UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO INTERDISCIPLINAR DE NOVAS TECNOLOGIAS NA EDUCAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA NA EDUCAÇÃO

TESE DE DOUTORADO

AIPairCode: Um Par de Programação Baseado em Inteligência Artificial Generativa - Um Estudo sobre Eficácia e Motivação no Ensino de Programação

Teresinha Letícia da Silva

Porto Alegre, RS, Brasil
2026

TERESINHA LETÍCIA DA SILVA

AIPairCode: Um Par de Programação Baseado em Inteligência Artificial Generativa - Um Estudo sobre Eficácia e Motivação no Ensino de Programação

Tese apresentada ao Programa de Pós-Graduação em Informática na Educação do Centro Interdisciplinar de Novas Tecnologias na Educação da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Doutora em Informática na Educação.

Orientadora: Profª. Drª. Patrícia Fernanda da Silva

Orientadora: Profª. Drª. Liane Margarida Rockenbach Tarouco *(in memorian)*

Linha de pesquisa: Ambientes Informatizados e Ensino à distância

Porto Alegre, RS, Brasil
2026

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

CIP - Catalogação na Publicação

Silva, Teresinha Leticia da
AIPairCode: Um Par de Programação Baseado em Inteligência Artificial Generativa - Um Estudo sobre Eficácia e Motivação no Ensino de Programação / Teresinha Leticia da Silva. -- 2026.
256 f.
Orientadores: Drª Patricia Fernanda da Silva, Drª Liane Margarida Rockenbach Tarouco (in memorian).

Tese (Doutorado) -- Universidade Federal do Rio Grande do Sul, Centro de Estudos Interdisciplinares em Novas Tecnologias na Educação, Programa de Pós-Graduação em Informática na Educação, Porto Alegre, BR-RS, 2026.

1. Inteligência Artificial Generativa. 2. Ensino de Programação. 3. Programação em Pares. I. Silva, Drª Patricia Fernanda da, orient. II. , Drª Liane Margarida Rockenbach Tarouco (in memorian), orient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Marcia Barbosa
Vice-reitor: Pedro de Almeida Costa
Pró-reitora de Pós-Graduação: Claudia Wasserman
Diretor do Centro Interdisciplinar de Novas Tecnologias na Educação: Prof. Dr. Dante Augusto Couto Barone
Coordenador do Curso de Pós-Graduação em Informática na Educação: Prof. Dr. Crediné Silva de Menezes

ATA DA DEFESA DE TESE DE DOUTORADO

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CINTED – CENTRO INTERDISCIPLINAR DE NOVAS TECNOLOGIAS NA EDUCAÇÃO
PPGIE – PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA NA EDUCAÇÃO

ATA SOBRE A DEFESA DE TESE DE DOUTORADO
TERESINHA LETÍCIA DA SILVA

Às nove horas do dia quinze de junho de dois mil e vinte e seis, no endereço eletrônico https://mconf.ufrgs.br/webconf/00234410, conforme a portaria 02 de 10/10/2022 da PROPG/UFRGS que regulamenta a modalidade híbrida ou a distância para as bancas de defesas de cursos *stricto sensu*, reuniu-se a Comissão de Avaliação, composta pelos Professores Doutores: Adriana Soares Pereira, Dante Augusto Couto Barone e Erico Marcelo Hoff Amaral para a análise da Defesa de Tese de Doutorado intitulada "AIPairCode: Um Par de Programação Baseado em Inteligência Artificial Generativa – Um Estudo sobre Eficácia e Motivação no Ensino de Programação" da doutoranda de Pós-Graduação em Informática na Educação Teresinha Letícia da Silva sob a orientação da Profª. Drª. Patrícia Fernanda da Silva e da Profª. Drª. Liane Margarida Rockenbach Tarouco (in memorian). A Banca, reunida, após a apresentação e arguição, emite o parecer abaixo assinalado.

[X] Considera a Tese Aprovada
(X) sem alterações;
( ) sem alterações, com voto de louvor;
( ) e recomenda que sejam efetuadas as reformulações e atendidas as sugestões contidas nos pareceres individuais dos membros da Banca;

[ ] Considera a Tese Reprovada.

Considerações adicionais (a critério da Banca):
A aluna deve observar as figuras, sumário e demais considerações realizadas pelos membros da banca.

gov.br Documento assinado digitalmente
PATRICIA FERNANDA DA SILVA
Data: 15/06/2026 11:18:35-0300
Verifique em https://validar.iti.gov.br

Profª. Drª. Patrícia Fernanda da Silva
Orientadora

gov.br Documento assinado digitalmente
ADRIANA SOARES PEREIRA
Data: 15/06/2026 11:22:27-0300
Verifique em https://validar.iti.gov.br

Prof.ª Dr.ª Adriana Soares Pereira
UFSM

gov.br Documento assinado digitalmente
DANTE AUGUSTO COUTO BARONE
Data: 16/06/2026 18:21:40-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Dante Augusto Couto Barone
PPGIE/UFRGS

gov.br Documento assinado digitalmente
ERICO MARCELO HOFF DO AMARAL
Data: 16/06/2026 18:46:31-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Erico Marcelo Hoff Amaral
UNIPAMPA

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, pela vida, pela força e pela sabedoria concedidas ao longo desta caminhada. Em todos os momentos, especialmente nos mais difíceis, encontrei na fé o amparo necessário para seguir em frente, superar os desafios e concluir esta etapa tão significativa da minha trajetória acadêmica e pessoal.

À minha filha Rafaella, meu amor maior, agradeço por ser fonte constante de inspiração, alegria e motivação. Sua existência dá sentido aos meus esforços e me impulsiona a buscar sempre o melhor. Obrigada por compreender, mesmo sem perceber completamente, minhas ausências, minhas horas de estudo e dedicação, e por me lembrar diariamente da importância do amor, da persistência e da esperança.

Ao meu marido Rômulo, agradeço pelo companheirismo, apoio, paciência e incentivo durante todo este percurso. Obrigada por estar ao meu lado nos momentos de cansaço, incerteza e ansiedade, oferecendo suporte, compreensão e encorajamento para que eu pudesse continuar.

Aos meus familiares e amigos, agradeço pelo carinho, pelas palavras de incentivo, pela torcida e pelo apoio em diferentes momentos desta caminhada. Cada gesto de cuidado, compreensão e encorajamento foi fundamental para que eu pudesse chegar até aqui.

À Profª Drª Liane Margarida Rockenbach Tarouco, minha orientadora *in memoriam*, registro minha profunda gratidão e admiração. Sua trajetória, dedicação à pesquisa e contribuições para a Informática na Educação permanecem como inspiração. Agradeço pela oportunidade de ter contado com sua orientação e por seus ensinamentos, que continuam presentes neste trabalho e em minha formação acadêmica.

À minha orientadora, Profª Drª Patrícia Fernanda da Silva, expresso minha sincera gratidão pela orientação, confiança, disponibilidade e contribuições ao longo do desenvolvimento desta pesquisa. Agradeço pelas leituras, sugestões, diálogos e pelo acompanhamento cuidadoso, que foram essenciais para a construção e o aprimoramento deste trabalho.

Aos colegas do PPGIE que estiveram presentes ao longo desta trajetória, agradeço pela convivência, pelas trocas de experiências, pelas conversas, pelo apoio e pela colaboração. Compartilhar dúvidas, desafios, conquistas e aprendizados tornou este percurso mais leve e significativo.

Ao Programa de Pós-Graduação em Informática na Educação da Universidade Federal do Rio Grande do Sul, agradeço pela oportunidade de formação, pelo acolhimento acadêmico e pelas experiências proporcionadas ao longo do doutorado.

À Universidade Federal de Santa Maria, agradeço pela oportunidade de qualificação e aperfeiçoamento, bem como pelo espaço concedido para a realização desta pesquisa. Agradeço, em especial, ao Departamento de Tecnologia da Informação e aos alunos do curso de Sistemas de Informação, que contribuíram para o desenvolvimento deste trabalho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), agradeço pelo apoio financeiro concedido por meio da bolsa de estudos, que possibilitou a dedicação necessária ao desenvolvimento desta pesquisa.

Por fim, agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho. Cada palavra de incentivo, cada gesto de apoio e cada contribuição recebida foram importantes para a conclusão desta tese.

RESUMO

O ensino de programação, especialmente em disciplinas introdutórias, enfrenta diversos desafios, tais como altas taxas de evasão e dificuldades de aprendizagem, que são frequentemente associados à falta de suporte individualizado aos estudantes. Neste contexto, este trabalho investiga o uso da Inteligência Artificial (IA) generativa como par de programação no ensino de programação. A pesquisa fundamenta-se nas teorias de Vygotsky, em especial na zona de desenvolvimento proximal (ZDP) e no conceito de mediação, e apresenta o desenvolvimento do AIPairCode, um assistente de programação baseado em IA generativa. O sistema foi projetado para atuar como um par de programação com foco pedagógico, oferecendo suporte individualizado por meio de perguntas inspiradas no questionamento socrático, orientações passo a passo e restrição do fornecimento de respostas completas. Em vez de entregar soluções prontas, o assistente busca incentivar o raciocínio do estudante durante a resolução dos exercícios. Além disso, o nível de ajuda fornecido é ajustado de acordo com o progresso e as dificuldades identificadas ao longo da interação, procurando manter o suporte alinhado à ZDP do aluno. A pesquisa adotou abordagem mista, de natureza exploratória e aplicada, por meio de estudos de caso com grupos controle e experimental. Os resultados indicam que os estudantes que utilizaram o AIPairCode apresentaram maiores ganhos de aprendizagem em comparação aos métodos tradicionais, observado tanto nas diferenças entre pré-teste e pós-teste quanto nos tamanhos de efeito obtidos. As análises inferenciais também apontaram diferenças significativas entre as condições investigadas, sugerindo contribuição consistente do assistente para o desempenho dos estudantes. No aspecto motivacional, observaram-se resultados positivos em dimensões como interesse, percepção de competência e valor percebido da atividade. A análise das interações indicou ainda um maior engajamento dos estudantes no processo de resolução de problemas. Os resultados também demonstraram elevada usabilidade e boa aceitação da ferramenta pelos participantes. De modo geral, os achados sugerem que o uso de IA generativa como par de programação, aliado a princípios pedagógicos como mediação e questionamento socrático, pode contribuir para uma aprendizagem mais personalizada e significativa no ensino de programação.

