pesquisa tem por objetivo propor um *framework* adaptável para promover o Pensamento Computacional e as *Soft Skills* no ensino de programação técnico de nível médio, com vistas a melhorar o processo de aprendizagem.

Durante as aulas regulares de Lógica de Programação, conduzidas pelo professor da disciplina, os estudantes participam de atividades práticas e teóricas que aplicam o *framework*. Ao término das atividades, é aplicado um questionário de percepção, respondido de forma anônima, para captar a opinião dos estudantes sobre as atividades e sobre o próprio aprendizado. A participação de seu/sua filho/a refere-se às respostas a esse questionário.

A participação é voluntária, e seu/sua filho/a poderá desistir a qualquer momento, sem prejuízo. A população-alvo é composta pelos estudantes do 1º ano dos cursos Técnico Integrado ao Ensino Médio em Informática para Internet e em Redes de Computadores, cerca de 70 estudantes nas duas turmas. Os dados coletados serão utilizados exclusivamente para fins desta pesquisa e poderão ser publicados em revistas ou eventos científicos, sem identificação dos participantes. As informações serão mantidas em sigilo durante e após a pesquisa.

Riscos, providências e acompanhamento. A participação pode acarretar desconfortos mínimos: leve desconforto diante de alguma pergunta do questionário e desconforto visual ou postural pelo uso do computador durante as atividades. Para minimizá-los, o estudante tem o direito de não responder qualquer pergunta, são feitas pausas regulares com orientação sobre postura, e o pesquisador permanece disponível para esclarecimentos em ambiente confidencial. Durante a pesquisa, seu/sua filho/a será acompanhado de forma cuidadosa e, após o encerramento ou eventual interrupção, continuará a ter direito aos benefícios da pesquisa que lhe couberem. Não há coleta de nome, áudio ou imagem identificável do estudante.

Garantias. Seu/sua filho/a é livre para participar ou não e poderá retirar o consentimento a qualquer tempo, sem penalidade. Será mantido sigilo absoluto sobre sua identidade, com privacidade preservada durante e após a pesquisa. Não haverá pagamento nem despesa pela participação; havendo despesa decorrente, será ressarcida pelo pesquisador. Em caso de dano, seu/sua filho/a será indenizado nos termos da legislação brasileira. Após assinado, você receberá uma via deste TCLE, e seu/sua filho/a assinará o Termo de Assentimento Livre e Esclarecido (TALE), recebendo também uma via. A qualquer tempo, você poderá solicitar informações ao pesquisador responsável pelos contatos acima.

Você poderá entrar em contato com o Comitê de Ética em Pesquisa do Instituto Federal do Acre (CEP-IFAC), de segunda a sexta-feira, em horário de expediente: Via Chico Mendes, 3.084, Bairro Areal, CEP 69.906-302, Sala 20, Rio Branco-AC. Fone (68) 98101-8246. *E-mail cep@ifac.edu.br*. Poderá, ainda, contatar a Comissão Nacional de Ética em Pesquisa (CONEP) pelo telefone (61) 3315-5877 ou pelo *e-mail conep@saude.gov.br*.

Declaração do Pesquisador Responsável. Eu, Antonio Rege Lopes dos Santos, RG 363184 e CPF 689.423.462-00, declaro cumprir todas as exigências éticas contidas nos itens IV.3 e IV.4 da Resolução CNS Nº 466/2012, durante e após a realização da pesquisa.

Consentimento do representante legal. Eu, _____, RG Nº _____, CPF Nº _____, responsável legal pelo/a menor _____ declaro ter sido plenamente informado e esclarecido sobre a pesquisa e seus procedimentos apresentados neste TCLE e consinto de forma livre a participação de meu/minha filho/a ou menor sob minha responsabilidade.

Rio Branco-Acre, _____ de _____ de 202___.

_____

Assinatura do Representante Legal

_____

Assinatura do Pesquisador

I.3 Termo de Assentimento Livre e Esclarecido (TALE) – Estudante

Projeto de pesquisa: SoPensa: Um *Framework* para o Ensino de Programação com Base no Pensamento Computacional e *Soft Skills*.

Você está sendo convidado(a) a participar da pesquisa "SoPensa: Um *Framework* para o Ensino de Programação com Base no Pensamento Computacional e *Soft Skills*", coordenada pelo pesquisador Antonio Rege Lopes dos Santos.

Seu pai, mãe ou responsável legal permitiu a sua participação. Queremos descobrir, com esta pesquisa, como as atividades de Pensamento Computacional e *Soft Skills* podem melhorar o aprendizado em programação, tornando-o mais interessante e útil para você.