Palavras-chave: Inteligência Artificial Generativa; Ensino de Programação; Programação em Pares.

AIPairCode: A Generative Artificial Intelligence-Based Programming Partner - A Study on Effectiveness and Motivation in Programming Education

ABSTRACT

Programming education, especially in introductory courses, faces several challenges, such as high dropout rates and learning difficulties, which are often associated with the lack of individualized support for students. In this context, this study investigates the use of generative Artificial Intelligence (AI) as a programming partner in programming education. The research is grounded in Vygotsky's theories, particularly the Zone of Proximal Development (ZPD) and the concept of mediation, and presents the development of AIPairCode, a programming assistant based on generative AI. The system was designed to act as a pedagogically oriented programming partner, providing individualized support through questions inspired by Socratic questioning, step-by-step guidance, and restrictions on delivering complete answers. Instead of providing ready-made solutions, the assistant seeks to encourage students' reasoning during problem-solving activities. In addition, the level of assistance provided is adjusted according to the students' progress and the difficulties identified throughout the interaction, aiming to keep the support aligned with the learner's ZPD. The research adopted a mixed-methods approach of an exploratory and applied nature, conducted through case studies involving control and experimental groups. The results indicate that students who used AIPairCode achieved greater learning gains compared to traditional methods, as observed both in the differences between pre-test and post-test scores and in the obtained effect sizes. Inferential analyses also revealed significant differences between the investigated conditions, suggesting a consistent contribution of the assistant to students' performance. From a motivational perspective, positive results were observed in dimensions such as interest, perceived competence, and perceived value of the activity. The analysis of interactions also indicated greater student engagement in the problem-solving process. The results further demonstrated high usability and good acceptance of the tool among participants. Overall, the findings suggest that the use of generative AI as a programming partner, combined with pedagogical principles such as mediation and Socratic questioning, can contribute to a more personalized and meaningful learning experience in programming education.

Keywords: Generative Artificial Intelligence; Programming Education; Pair Programming.

LISTA DE FIGURAS

Figura 1 - Etapas para resolução de problemas com programação.....29
Figura 2 - Funcionamento dos recursos do GitHub Copilot.....56
Figura 3 - Classificação dos artigos com relação ao objetivo da pesquisa.....69
Figura 4 - Processo de busca e seleção dos artigos.....75
Figura 5 - Classificação da pesquisa.....83
Figura 6 - Fases da Pesquisa.....86
Figura 7 - Etapas do Estudo Preliminar.....91
Figura 8 - Roteiro básico dos estudos do grupo experimental e controle.....94
Figura 9 - Arquitetura Geral do AIPairCode.....102
Figura 10 - Estrutura de construção do *prompt* base.....107
Figura 11 - Exemplo de inserção dinâmica de contexto pedagógico no *backend*.....109
Figura 12 - Exemplo de implementação de *guardrails* no *backend*.....112
Figura 13 - Diagrama de casos de uso da extensão do AIPairCode.....116
Figura 14 - Telas de login e interface inicial da extensão.....116
Figura 15 - Iniciar novo exercício na extensão.....117
Figura 16 - Uso do botão *"Como começar?"*.....118
Figura 17 - Exemplo do botão *"Explica esse trecho"*.....119
Figura 18 - Exemplos de uso do botão *"Corrige esse trecho"*.....119
Figura 19 - Sugestões de correções *inline* no editor.....120
Figura 20 - Tela de listagem de exercícios.....121
Figura 21 - Diagrama de casos de uso do módulo web do estudante.....123
Figura 22 - Página de login e interface inicial do módulo web do estudante.....124
Figura 23 - Gráficos de interações por semana e dificuldades mais frequentes do aluno.....124
Figura 24 - Diagnóstico de aprendizagem e plano rápido da semana.....125
Figura 25 - Lista de exercícios desenvolvidos.....126
Figura 26 - Interface de resolução de um exercício.....127
Figura 27 - Exemplo do processo de execução de testes automatizados.....128
Figura 28 - Interface de *chat* do módulo web do estudante.....129
Figura 29 - Mini desafios.....130
Figura 30 - Exercícios extras.....130
Figura 31 - Materiais da disciplina.....131
Figura 32 - Diagrama de casos de uso do módulo web do professor.....133
Figura 33 - Página inicial do professor com indicadores gerais da turma.....134
Figura 34 - Gráfico de resumo das interações.....135
Figura 35 - Visualização das dificuldades mais frequentes da turma.....136
Figura 36 - Visualização das dificuldades por exercício.....136
Figura 37 - Página de acompanhamento individual de um aluno.....137
Figura 38 - Histórico de interações entre aluno e assistente.....138

Figura 39 - Interface de gerenciamento de materiais da disciplina (RAG).....139
Figura 40 - Interface geração da lista de exercícios.....140
Figura 41 - Lista de exercícios gerada.....141
Figura 42 - Estudo preliminar: nível de conhecimento em programação.....147
Figura 43 - Estudo preliminar: nível de conhecimento em C++.....147
Figura 44 - Estudo preliminar: Média geral motivação por dimensão.....151
Figura 45 - Estudo preliminar: Percentual médio de respostas positivas por dimensão.....152
Figura 46 - Estudo Final 1: autoavaliação do conhecimento prévio em JavaScript.....162
Figura 47 - Estudo Final 1: Gráfico de desempenho médio no pré-teste por grupo.....164
Figura 48 - Estudo Final 1: Gráfico das médias do pré-teste e do pós-teste nos grupos controle e experimental.....166
Figura 49 - Estudo Final 1: Distribuição do ganho de aprendizagem nos grupos controle e experimental.....168
Figura 50 - Estudo Final 1: Média do IMI por Dimensão.....170
Figura 51 - Estudo Final 1: Distribuição dos escores SUS por estudante.....173
Figura 52 - Estudo Final 1: Detalhamento das respostas do SUS por questão.....174
Figura 53 - Estudo Final 1: Média de interações por aluno/exercício.....179
Figura 54 - Estudo Final 1: Distribuição percentual dos tipos de resposta da IA.....183
Figura 55 - Estudo Final 2: Autoavaliação do conhecimento prévio em C++ por grupo.....192
Figura 56 - Estudo Final 2: Desempenho médio no pré-teste por grupo.....193
Figura 57 - Estudo Final 2: Médias do pré-teste e do pós-teste nos grupos controle e experimental.....195
Figura 58 - Estudo Final 2: Distribuição do ganho de aprendizagem nos grupos controle e experimental.....197
Figura 59 - Estudo Final 2: Gráfico da média do IMI por dimensão.....199
Figura 60 - Estudo Final 2: Distribuição dos escores SUS totais para a extensão do VS Code 203
Figura 61: - Estudo Final 2: Detalhamento das respostas do SUS por questão para a Extensão do VS Code.....204
Figura 62 - Estudo Final 2: Distribuição dos escores SUS do módulo web do AIPairCode. .208
Figura 63 - Estudo Final 2: Detalhamento das respostas do SUS por questão para o módulo web do AIPairCode.....208
Figura 64 - Estudo Final 2: Recursos mais utilizados no módulo web.....210
Figura 65 - Estudo Final 2: Média de interações por aluno/exercício.....215
Figura 66 - Estudo Final 2: Distribuição percentual das categorias de resposta da IA.....219

LISTA DE TABELAS

Tabela 1 - Resultados da Busca.....68
Tabela 2 - Resumo da participação dos estudantes no Estudo Preliminar.....145
Tabela 3 - Estudo preliminar: Categorias de interação com o ChatGPT.....149
Tabela 4 - Resumo da participação dos estudantes no Estudo Final 1.....161
Tabela 5 - Estudo Final 1: Estatística Descritiva do Pré-teste.....164
Tabela 6 - Estudo Final 1: Estatísticas descritivas do ganho de aprendizagem.....167
Tabela 7 - Estudo Final 1: Categorias de solicitação dos estudantes ao AIPairCode.....180
Tabela 8 - Resumo da participação dos estudantes no Estudo Final 2.....190
Tabela 9 - Estudo Final 2: Estatística Descritiva do Pré-teste.....193
Tabela 10 - Estudo Final 2: Estatísticas descritivas do ganho de aprendizagem por grupo…196
Tabela 11 - Estudo Final 2: Categorias de solicitação dos estudantes ao AIPairCode.....216

LISTA DE QUADROS

Quadro 1 - Principais técnicas utilizadas em IA generativa ……. 53
Quadro 2 - Sugestões de melhoria indicadas pelos participantes do estudo piloto …….. 143
Quadro 3 - Estudo preliminar: Opiniões dos alunos sobre o uso de IA ……... 158