Gostaríamos de contar com você, mas você não é obrigado(a) a participar e não tem problema se desistir. A pesquisa acontece no campus do IFAC, e a sua participação envolve duas coisas. Durante as aulas de Lógica de Programação, você participa de atividades práticas e teóricas, junto com o professor, para explorar e aplicar conceitos de programação. As atividades usam o computador, que é seguro, mas você pode sentir

um pouco de cansaço nos olhos se ficar muito tempo usando. Ao final das atividades, você responde a um questionário, sem o seu nome, para nos contar o que achou. É esse questionário que será usado na pesquisa.

Se você se sentir desconfortável, você, seus pais ou responsáveis podem nos procurar pelo telefone (68) 99911-1042 ou pelo *e-mail antonio.rsantos@ifac.edu.br*. A sua participação é importante porque vai nos ajudar a melhorar a forma de ensinar programação. Suas informações ficam em segredo, e ninguém saberá que você participou. Os resultados serão publicados em revistas e eventos científicos, mas sem usar seu nome, imagens ou áudios.

Você pode deixar de participar a qualquer momento, ou não responder a uma pergunta, se não quiser, sem nenhum problema ou prejuízo. Como o questionário é respondido sem identificação, basta não entregá-lo ou deixá-lo em branco para não participar. Se concordar, assine abaixo. Vamos te entregar uma via deste documento.

Assentimento (concordância) de participação. Eu, _____, aceito participar da pesquisa "SoPensa: Um *Framework* para o Ensino de Programação com Base no Pensamento Computacional e *Soft Skills*". Entendi que pode haver riscos e benefícios. Entendi que posso dizer "sim" e participar, mas que, a qualquer momento, posso dizer "não" e desistir, e que ninguém vai ficar com raiva de mim. Os pesquisadores tiraram minhas dúvidas e conversaram com os meus responsáveis. Recebi uma via deste termo, li e concordo em participar.

Rio Branco-Acre, _____ de _____ de 202___.

_____ _____

Assinatura do Participante Assinatura do Pesquisador

Em caso de dúvidas sobre os aspectos éticos desta pesquisa, você poderá consultar o Comitê de Ética em Pesquisa do Instituto Federal do Acre (CEP/IFAC): Via Chico Mendes, 3.084, Bairro Areal, CEP 69.906-302, Sala 20, Rio Branco-AC. Fone (68) 98101-8246. *E-mail cep@ifac.edu.br*.

APÊNDICE J – Registro do sítio eletrônico do SoPensa

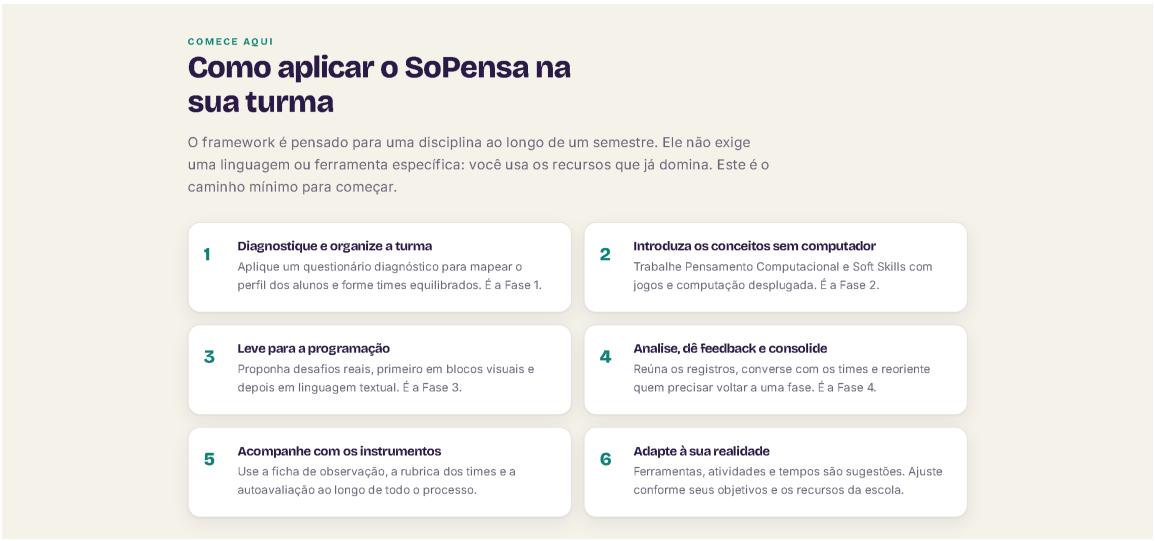
O SoPensa e seus instrumentos foram reunidos em um sítio eletrônico de acesso público, desenvolvido no âmbito desta pesquisa e disponível em https://sopensa.netlify.app. Destinado aos professores que pretendam aplicar o *framework*, o sítio reúne a proposta, as quatro fases, os instrumentos, as atividades e a ferramenta de apoio à formação dos times. Este apêndice apresenta uma síntese das principais telas do sítio. A Figura 47 apresenta a página inicial.