LISTA DE ABREVIATURAS E SIGLAS

ABP	Aprendizagem Baseada em Problemas
ABPj	Aprendizagem Baseada em Projetos
BNCC	Base Nacional Comum Curricular
CES/CNE	Câmara de Educação Superior do Conselho Nacional de Educação
DCNs	Diretrizes Curriculares Nacionais
GANs	Redes Adversariais Generativas
GPT	*Generative Pre-trained Transformer*
IA	Inteligência Artificial
IDE	Ambiente de Desenvolvimento Integrado
IES	Instituições de Educação Superior
LLM	*Large Language Models*
ML	*Machine Learning*
MSL	Mapeamento Sistemático da Literatura
PC	Pensamento Computacional
RFs	Referenciais de Formação em Computação
RNAs	Redes Neurais Artificiais
RSL	Revisão Sistemática da Literatura
SBC	Sociedade Brasileira de Computação
UFRGS	Universidade Federal do Rio Grande do Sul
UFSM	Universidade Federal de Santa Maria
TIC	Tecnologias de Informação e Comunicação
ZDP	Zona de Desenvolvimento Proximal

SUMÁRIO

1 INTRODUÇÃO....16
1.1 Questão de Pesquisa....23
1.2 Objetivos....24
1.2.1 Objetivo Geral....24
1.2.2 Objetivos Específicos....24
1.3 Estrutura da Tese....25
2 REFERENCIAL TEÓRICO....27
2.1 Ensino de Programação de Computadores....27
2.1.1 Diretrizes Curriculares para Cursos de Computação no Brasil....30
2.1.2 Dificuldades Inerentes à Aprendizagem de Programação....33
2.2 Teorias de Aprendizagem Aplicadas ao Ensino de Programação....36
2.2.1 Teoria Sociocultural de Vygotsky (ZDP e Mediação)....37
2.2.2 Método Socrático....40
2.2.3 Metodologias Ativas....42
2.2.4 Aprendizagem Baseada em Problemas....45
2.2.5 Programação em Pares (*Pair Programming*)....46
2.3 Inteligência Artificial na Educação....49
2.3.1 Inteligência Artificial Generativa....51
2.3.2 Aplicações da IA Generativa na Educação....56
2.3.3 Desafios da IA Generativa na Educação....60
2.3.4 Inteligência Artificial Generativa no Ensino de Programação....61
3. TRABALHOS RELACIONADOS....67
3.1 MSL - Panorama Geral Sobre o Uso Inteligência Artificial Generativa no Ensino de Programação....67
3.2 RSL - IA Generativa no Ensino de Programação: Estratégias Pedagógicas e Metodologias....74
3.3 Diferenciais da Pesquisa....81
4 PROCEDIMENTOS METODOLÓGICOS....83
4.1 Classificação da Pesquisa....83
4.2 Participantes da Pesquisa....85
4.3 Fases da Pesquisa....86
4.3.1. Fase A: Fundamentação....86
4.3.2 Fase B: Planejamento e Desenvolvimento....87
4.3.2.1 Instrumentos....88
4.3.2.2 Estudo Preliminar....89
4.3.2.3 Desenvolvimento do Assistente de Programação....91
4.3.3. Fase C: Estudos Empíricos....92

4.3.4. Fase D: Análise dos Dados....95
5. DESENVOLVIMENTO DO ASSISTENTE DE PROGRAMAÇÃO AIPAIRCODE....98
5.1 Arquitetura Geral do Sistema....100
5.1.1 Assistants API....103
5.1.2 *Retrieval-Augmented Generation* (RAG)....105
5.1.3 Engenharia de *Prompt*....106
5.1.4 Inserção de *Guardrails* no AIPairCode....111
5.2 Módulos do AIPairCode....114
5.2.1 Extensão para o Visual Studio Code....114
5.2.2 Módulo Web do Estudante....122
5.2.3 Módulo Web do Professor....131
5.3 Avaliação Inicial do AIPairCode....142
6. RESULTADOS....144
6. 1 Estudo Preliminar: Uso de IA como Par de Programação....144
6.1.1 Perfil dos Participantes....146
6.1.2 Padrões de uso da IA como Par de Programação (Grupo Aluno+IA)....148
6.1.3 Questionário de Motivação....151
6.1.3.1 Dimensão: Interesse/Prazer....152
6.1.3.2 Dimensão: Competência Percebida....153
6.1.3.3 Dimensão: Valor/Utilidade Percebida....154
6.1.3.4 Dimensão: Autonomia....154
6.1.3.5 Dimensão: Engajamento/Motivação para Continuidade....155
6.1.3.6 Dimensão: Relacionamento com o Par....156
6.1.3.7 Síntese geral dos resultados do Questionário de Motivação....156
6.1.4 Percepções Qualitativas dos Estudantes....158
6.2 Estudo Final 1 - Aplicação do AIPairCode em uma Disciplina de JavaScript....160
6.2.1 Caracterização das Turmas e Perfil dos Participantes....161
6.2.2 Resultados de Aprendizagem: Análise do Pré-teste e Pós-teste....163
6.2.2.1 Desempenho Inicial no Pré-teste....163
6.2.2.2 Evolução do Desempenho ao Longo da Intervenção....165
6.2.2.3 Comparação do Ganho de Aprendizagem entre os Grupos....167
6.2.3 Motivação e Percepções dos Estudantes (IMI)....169
6.2.4 Avaliação de Usabilidade do Assistente (SUS)....172
6.2.5 Percepções dos Estudantes Sobre o Uso do Assistente de Programação....175
6.2.6 Padrões de Interação com o AIPairCode....177
6.2.6.1 Categorias de solicitação dos estudantes ao AIPairCode....178
6.2.6.2 Tipos de resposta do AIPairCode às solicitações dos estudantes....183
6.2.7 Discussão Integrada dos Resultados do Estudo Final 1....186
6.3 Estudo Final 2: Aplicação do AIPairCode em uma Disciplina de C++....188

6.3.1 Caracterização das Turmas e Perfil dos Participantes.....190
6.3.2 Resultados de Aprendizagem: Análise do Pré-teste e Pós-teste.....192
6.3.2.1 Desempenho Inicial no Pré-teste.....193
6.3.2.2 Evolução do Desempenho ao Longo da Intervenção.....194
6.3.2.3 Comparação do Ganho de Aprendizagem entre os Grupos.....196
6.3.3 Motivação e Percepções dos Estudantes (IMI).....198
6.3.4 Avaliação de Usabilidade do Assistente (SUS).....202
6.3.4.1 Usabilidade da Extensão AIPairCode no Visual Studio Code.....203
6.3.4.2 Percepções dos estudantes sobre o uso da Extensão do VS Code.....205
6.3.4.3 Usabilidade do Módulo Web do AIPairCode.....207
6.3.4.4 Percepções dos estudantes sobre o uso do Módulo Web.....209
6.3.5 Padrões de interação com o AIPairCode.....214
6.3.5.1 Categorias de solicitação dos estudantes ao AIPairCode.....214
6.3.5.2 Categorias de resposta da IA às solicitações dos estudantes.....219
6.3.6 Discussão Integrada do Estudo Final 2.....221
6.4 Síntese dos Resultados dos Estudos Empíricos.....224
7 CONSIDERAÇÕES FINAIS.....226
REFERÊNCIAS.....229
APÊNDICE I - TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO.....244
APÊNDICE II - QUESTIONÁRIO DEMOGRÁFICO SOBRE O PERFIL DOS ESTUDANTES PARTICIPANTES DA ATIVIDADE.....245
APÊNDICE III - QUESTIONÁRIO DE MOTIVAÇÃO BASEADO NO IMI.....247
APÊNDICE IV - QUESTIONÁRIOS DE AVALIAÇÃO DE USABILIDADE.....248
APÊNDICE V - QUESTIONÁRIO DE PERCEPÇÃO E MOTIVAÇÃO COM O PAR DE PROGRAMAÇÃO (IA OU HUMANO) - ESTUDO PRELIMINAR.....249
APÊNDICE VI - TABELA DE NOTAS DOS ALUNOS DO ESTUDO FINAL 1.....250
APÊNDICE VII - TABELA DE NOTAS DOS ALUNOS DO ESTUDO FINAL 2.....251
APÊNDICE VIII - LISTA DE PUBLICAÇÕES REALIZADAS DURANTE O DESENVOLVIMENTO DA TESE.....252

1 INTRODUÇÃO

Na sociedade contemporânea, o avanço tecnológico tem redefinido não apenas a maneira como as pessoas vivem, mas também as habilidades necessárias para prosperar no mundo atual. As inovações tecnológicas trouxeram mudanças significativas, transformando a forma de realizar diversas atividades nos âmbitos social e profissional.

Nesse contexto, as competências digitais tornaram-se fundamentais para promover a inovação, a criatividade e o empreendedorismo no ambiente socioeducativo, sendo consideradas essenciais para o sucesso e a participação efetiva na era da informação. Essas competências envolvem a capacidade de utilizar, compreender e explorar tecnologias digitais para comunicação, colaboração, busca de informações, resolução de problemas e criação de conteúdo (PIMENTEL *et al.*, 2024; TAROUCO *et al*., 2023).

A competência digital é indispensável para todos os cidadãos do século XXI, tanto nas esferas pessoais quanto profissionais, conferindo uma vantagem significativa na ampliação das oportunidades proporcionadas pela era digital. Lucas, Moreira e Trindade (2022), destacam essa competência como essencial para a aprendizagem ao longo da vida, envolvendo o uso confiante, crítico e responsável de tecnologias digitais para fins educacionais e profissionais. Essas habilidades abrangem desde o uso básico até a criação, programação e proteção de conteúdo digital, bem como, a interação efetiva com softwares, dispositivos, inteligência artificial e robôs.

Além disso, a competência digital é um fator determinante no processo de aprendizagem, pois permite que os alunos acessem o conhecimento de forma dinâmica por meio de ferramentas tecnológicas, impactando diretamente seu desempenho acadêmico.

No contexto educacional, as instituições de ensino desempenham um papel fundamental na promoção do desenvolvimento dessa competência, por meio de diferentes abordagens. Entre elas, destacam-se a capacitação dos estudantes no uso eficiente de ferramentas digitais, a integração de recursos tecnológicos ao ensino das disciplinas tradicionais e o uso de programação como estratégia para incentivar os estudantes a se tornarem criadores de tecnologias. Quando essas abordagens são combinadas, os estudantes passam a interagir de maneira mais significativa e a contribuir ativamente no ambiente digital (TAROUCO *et al*., 2023).

No entanto, a abordagem voltada à formação de criadores de tecnologia, por meio da

programação, ainda não recebe a devida atenção. A habilidade de programar representa um avanço significativo na competência digital, permitindo que os indivíduos ultrapassem o papel de meros usuários e se tornem desenvolvedores de novas tecnologias (GARCIA; ATOCHE, 2022; PIMENTEL *et al.*, 2024; SILVA; TAROUCO, 2022).

As habilidades de programação permitem que os indivíduos compreendam e naveguem no ambiente digital com maior eficiência, sendo essenciais para o desenvolvimento de habilidades de resolução de problemas e pensamento crítico, proporcionando uma forma estruturada de expressar ideias e enfrentar desafios em diferentes contextos. Mesmo para quem não atua diretamente como desenvolvedor de software, a capacidade de programar pode ser útil em diversas áreas. Nesse sentido, a educação em programação contribui para o desenvolvimento da criatividade e da inovação, capacitando os indivíduos a criar tecnologias e ferramentas digitais que impulsionem o crescimento econômico (OLIVEIRA *et al.*, 2024; YILMAZ; KARAOGLAN YILMAZ, 2023).

O ensino de programação busca capacitar os estudantes a resolver problemas de forma organizada, eficiente e produtiva, tornando-os aptos a compreender e criar algoritmos, sistemas e tecnologias para apoiar a tomada de decisões em diversos segmentos da sociedade (OLIVEIRA *et al.*, 2024). A programação está diretamente relacionada à habilidade de solucionar problemas, exigindo e promovendo o desenvolvimento de competências como abstração, compreensão, planejamento e criatividade. Além disso, é amplamente reconhecida como uma ferramenta eficaz para estimular o Pensamento Computacional, que envolve a resolução de problemas por meio de estratégias da Ciência da Computação (WING, 2006).

Embora o aprendizado de programação tenha impacto positivo na vida acadêmica dos estudantes, aprender a programar apresenta desafios para muitos alunos devido à sua natureza intrinsecamente complexa, que requer habilidades avançadas de abstração e raciocínio lógico (TSAI, 2019; YUSOFF *et al*., 2020). Além desses aspectos cognitivos, estudos recentes apontam que os estudantes também enfrentam dificuldades relacionadas à resolução de problemas, à elaboração de estruturas lógicas e à compreensão da sintaxe das linguagens de programação, bem como barreiras emocionais, como falta de motivação, ansiedade e frustração (LIMA; MENEZES, 2024). Como resultado, muitos alunos desistem antes de concluir com êxito a formação em programação, e essas disciplinas apresentam altos índices de reprovação e evasão (WATSON; LI, 2014).

As disciplinas de programação são parte integrante do currículo de cursos nas áreas de computação e engenharias, sendo geralmente voltadas ao desenvolvimento de habilidades de

resolução de problemas, compreensão de conceitos, assimilação da sintaxe e da semântica das linguagens de programação, além da aplicação dessas linguagens na construção de soluções. Essas disciplinas, especialmente no início do curso, desempenham um papel fundamental no percurso acadêmico e profissional dos estudantes (MEDEIROS; FALCÃO; RAMALHO, 2020; ZORZO *et al.*, 2017).

O processo de ensino e aprendizagem de programação é complexo, pois tanto alunos quanto professores enfrentam diversos desafios, resultando em altas taxas de retenção e evasão nessas disciplinas (ALVIM; BITTENCOURT; DURAN, 2024; BENNEDSEN; CASPERSEN, 2019). Não há consenso sobre os principais problemas enfrentados nesse processo, já que as dificuldades podem ocorrer por diferentes razões (DURAN *et al.*, 2023; MEDEIROS; RAMALHO; FALCÃO, 2019). A programação exige que o aluno elabore soluções que integrem linguagem de programação e raciocínio lógico, sendo esse um dos pontos em que surgem dificuldades para estruturar e formalizar soluções adequadas aos problemas propostos (FERREIRA; REATEGUI, 2020).

Além disso, programar pode ser compreendido como um comportamento humano complexo, que envolve múltiplas interações cognitivas e contextuais, tornando seu aprendizado ainda mais desafiador (HENKLAIN; CARVALHO; FEITOSA, 2025). Nesse contexto, destacam-se dificuldades recorrentes relacionadas à conversão de abstrações em algoritmos funcionais, à depuração de erros e à interpretação de problemas, especialmente em disciplinas introdutórias (LIMA; MENEZES, 2024).

Pesquisas indicam que a dificuldade com abstração e a construção de raciocínio lógico, especialmente na compreensão de algoritmos, é um dos maiores obstáculos (BOSSE, 2020). Caspersen (2023) também aponta que, entre as dificuldades enfrentadas por iniciantes em programação, estão os problemas relacionados ao domínio formal das linguagens de programação (sintaxe e semântica) e as dificuldades em adquirir esquemas padronizados para resolver problemas de programação.

Outro fator que contribui para as dificuldades na aprendizagem de programação é a metodologia de ensino adotada. O uso de abordagens tradicionais pode ser considerado um dos agravantes desses problemas, pois muitas vezes não consegue motivar o estudante a se interessar pelo conteúdo, nem compreender sua relevância. O ensino expositivo, focado exclusivamente na resolução de problemas e exercícios, pode resultar em falta de motivação e baixo desempenho dos alunos. Além disso, a indisponibilidade dos professores para oferecer um atendimento personalizado, capaz de identificar as dificuldades e lacunas individuais de

cada aluno, torna mais grave os desafios no processo de aprendizagem (AMARAL, 2015; FERREIRA, 2019).

Adicionalmente, estudos indicam que a ausência de metodologias mais dinâmicas e interativas pode intensificar essas dificuldades, reforçando a necessidade de abordagens pedagógicas mais ativas e centradas no aluno (LIMA; MENEZES, 2024). Essas dificuldades não se restringem apenas a aspectos cognitivos, mas também estão relacionadas a fatores como interpretação de texto, formação prévia em matemática e condições do contexto educacional, o que evidencia a natureza diversificada do problema (HENKLAIN; CARVALHO; FEITOSA, 2025).

Neste sentido, visando aprimorar os processos de ensino e aprendizagem nas disciplinas de programação, é necessário ajustar os métodos de ensino atuais, buscando proporcionar um processo mais envolvente e eficaz tanto para alunos quanto para professores. Uma abordagem amplamente utilizada, tanto no apoio a estudantes do ensino superior que estão se familiarizando com esses conceitos, quanto no desenvolvimento do pensamento computacional na educação básica, é a combinação de ferramentas tecnológicas com metodologias educacionais apropriadas. Essa combinação visa promover uma interação mais ativa e motivadora dos estudantes com o aprendizado de programação.

Sobral (2021a) realizou uma revisão sistemática da literatura sobre estratégias de ensino de programação no ensino superior, abrangendo periódicos indexados na *Scopus* no período de 2000 a 2019. Entre as abordagens analisadas, destacam-se as metodologias ativas, que envolvem os alunos em atividades que estimulam a reflexão sobre ideias e suas aplicações, mantendo-os mentalmente engajados por meio de tarefas que incluem a coleta de informações, o pensamento crítico e a resolução de problemas. Foram identificadas como principais metodologias aplicadas ao ensino de programação: a programação em pares (OTHMAN *et al*., 2019), a aprendizagem baseada em jogos (STEPHAN; OLIVEIRA; RENHE, 2020; KARLING; RIZZI; RIZZI, 2021) e a aprendizagem baseada em problemas e projetos (COELHO; GUEDES, 2021; CHANG; CHUNG; CHANG, 2020).

O estudo de Oliveira *et al.* (2024) evidencia também que, no contexto da Educação 4.0 no ensino superior, as metodologias didáticas para o ensino de programação incluem, em sua maioria, a aprendizagem baseada em problemas, que incentiva a resolução de problemas reais, e a aprendizagem colaborativa, que promove o trabalho em equipe e a troca de conhecimentos entre os alunos. Além disso, abordagens adaptativas, que personalizam o ensino conforme as necessidades individuais dos estudantes, são cada vez mais comuns, considerando os

diferentes níveis de familiaridade com a programação. O uso de ferramentas tecnológicas, como programação em blocos e ambientes virtuais de aprendizagem, também é apontado como estratégia para tornar o aprendizado mais acessível, motivador e alinhado às demandas do mercado digital.