Figura 47: Página inicial do sítio eletrônico do SoPensa

Fonte: Autoria própria.

Entre os recursos disponíveis, o roteiro de entrada sintetiza, em seis passos, o percurso mínimo para conduzir o *framework* ao longo de um semestre letivo. Cada passo indica a fase correspondente e reafirma que ferramentas, atividades e tempos constituem sugestões ajustáveis às condições da instituição. A Figura 48 reproduz a seção.

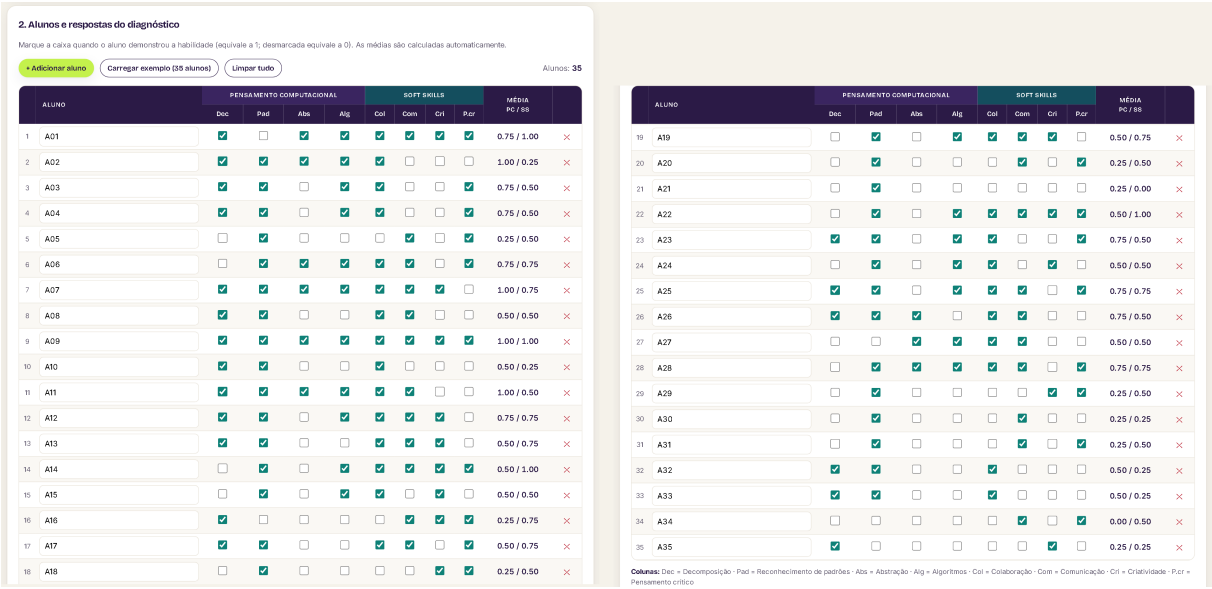
Figura 48: Roteiro de entrada para aplicação do SoPensa apresentado no sítio



Fonte: Autoria própria.

A ferramenta de apoio à formação dos times opera o procedimento adotado na Fase 1 e detalhado na Seção 6.3.3. O professor importa as respostas do questionário diagnóstico do Apêndice E ou registra manualmente os resultados, e a ferramenta calcula as médias individuais nos quatro componentes do PC e nas quatro competências de SS. A Figura 49 apresenta a tela de entrada, com o exemplo de trinta e cinco estudantes.