Como embasamento teórico desta tese, destaca-se o conceito sociointeracionista de Vygotsky (1978), que compreende o aprendizado como um processo social, no qual o desenvolvimento do aluno é influenciado pelas interações com seus pares. A zona de desenvolvimento proximal (ZDP) é o conceito central dessa teoria, sendo definida como o espaço entre o nível de desenvolvimento real, caracterizado pela capacidade de resolver problemas individualmente, e o nível de desenvolvimento potencial, alcançado com o apoio de um adulto ou em colaboração com colegas mais experientes. Com base neste conceito, as interações sociais tornam-se fundamentais para a construção do conhecimento, especialmente aquelas que envolvem diálogo, colaboração, troca de informações e confronto de diferentes pontos de vista.

Nesse contexto, destaca-se a aprendizagem colaborativa, entendida como o processo que ocorre quando duas ou mais pessoas aprendem, ou buscam aprender algo juntas. Segundo Dillenbourg (1999), a aprendizagem entre pares não ocorre simplesmente pela presença de dois alunos, mas pela realização de atividades que ativam mecanismos específicos de aprendizagem. Isso inclui tarefas realizadas individualmente, uma vez que a cognição individual é mantida mesmo durante a interação com o outro. Além disso, a interação entre os indivíduos promove atividades adicionais, como explicações, debates e regulação mútua, que desencadeiam mecanismos cognitivos complementares. Assim, a aprendizagem colaborativa entre pares constitui um aspecto relevante a ser considerado no ensino de programação.

Adicionalmente, estratégias baseadas em questionamentos orientadores, inspiradas no método socrático (PAUL; ELDER, 2016), têm sido utilizadas para estimular o pensamento crítico e a reflexão dos estudantes, incentivando a construção ativa do conhecimento. Esse tipo de abordagem torna-se especialmente relevante em contextos mediados por tecnologias, nos quais a orientação pode ser estruturada por meio de interações guiadas.

No que se refere às ferramentas tecnológicas, Brezolin e Silveira (2021) apresentam uma visão abrangente do cenário brasileiro em relação ao uso dessas ferramentas no desenvolvimento do pensamento computacional e no ensino de programação. Os autores indicam que a maioria dos estudos analisados utiliza linguagens de programação visual com blocos como principal recurso, seguida por jogos, linguagens de programação textuais,

ambientes de desenvolvimento integrado (IDEs) visuais e kits de robótica como Mindstorms e Arduino.

No contexto da aplicação de metodologias pedagógicas e ferramentas tecnológicas no ensino de programação, Thinakaran e Ali (2015) ressaltam que a melhor forma de ensinar programação é por meio da interação individual entre professor e aluno. A importância dessa interação já havia sido enfatizada por Bloom (1984), em seu trabalho intitulado "*The Sigma-2 Problem*", que demonstrou melhorias significativas nos resultados de aprendizagem quando os alunos recebem atendimento individualizado. A assistência personalizada, seja por professores ou colegas mais experientes, desempenha um papel fundamental na assimilação de conteúdos complexos, como a programação. A orientação focada e o acompanhamento individualizado podem ter um impacto positivo profundo na compreensão e aplicação desses conceitos. No entanto, esse tipo de interação é difícil de ser implementado em turmas grandes, o que tem motivado pesquisadores a explorar novos recursos tecnológicos que apoiem o processo de ensino e aprendizagem.

Para atender à demanda por um ensino de programação mais personalizado e alinhado às necessidades individuais dos alunos, as pesquisas sobre o uso de Inteligência Artificial (IA) generativa na educação têm recebido crescente atenção. Esses estudos indicam que a IA pode possibilitar um processo de aprendizagem personalizado na resolução de exercícios e problemas de programação, fornecendo orientação e *feedback* de maneira semelhante à tutoria entre professor e aluno.

Atualmente, ferramentas de IA generativa fornecem soluções rápidas e, em sua maioria, precisas para os desafios enfrentados pelos indivíduos na programação (BECKER *et al.*, 2023; DENNY *et al.,* 2024; JALIL *et al.*, 2023; YILMAZ; KARAOGLAN YILMAZ, 2023; SUN *et al.*; 2024, PRATHER *et al.*, 2024).

De acordo com Chen *et. al* (2022), a IA generativa, especialmente os modelos de linguagem de grande escala (*Large Language Models* - LLMs), tem sido cada vez mais aplicados no campo da educação, abrindo novas possibilidades para o ensino e a aprendizagem. Sua utilização no ensino de programação apresenta diversas vantagens que podem facilitar o processo educacional. Entre as principais contribuições, destaca-se a capacidade de oferecer conteúdo personalizado e adaptativo, permitindo que os alunos progridam de acordo com seu próprio ritmo e atendendo às suas necessidades individuais (HOLMES; BIALIK; FADEL, 2023).

Além disso, a IA pode fornecer *feedback* imediato sobre erros e problemas no código, permitindo que os estudantes corrijam suas falhas de forma rápida e aprendam com suas experiências. Outro benefício relevante é a capacidade de apresentar exemplos de código e explicar conceitos de programação de maneira clara e concisa, tornando a aprendizagem mais acessível. A IA também possibilita a criação de experiências de aprendizagem mais envolventes, ao integrar tecnologias como ambientes de programação interativos, jogos educacionais e realidade virtual (SILVA JUNIOR *et al.*, 2023).

Com base nessas premissas, a IA se destaca como uma ferramenta valiosa para apoiar o aprendizado de programação. Quando utilizados como assistentes de aprendizagem, os LLMs oferecem vantagens importantes: estão sempre disponíveis para responder dúvidas, fornecem explicações detalhadas e adaptadas ao contexto, e são capazes de ajudar na resolução de problemas complexos com sugestões e orientações. Esses recursos são especialmente benéficos no contexto educacional, onde a assistência individualizada e o *feedback* imediato são fundamentais para uma aprendizagem eficaz (KASNECI *et al*., 2023; LO, 2023).

Entretanto, a eficácia desses modelos na aprendizagem de programação permanece pouco explorada. Yilmaz e Karaoglan Yilmaz (2023) apontam como lacuna a ausência de pesquisas suficientes que investiguem o uso de IA generativa nesse contexto. Embora as ferramentas com suporte de IA sejam consideradas promissoras para melhorar habilidades de pensamento computacional, autoeficácia e motivação, o número e a diversidade de estudos que analisam o impacto da IA generativa no ensino de programação ainda são limitados. Assim, a aplicação da IA nesse contexto encontra-se em estágios iniciais, sendo necessárias mais investigações para compreender seu potencial e formas de uso mais eficazes.

Estudos de Harsley *et al*. (2017) e Cho e Lim (2017) indicam que a colaboração, quando comparada ao esforço individual, tende a gerar melhores resultados, como maior retenção de conteúdo e desenvolvimento do pensamento crítico. Além disso, a colaboração estimula a criatividade na resolução de problemas e fortalece a persistência dos estudantes, acelerando o processo de aprendizagem. Essas interações ocorrem, em geral, entre pares humanos, como colegas, tutores e professores.

No entanto, em algumas situações, a interação entre pares (entre alunos ou com professores) pode ser limitada, difícil ou até inviável. Nesse contexto, a IA generativa pode ser utilizada como um par de programação, oferecendo suporte em tempo real, sugerindo soluções e explicando conceitos de forma personalizada. Dessa forma, seu uso pode contribuir para a criação de um ambiente de aprendizagem mais adaptativo e contínuo, promovendo a

autonomia dos estudantes e complementando a mediação humana no processo de ensino.

Conforme destacado por Banić, Konecki e Konecki (2023), ferramentas de IA generativa podem atuar como pares virtuais de programação, auxiliando na resolução de problemas, geração de código, correção de erros e exploração de ideias. Além disso, favorecem o aprendizado colaborativo e tornam o processo de desenvolvimento mais eficiente e produtivo.

Diante do exposto, observa-se a existência de diversas estratégias e metodologias voltadas ao ensino de programação. No entanto, com base na experiência docente da autora desta tese e na literatura analisada, ainda há muitas lacunas a serem exploradas, especialmente quanto à identificação das dificuldades enfrentadas por alunos e professores, bem como na investigação de possíveis soluções para minimizá-las. Nesse contexto, a presente pesquisa propõe investigar o uso de IA generativa como ferramenta de apoio à aprendizagem de programação.

Para contribuir com esse processo, esta tese incorpora dois elementos centrais: a programação baseada em pares e ferramentas de IA generativa. A programação em pares (*pair programming*) promove o trabalho colaborativo e a resolução conjunta de problemas. As ferramentas de IA, por sua vez, possibilitam uma aprendizagem personalizada e adaptativa, permitindo que os estudantes progridam de acordo com seu próprio ritmo e atendam às suas necessidades individuais, além de atuar como um potencial par de programação virtual.

1.1 Questão de Pesquisa

Considerando a importância da programação de computadores e as dificuldades inerentes ao seu ensino e aprendizagem, torna-se relevante investigar e desenvolver recursos tecnológicos que possam contribuir para a melhoria desse processo, especialmente diante do fato que o insucesso nessas disciplinas é apontado como uma das principais causas do elevado índice de evasão dos cursos de computação.

A adoção de tecnologias emergentes no ensino e na aprendizagem apresenta-se como uma alternativa promissora para engajar e motivar os alunos, além de potencializar a eficácia do ensino de programação. Dentre essas tecnologias, destaca-se o uso de ferramentas de IA generativa, que, quando utilizadas de forma apropriada, podem enriquecer significativamente esse processo.

Assim, com base na contextualização, problemática, justificativa e motivação apresentadas, busca-se responder à seguinte questão de pesquisa:

"De que forma o uso de uma ferramenta de IA generativa, atuando como um par de programação, contribui para a eficácia do aprendizado e para a motivação dos alunos na aprendizagem de programação?"

A hipótese que sustenta esta investigação é que o uso de uma ferramenta de IA generativa, concebida como par de programação e orientada por princípios pedagógicos, favorece a aprendizagem de programação ao oferecer suporte individualizado, *feedback* contextualizado e mediação durante a resolução de problemas. Espera-se, assim, que os estudantes que utilizam o AIPairCode apresentem maior ganho de aprendizagem, percepções positivas de motivação e boa aceitação da ferramenta em comparação aos estudantes que realizam as atividades sem esse apoio.

1.2 Objetivos

1.2.1 Objetivo Geral

Compreender de que maneira o uso de uma ferramenta de Inteligência Artificial generativa, atuando como um par de programação, impacta a eficácia do aprendizado de alunos em disciplinas de programação, considerando seus efeitos na motivação e na interação ao longo do processo de aprendizagem.

O estudo analisa como a interação com a IA pode melhorar a compreensão de conceitos, auxiliar na resolução de problemas de programação e influenciar positivamente a percepção dos alunos em relação à programação.

1.2.2 Objetivos Específicos

Os objetivos necessários para atingir o objetivo geral, estão elencados a seguir:

- Realizar uma revisão de literatura para identificar os principais problemas e desafios no ensino e na aprendizagem de programação, bem como as oportunidades para o uso de ferramentas de IA generativa nesse contexto;
- Desenvolver uma ferramenta de IA generativa personalizada para atuar como um par de programação, oferecendo suporte individualizado, orientações e *feedback* durante o processo de aprendizagem;

- Validar a proposta com estudantes de computação, utilizando a IA como par de programação em atividades práticas de resolução de problemas e desenvolvimento de código;
- Investigar o processo de aprendizagem dos estudantes com o uso da ferramenta de IA, por meio da análise das interações registradas, identificando padrões de apoio, dificuldades e estratégias adotadas durante a resolução de problemas;
- Avaliar a eficácia do aprendizado dos estudantes por meio da aplicação de pré-teste e pós-teste, analisando o desempenho, a compreensão de conceitos e a evolução ao longo do período de intervenção;
- Examinar os efeitos do uso da IA na motivação dos alunos, considerando aspectos como interesse, utilidade percebida, percepção de autonomia e competência durante as atividades de programação;
- Coletar e analisar o *feedback* dos alunos sobre sua experiência com a IA como par de programação, identificando pontos positivos, limitações e oportunidades para o aprimoramento da ferramenta e da abordagem.

1.3 Estrutura da Tese

Esta tese está organizada em capítulos, conforme descrito a seguir.

O Capítulo 1 apresenta a introdução ao tema trabalhado, bem como a problemática e a justificativa do estudo, seguidas da questão de pesquisa, dos objetivos gerais e específicos e da estrutura do documento.

O Capítulo 2 apresenta o referencial teórico que embasa esta pesquisa. A primeira seção aborda o ensino de programação de computadores, apresentando suas diretrizes curriculares, dificuldades e principais estratégias utilizadas. Na segunda seção, são apresentadas as teorias de aprendizagem utilizadas neste estudo, as metodologias ativas e a programação baseada em pares. A terceira seção apresenta os conceitos de Inteligência Artificial Generativa e seu uso na educação.

O Capítulo 3 apresenta a revisão sistemática da literatura e os trabalhos correlatos.

No Capítulo 4 é descrita a metodologia da pesquisa, incluindo sua caracterização e delineamento, os procedimentos, a ambientação e o público-alvo, a condução da pesquisa, os

instrumentos de coleta e análise de dados e as etapas do estudo.

O Capítulo 5 apresenta o desenvolvimento do assistente de programação proposto nesta pesquisa, incluindo sua concepção pedagógica, os fundamentos teóricos adotados, a arquitetura do sistema e a descrição de seus principais módulos e funcionalidades.

O Capítulo 6 detalha os estudos empíricos realizados e os resultados obtidos a partir da aplicação da proposta, contemplando a análise dos dados coletados, incluindo o desempenho dos estudantes, a motivação e a análise das interações com a ferramenta, bem como a discussão dos achados.

Por fim, o Capítulo 7 apresenta as considerações finais da pesquisa, destacando as principais contribuições do estudo, suas limitações e sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta uma revisão bibliográfica sobre os principais tópicos abordados nesta tese, com foco no ensino de programação, nas metodologias pedagógicas e no uso de Inteligência Artificial Generativa na Educação.

2.1 Ensino de Programação de Computadores

O objetivo do ensino de programação é estimular e desenvolver nos alunos as habilidades e competências necessárias para serem capazes de desenvolver soluções computacionais para resolver problemas complexos do mundo real. As habilidades abrangem codificação (a capacidade de escrever código de computador usando linguagens de programação específicas para criar programas e soluções), resolução de problemas, pensamento lógico, depuração e abstração.

No documento que apresenta as diretrizes de formação para cursos de graduação em computação, elaborado pela Sociedade Brasileira de Computação (SBC) e específico para os cursos de Ciência da Computação e Engenharia de Computação (ZORZO *et al*., 2017), a programação é identificada como uma disciplina central nos fundamentos da computação, abrangendo tanto aspectos científicos quanto técnicas fundamentais para uma formação sólida dos graduandos. O guia de referência da ACM e IEEE para cursos de Ciência da Computação (CC2020, 2020), ressalta que os alunos devem abordar a computação não apenas como o aprendizado de linguagens de programação, mas compreender os conceitos mais amplos da computação, como abstração e decomposição, dentro do contexto do aprendizado da programação.

Com base nesse raciocínio, percebe-se que o ensino de programação vai além do estudo de linguagens específicas e abrange uma variedade de princípios, técnicas e formalidades destinadas a capacitar os alunos no desenvolvimento de programas de computador. De acordo com Koorsse, Cilliers e Calitz (2015), o processo de desenvolvimento de programas de computador envolve três etapas distintas:

1. Compreender o problema e formular uma solução: é essencial examinar cuidadosamente a descrição do problema e os requisitos associados, e depois decidir qual a estratégia mais adequada a ser adotada;

2. Criar um algoritmo para resolver o problema: essa etapa requer um entendimento sólido dos princípios e conceitos fundamentais de programação, para serem selecionados e aplicados de maneira eficaz;
3. Transpor o algoritmo para uma linguagem de programação: essa etapa demanda familiaridade com a sintaxe específica da linguagem de programação escolhida. O código do programa é então testado e modificado até que atenda aos requisitos originais e resolva o problema proposto.

Araújo e Falkembach (2006) explicam que a criação de um algoritmo requer compreender o enunciado, identificar as informações-chave e, a partir disso, desenvolver estratégias para resolver o problema. No entanto, as estratégias podem variar consideravelmente, dependendo do pensamento de cada estudante. Em termos gerais, a programação é vista como um método para resolver problemas, englobando desde a compreensão inicial do problema até sua resolução final (AMBRÓSIO *et al.*, 2011). Alguns autores argumentam que, devido à sua complexidade, o processo de programação pode ser dividido em fases ou etapas distintas.

A construção de um algoritmo computacional, ou a programação de um software, vai além da simples elaboração de instruções. Segundo Farrel (2010), o processo de desenvolvimento de um programa pode ser dividido em seis etapas: 1. Compreensão do problema; 2. Planejamento da lógica; 3. Codificação do programa; 4. Compilação do código para a linguagem de máquina; 5. Execução e testes do programa codificado; e 6. Disponibilização do programa para uso.

Ambrósio *et al.* (2011) destacam quatro fases: compreensão do problema, planejamento da solução, implementação e verificação. Tavares (2018) divide a tarefa em cinco etapas: análise do problema, especificação dos requisitos e restrições, desenvolvimento do algoritmo, codificação e teste. Por sua vez, Lemos (2004) sugere cinco passos: identificação dos objetivos ou metas, seleção de planos de programação, escolha de ações para cada plano, decomposição em subplanos e codificação de cada subplano em uma linguagem de programação.

Morais (2022) compila essas partes apresentadas por diferentes autores em 5 etapas: 1) interpretar e compreender o problema, verificando seus objetivos e metas; 2) especificar os requisitos necessários para resolver o problema; 3) implementar um plano de resolução desenvolvendo um algoritmo com base nos requisitos especificados; 4) codificar o algoritmo

usando uma linguagem de programação adequada para o tipo de problema; 5) executar e testar o programa para verificar se está correto. Essas etapas são complementares e seguem uma ordem lógica, sendo complexas, pois envolvem uma série de habilidades inter-relacionadas, diferentes áreas de conhecimento e processos cognitivos. No entanto, o processo não é rigidamente linear, pois os programadores podem identificar falhas ou inconsistências em qualquer etapa anterior ao desenvolvimento do software. Assim, a resolução de problemas na programação é iterativa, permitindo que os desenvolvedores retrocedam para etapas anteriores sem seguir uma sequência fixa, conforme ilustrado na Figura 1 (MORAIS, 2022).

Figura 1 - Etapas para resolução de problemas com programação



Fonte: Morais (2022).

Neste contexto, aprender a programar envolve mais do que apenas dominar uma linguagem específica, pois requer também a aplicação de diversos conhecimentos e a rigorosa execução de diferentes etapas para resolver problemas. Segundo Medeiros, Ramalho e Falcão (2019), programar é essencialmente dar instruções para que um computador execute determinadas atividades, demandando o conhecimento e a escolha adequada da linguagem de programação. Tavares (2018) ressalta que a programação visa resolver problemas computacionalmente, preparando o computador para transformar dados em resultados que solucionem o problema em questão, independentemente da linguagem utilizada. Assim, programar ultrapassa a simples aprendizagem de sintaxe e semântica de uma linguagem; requer habilidades e conhecimentos para compreender e resolver problemas utilizando a linguagem de programação como ferramenta.