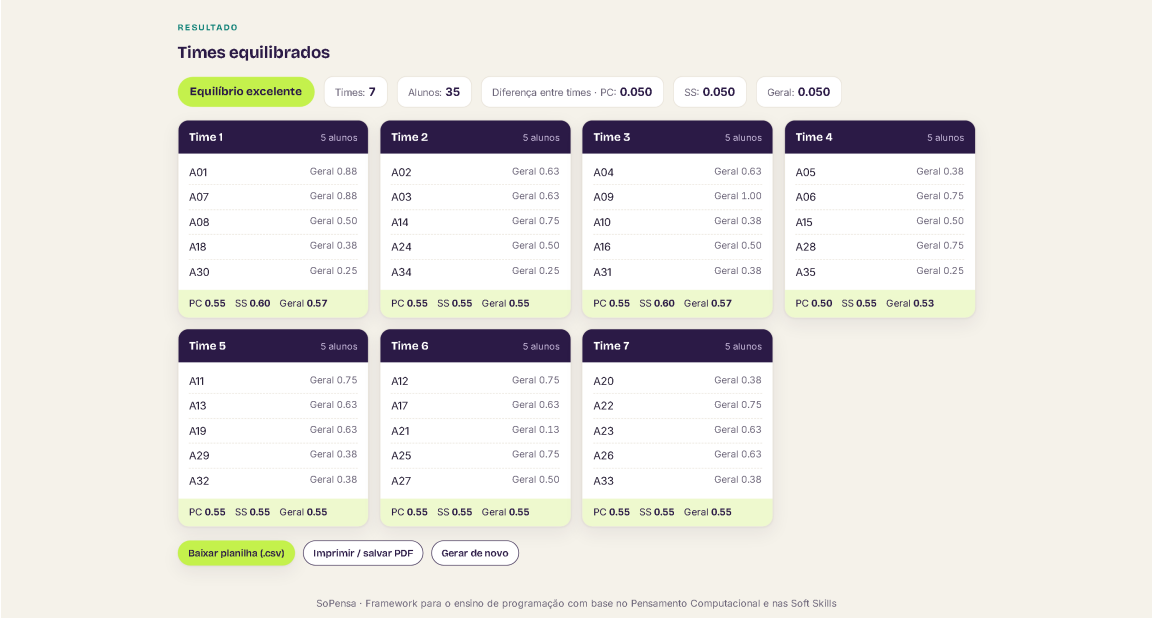
Figura 49: Entrada das respostas do diagnóstico na ferramenta de formação de times



Fonte: Autoria própria. Dados fictícios do exemplo embutido na ferramenta.

A partir dessas médias, a ferramenta compõe os times de modo a aproximar as médias de PC e de SS entre eles e apresenta, além da composição, as diferenças máximas observadas nas duas dimensões e na média geral. Esses indicadores correspondem às medidas de equilíbrio empregadas na aplicação e reportadas nas Tabelas 20 e 21. A Figura 50 apresenta o resultado gerado para o exemplo da tela anterior.

Figura 50: Times gerados pela ferramenta a partir das médias de PC e de SS



Fonte: Autoria própria. Dados fictícios do exemplo embutido na ferramenta.

As figuras registram a versão consultada em 30 de junho de 2026, preservando o conteúdo do produto técnico independentemente da permanência do endereço eletrônico. O sítio indica esta tese como referência de origem para quem utilizar o *framework* e seus instrumentos.

ANEXO A – Capa e Ementa do Curso Técnico Integrado ao Ensino Médio em Informática para Internet

Figura 51: Capa do Projeto Pedagógico do Curso Técnico em Informática para Internet

Fonte: Projeto Pedagógico do Curso Técnico em Informática para Internet, IFAC, Campus Rio Branco, 2017.

Figura 52: Ementa da disciplina de Lógica de Programação do Curso Técnico em Informática para Internet

MINISTÉRIO DA EDUCAÇÃO
Secretaria de Educação Profissional e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia do Acre

- **Ementários 1º ANO**

Componente Curricular	Lógica de Programação		
CH	120h	**Período letivo**	1º
Ementa			
Lógica proposicional. Conceitos de algoritmos. Tipos de dados. Variáveis e constantes. Entrada e saída de dados. Operadores aritméticos, relacionais e lógicos. Comentários. Estruturas de seleção e repetição. Funções e Procedimentos. Recursividade. Vetores e matrizes.			
Ênfase tecnológica			
Raciocínio lógico. Construção de algoritmos.			
Áreas de Integração			
Matemática: operações matemáticas e matrizes. Física: vetores.			
Bibliografia Básica			
SOUZA, João Nunes de. **Lógica para Ciência da Computação.** Campus, 2008. VILLAR, André et. All. **Lógica de Programação – A construção de algoritmos e estrutura de dados**. 3ª Edição. São Paulo, Pearson Prentice Hall, 2005. MANZANO, José Augusto N. G. & OLIVEIRA, Jayr Figueiredo. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores.** São Paulo. Érica, 1996. 270p.			
Bibliografia Complementar			
GUIMARÃES, Ângelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e Estruturas de Dados.** Rio de Janeiro: Editora LTC, 1994. BOOCH, G.; RUMBAUGH, J.; JACOBSON,I. **UML guia do Usuário.** Tradução de Fabio Freitas da Silva. Rio de Janeiro: Elsevier, 2005. CORMEN, Thomas H. RIVEST, Ronald L. LEISERSON, Charles E. STEIN, Clifford. **Algoritmos - Teoria e Prática**. Elsevier – Campus. 2012. GOODRICH, M.T.; TAMASSIA, R.; **Estruturas de dados e algoritmos em Java.** Bookman, 2002. PUGA, S; RISSETTI, Gerson. **Lógica de programação e estruturas de dados com aplicações em Java.** 2. ed. – São Paulo. Pearson Prentice Hall, 2009.			