2.1.1 Diretrizes Curriculares para Cursos de Computação no Brasil

Para compreender o processo de ensino e aprendizagem dos cursos de graduação em Computação no Brasil, deve-se analisar, primeiramente, as Diretrizes Curriculares Nacionais (DCNs), estabelecidas pela Câmara de Educação Superior do Conselho Nacional de Educação (CES/CNE), pois são elas que definem os princípios, os fundamentos, as condições e as finalidades para a aplicação, organização, desenvolvimento e avaliação do curso de graduação em Computação das Instituições de Educação Superior (IES) (MEC, 2016).

A resolução CNE/CES n.º 5 de 16 de novembro de 2016, institui as Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação, abrangendo os cursos de bacharelado em Ciência da Computação, em Sistemas de Informação, em Engenharia de Computação, em Engenharia de Software e de licenciatura em Computação. Este documento define as diretrizes que as IES devem considerar na organização, no desenvolvimento e na avaliação, no âmbito dos Sistemas de Educação Superior do país (MEC, 2016). Para identificar o perfil dos profissionais formados em Computação, o Art. 4º deste documento, apresenta as seguintes características, entre outras:

> Os cursos de bacharelado e de licenciatura da área de Computação devem assegurar a formação de profissionais dotados:
> I - de conhecimento das questões sociais, profissionais, legais, éticas, políticas e humanísticas;
> II - da compreensão do impacto da computação e suas tecnologias na sociedade no que concerne ao atendimento e à antecipação estratégica das necessidades da sociedade;
> III - de visão crítica e criativa na identificação e resolução de problemas, contribuindo para o desenvolvimento de sua área;
> IV - da capacidade de atuar de forma empreendedora, abrangente e cooperativa no atendimento às demandas sociais da região onde atua, do Brasil e do mundo;
> V - de utilizar racionalmente os recursos disponíveis de forma transdisciplinar;
> VI - da compreensão das necessidades da contínua atualização e aprimoramento de suas competências e habilidades;
> VII - da capacidade de reconhecer a importância do pensamento computacional na vida cotidiana, como também sua aplicação em outros domínios e ser capaz de aplicá-lo em circunstâncias apropriadas; e
> VIII - da capacidade de atuar em um mundo de trabalho globalizado.

No Art. 5º deste mesmo documento são identificadas as competências e habilidades comuns para os egressos dos cursos de bacharelado e licenciatura da área de Computação (MEC, 2016):

> I - identificar problemas que tenham solução algorítmica;
> II - conhecer os limites da computação;
> III - resolver problemas usando ambientes de programação;
> IV - tomar decisões e inovar, com base no conhecimento do funcionamento e das características técnicas de hardware e da infraestrutura de software dos sistemas de

> computação, consciente dos aspectos éticos, legais e dos impactos ambientais decorrentes;
> V - compreender e explicar as dimensões quantitativas de um problema;
> VI - gerir a sua própria aprendizagem e desenvolvimento, incluindo a gestão de tempo e competências organizacionais;
> VII - preparar e apresentar seus trabalhos e problemas técnicos e suas soluções para audiências diversas, em formatos apropriados (oral e escrito);
> VIII - avaliar criticamente projetos de sistemas de computação;
> IX - adequar-se rapidamente às mudanças tecnológicas e aos novos ambientes de trabalho;
> X - ler textos técnicos na língua inglesa;
> XI - empreender e exercer liderança, coordenação e supervisão na sua área de atuação profissional;
> XII - ser capaz de realizar trabalho cooperativo e entender os benefícios que este pode produzir.

Além disso, no intuito de orientar o ensino de computação no Brasil, a Sociedade Brasileira de Computação (SBC) tem desempenhado um papel fundamental ao longo das últimas décadas. A SBC tem promovido discussões sobre a condução dos cursos de graduação, seja por meio da constituição de comissões para elaboração de currículos de referência, ou ao debater as formas de avaliação desses cursos junto ao Ministério da Educação. Como resultado, foram elaborados os "Referenciais de Formação em Computação" (RFs) para os cursos estabelecidos nas DCNs: Ciência da Computação, Engenharia de Computação, Engenharia de Software, Licenciatura em Computação e Sistemas de Informação. Os RFs constituem um conjunto de documentos que auxiliam os coordenadores de curso de graduação na elaboração dos projetos pedagógicos (ZORZO *et al*., 2017).

Com base nas DCNs e nos RFs, as atividades de aprendizagem nos cursos de computação devem contemplar e garantir o desenvolvimento das competências do perfil do egresso. De acordo com Zorzo *et al.* (2017), as principais vantagens de utilizar uma abordagem por competências são: sua capacidade comprovada de atribuir significado aos conteúdos que compõem o currículo, a ampliação do currículo para incluir habilidades e atitudes, além de conhecimento, e uma maior aderência ao perfil do egresso esperado pelo curso.

Entre as diversas definições de competência encontradas na literatura, Zorzo *et al*. (2017) propuseram, para a elaboração dos RFs, um modelo de referência baseado na Taxonomia de Bloom Revisada (FERRAZ; BELHOT, 2010). Nesse modelo, uma competência reflete os conhecimentos, as habilidades ou as atitudes esperadas do egresso do curso, sob a perspectiva de objetivos de aprendizagem (aquilo que o aluno será capaz de fazer). Dessa forma, um conteúdo associado a determinada competência dos cursos da área de computação está vinculado a um ou mais desses objetivos.

Por exemplo, um egresso de computação tem a competência de desenvolver sistemas computacionais. Para isso, a sua formação deverá contemplar, dentre outros, os seguintes conteúdos: Métodos Formais, Algoritmos, Técnicas de Programação e Ética. Cada um desses conteúdos proporcionará ao aluno parte dos conhecimentos e das habilidades necessários para desenvolver sistemas computacionais (o saber fazer), além de influenciar a forma de aplicar tais conhecimentos e habilidades (o saber ser) (ZORZO *et al*., 2017).

Além disso, a articulação e estruturação das competências nos RFs foram organizadas em eixos temáticos de formação. As competências e habilidades, tanto gerais quanto específicas, estabelecidas pelas DCNs para os egressos dos cursos de computação foram agrupadas nesses eixos. Cada eixo de formação para os RFs corresponde a uma macrocompetência e relaciona um conjunto de competências derivadas (competências e habilidades previstas nas DCNs) que, se desenvolvidas conjuntamente, capacitarão o estudante a atingir a competência central do eixo.

Em conjunto, esses eixos permitem ao egresso a lidar profissionalmente com as múltiplas características das atividades de computação. Os eixos de formação refletem o entendimento de que a formação em computação deve considerar: a capacidade de atuar em todas as fases que envolvem a aplicação da ciência da computação para a solução de problemas diversos, desde a concepção de sistemas computacionais até a implementação de soluções adequadas; a habilidade de se atualizar continuamente e buscar novos conhecimentos; e a competência para seguir estudos avançados visando ao progresso da ciência e da tecnologia (ZORZO *et al.*, 2017).

Para atingir esses resultados, é essencial o uso de metodologias que promovam uma visão crítica e criativa na identificação e resolução de problemas, a autonomia do estudante, o trabalho interdisciplinar e colaborativo, e o desenvolvimento do pensamento computacional com aplicação em outros domínios.

Os conteúdos de Algoritmos e Programação permeiam os diferentes eixos de formação descritos nas RFs e nas DCNs dos cursos de computação, destacando-se como componentes essenciais dos cursos desta área. Sua relevância está na preparação dos alunos para as demais disciplinas relacionadas à programação dentro do currículo. As disciplinas introdutórias de programação formam a base para o desenvolvimento lógico e algorítmico dos estudantes, sendo indispensáveis para estabelecer os fundamentos necessários à compreensão de conceitos mais avançados (FRANÇA; TEDESCO, 2015). O objetivo é capacitar os alunos a adquirirem os conhecimentos essenciais e as habilidades necessárias para desenvolver

programas que solucionem problemas do mundo real (MEDEIROS; RAMALHO; FALCÃO, 2019).

Embora a instrução em programação seja predominantemente associada aos cursos de computação, como mencionado anteriormente, observa-se uma crescente inclusão dessa disciplina, mesmo que de forma extracurricular, em uma variedade de cursos superiores e também no Ensino Médio e Fundamental. O objetivo dessa inclusão é proporcionar aos estudantes uma compreensão inicial de programação que contribuirá para sua formação geral, além de promover a familiarização com linguagens de programação e incentivar a resolução de problemas práticos, contextualizados no âmbito de seus respectivos cursos.

2.1.2 Dificuldades Inerentes à Aprendizagem de Programação

Aprender lógica de programação é um dos pilares da formação dos profissionais de computação, sendo destacada pela Sociedade Brasileira de Computação como essencial nos cursos superiores da área (ZORZO *et al.*, 2017). As Diretrizes Curriculares Nacionais para área de computação reforçam a importância de uma base sólida em conteúdos relacionados ao raciocínio lógico e à programação de computadores (MEC, 2016). As disciplinas introdutórias que abordam essas habilidades são fundamentais no início do curso e influenciam diretamente na progressão no currículo.