Componente Curricular	Fundamentos de Desenvolvimento Web		
CH	120h	**Período letivo**	1º
Ementa			
Introdução à web. Diferença e relação entre web e internet. Conceitos básicos de websites, páginas e aplicações web. Linguagem de marcação para web. Linguagem de estilo para web. Linguagem de script para web.			
Ênfase tecnológica			
Construção de páginas web.			
Áreas de Integração			
Língua portuguesa: produção de texto. Língua inglesa: tradução de comandos. Lógica de programação: criação de códigos. Design para sistemas interativos: usabilidade e criação de conteúdo multimídia.			
Bibliografia Básica			
FLANAGAN, David. **Javascript - o Guia Definitivo.** Bookman, 2013. MEYER,Eric A. **Smashing Css - Tecnicas Profissionais Para Um Layout Moderno**. Bookman. 2012. DUCKETT, J. **Introdução à Programação Web com HTML, XHTML e CSS**. Rio de Janeiro: Ciência Moderna, 2010.			
Bibliografia Complementar			

Fonte: Projeto Pedagógico do Curso Técnico em Informática para Internet, IFAC, Campus Rio Branco, 2017.

ANEXO B – Capa e Ementa do Projeto Pedagógico do Curso Técnico Integrado ao Ensino Médio em Redes de Computadores

Figura 53: Capa do Projeto Pedagógico do Curso Técnico em Redes de Computadores

Fonte: Projeto Pedagógico do Curso Técnico em Redes de Computadores, IFAC, Campus Rio Branco, 2017.

Figura 54: Ementa da disciplina de Lógica de Programação do Curso Técnico em Redes de Computadores

CAPRON, H. L.;JOHNSON, J.A. **Introdução à Informática**. São Paulo: Pearson Prentice Hall, 2004.
SILVA, Mário Gomes da. **Informática- terminologia: microsoft windows 8, internet, segurança, microssoft office word 2010, microssoft excel 2010**. 1.ed. São Paulo: Érica, 2012.
VELOSO, Fernando de Castro. **Informática - Conceitos Básicos**. 8.ed. Rio de Janeiro: Elsevier, 2011.
SILBERSCHATZ, Abraham. **Fundamentos de sistemas operacionais**. 8.ed. Rio de Janeiro: LTC, 2010.

Bibliografia Complementar

MANZANO, José Augusto N. G. **BrOffice. Org 3.2.1: Guia prático de aplicações.**1.ed. São Paulo: Érica, 2010.
NEMETH, Evi. **Manual completo do linux**. São Paulo: Pearson Prentice Hall, 2007.
PREPPERNAU, Jouan. **WINDOWS 7 - Passo a passo**. Porto Alegre: Bookman, 2010.
STALLINGS, William. **Arquitetura e organização de computadores**. 8.ed. São Paulo: Pearson Prentice Hall, 2010.
TANENBAUM, Andrew S. **Organização Estruturada de computadores**. 6.ed. São Paulo: Pearson Prentice Hall, 2013

COMPONENTE CURRICULAR: LÓGICA DE PROGRAMAÇÃO			
Carga Horária:	60 h/r	**Período Letivo:**	1º ano

Ementa

Lógica proposicional. Lógica binária e digital. Algoritmos (variáveis, operações básicas, estruturas condicionais, estruturas de repetição, vetores, procedimentos e funções). Linguagem de programação. Programação em Shell Script.

Ênfase Tecnológica

Desenvolvimento de conhecimentos em lógica. Desenvolvimento e construção de conhecimento de programação (com seus elementos básicos) e aplicação do conhecimento de programação em redes de computadores.

Áreas de Integração

INSTITUTO FEDERAL Acre | Campus Rio Branco
Avenida Brasil, 920, Bairro Xavier Maia
Rio Branco/AC - CEP 69.903-068
Telefones: (68) **2106-5900** - (68) **2106-5907** e (68) **2106-5906**
E-mail: campusriobranco@ifac.edu.br

Fonte: Projeto Pedagógico do Curso Técnico em Redes de Computadores, IFAC, Campus Rio Branco, 2017.