No entanto, o processo de ensino e aprendizagem de programação é bastante complexo, envolvendo diversos desafios enfrentados por alunos e professores. Muitos estudantes encontram dificuldades significativas no desenvolvimento das habilidades necessárias para programar, o que contribui para os altos índices de reprovação e evasão nas disciplinas relacionadas, resultando em elevados índices de abandono nos cursos de Computação (ALVIM; BITTENCOURT; DURAN, 2024; BENNEDSEN; CASPERSEN, 2019).

Buscando quantificar o problema, Bennedsen e Caspersen (2007) realizaram uma pesquisa em 63 instituições internacionais, coletando dados sobre o número de alunos que abandonaram, não concluíram o exame final ou reprovaram em disciplinas introdutórias de programação. Os autores observaram uma taxa média de insucesso de cerca de 33%, frequentemente ultrapassando os 50% em algumas instituições. Em um estudo subsequente conduzido por Watson e Li (2014), ao analisarem dados de 51 instituições, encontraram resultados semelhantes, com a mesma taxa média de falhas e grande variação entre as turmas investigadas. Os autores também identificaram que essas taxas variavam conforme os países

analisados, mas não em função da linguagem de programação utilizada. Ambos os estudos destacaram, ainda, uma correlação entre turmas menores e índices de reprovação mais baixos.

No contexto brasileiro, Bosse e Gerosa (2015) examinaram os resultados de disciplinas introdutórias de programação na Universidade de São Paulo (USP), entre 2010 a 2014, identificando uma taxa média de reprovação de 30%, com diversos casos superiores a 50%. Além disso, essas taxas permaneceram estáveis ao longo dos anos analisados, e mais de 25% dos estudantes aprovados nesse período precisaram cursar a disciplina duas ou mais vezes. A partir de microdados do INEP, Alvim, Bittencourt e Duran (2024) verificaram que, no período de 2015 a 2019, a taxa de evasão nos cursos da área no Brasil alcançou 19,3%, considerando os casos de desligamento ou transferência, e 32,1%, quando incluídas também as matrículas trancadas.

Em complemento, Simon *et al*. (2019) indicam que as taxas de insucesso em disciplinas introdutórias de programação giram em torno de 30%, o que faz com que essas disciplinas sejam consideradas um dos principais desafios nos cursos de computação, informática e outros que incluem essas disciplinas em sua matriz curricular, dificultando ou até mesmo impedindo a continuidade dos estudantes nos cursos.

No contexto específico de programação e algoritmos, é importante abordar aspectos que estão diretamente relacionados às causas das dificuldades de aprendizagem enfrentadas pelos alunos e que influenciam os altos índices de evasão. Segundo Bosse (2020), um dos desafios no processo de aprendizagem de programação é a quantidade de novos conceitos e práticas exigidos, o que pode dificultar o aprendizado dos iniciantes na área. Para muitos alunos em disciplinas introdutórias de programação, os conceitos e habilidades necessários para programar têm pouca relação com suas experiências educacionais anteriores. Fatores como habilidades cognitivas, estilos de aprendizagem, motivação e atitudes têm sido identificados como contribuintes significativos para os altos índices de reprovação e evasão. Aprender programação requer o domínio de conhecimentos específicos, estratégias e modelos mentais, além de habilidades como abstração, raciocínio lógico e compreensão da sintaxe e semântica das linguagens de programação.

Tentando identificar as características e dificuldades nesse processo, Medeiros, Ramalho e Falcão (2019) apresentam uma revisão sistemática da literatura sobre o ensino e aprendizagem de programação introdutória no ensino superior, identificando habilidades essenciais e dificuldades enfrentadas pelos educadores e alunos. Segundo o estudo, as principais habilidades consideradas necessárias pelos educadores incluem raciocínio lógico

(mencionado em 24% dos 84 artigos analisados), abstração (mencionado em 15% dos artigos), noções matemáticas (mencionado em 14% dos artigos) e resolução de problemas (mencionado em 13% dos artigos). Em relação aos estudantes, os artigos indicam que as maiores dificuldades são: resolução de problemas (25%); abstração (21%), e raciocínio lógico e algorítmico (12%). É possível perceber a dificuldade inerente à abstração e construção do raciocínio lógico dos estudantes, especialmente no que tange à forma de compreensão de esquemas algorítmicos.

Para melhorar os resultados no ensino de programação, é essencial adaptar os métodos atuais para tornar o processo mais atraente e eficaz. A classificação e organização dessas dificuldades são valiosas para orientar professores na identificação de áreas que requerem maior atenção em sala de aula, bem como para auxiliar pesquisadores no desenvolvimento de novas ferramentas educacionais e abordagens de ensino.

De acordo com Gomes e Mendes (2014), as dificuldades dos alunos em programação podem ser agrupadas em três categorias: estratégias de ensino, atitudes dos alunos e métodos de estudo, e dificuldades inerentes à própria programação. De forma semelhante, Kadar *et al*. (2021), em sua a revisão sistemática da literatura, identificaram que os principais desafios no ensino e na aprendizagem de programação podem ser divididos em três grupos: dificuldades relacionadas à linguagem de programação, dificuldades enfrentadas pelos alunos ao aprender uma linguagem e desafios enfrentados pelos educadores ao ensiná-la.

Entre os desafios relacionados à linguagem de programação, destacam-se dificuldades na compreensão da estrutura básica da linguagem, na criação de programas completos para resolver problemas específicos, na identificação da sintaxe e na correção de erros no código. A identificação e correção de erros (*bugs*) representam obstáculos significativos para iniciantes, geralmente associados à falta de compreensão dos conceitos fundamentais, à complexidade na elaboração de algoritmos e à dificuldade em traduzi-los para uma linguagem específica (KADAR *et al.*, 2021).

Em relação às dificuldades enfrentadas pelos alunos, Kadar *et al*. (2021) aponta três aspectos principais. O primeiro refere-se à compreensão de conceitos básicos, como estruturas de controle, estruturas de dados e algoritmos, o que compromete sua aplicação na resolução de problemas. O segundo envolve a elaboração de programas completos, especialmente na decomposição do problema em etapas menores, na organização do código e na implementação de soluções funcionais. O terceiro diz respeito à sintaxe das linguagens, incluindo erros frequentes, como ausência de declaração de variáveis, uso inadequado de

expressões *booleanas* e confusão entre operadores de atribuição e comparação.

Essas dificuldades ressaltam a importância de fornecer suporte adequado aos estudantes, por meio de aulas práticas, exemplos claros, *feedback* construtivo e oportunidades frequentes para praticar e aplicar os conceitos aprendidos.

Kadar *et al*. (2021) também destaca que, em relação aos desafios enfrentados pelos educadores ao ensinar linguagens de programação, podem ser citados:

- Dificuldade em transmitir os conceitos de programação de forma clara e compreensível, garantindo que os alunos compreendam os fundamentos;
- Necessidade de adaptar as estratégias de ensino às diferentes necessidades dos alunos. Cada aluno possui um estilo de aprendizagem único e diferentes níveis de habilidade em programação, o que exige que os educadores adaptem suas abordagens para garantir que todos acompanhem o ritmo das aulas;
- Desafios na avaliação e *feedback* dos alunos, especialmente ao identificar as áreas onde os alunos encontram dificuldades e ao oferecer orientação personalizada para melhorar o desempenho acadêmico.

Diante do exposto, é possível perceber que os aprendizes de programação enfrentam diversos obstáculos. Para superar esses desafios, os educadores podem se beneficiar do uso de estratégias de ensino inovadoras, como aulas práticas, estudos de caso, tutoriais interativos e feedback individualizado. Além disso, a colaboração entre estudantes e a implementação de abordagens pedagógicas eficazes podem contribuir para a melhoria da qualidade do ensino de linguagens de programação, promovendo um ambiente de aprendizagem mais eficiente e envolvente.

2.2 Teorias de Aprendizagem Aplicadas ao Ensino de Programação

O ensino de programação, especialmente em níveis iniciais, apresenta desafios que vão além da simples transmissão de conteúdo técnico, exigindo esforço e uma abordagem diferenciada na forma como é aprendido e ensinado, conforme discutido na seção anterior. Nesse sentido, conforme Silva e Tarouco (2022), uma estratégia utilizada tanto para apoiar estudantes universitários iniciantes na exploração desses conceitos, quanto para promover o desenvolvimento do pensamento computacional em alunos da Educação Básica, é a combinação de ferramentas tecnológicas com metodologias educacionais adequadas. Essa abordagem visa promover uma interação mais envolvente dos estudantes com o processo de

aprendizagem.

Para que os alunos desenvolvam competências de pensamento computacional e resolução de problemas, é fundamental considerar as bases teóricas que sustentam o processo de aprendizagem. Nesse contexto, as teorias de aprendizagem oferecem um referencial essencial para entender como os alunos constroem conhecimento e como podem ser melhores apoiadas ao longo desse percurso. Entre as abordagens utilizadas como base para esta tese está a Teoria Sociocultural de Vygotsky, com o conceito de zona de desenvolvimento proximal (ZDP) e mediação, o método socrático, bem como as metodologias ativas de ensino como a aprendizagem baseada em problemas e projetos e a programação em pares. Essas teorias destacam a importância da interação, do suporte adequado e da resolução de problemas no desenvolvimento de habilidades cognitivas, tornando-as essenciais para fundamentar novas abordagens no ensino de programação, como o uso de Inteligência Artificial.

2.2.1 Teoria Sociocultural de Vygotsky (ZDP e Mediação)

As concepções de educação e técnicas de aprendizagem adotadas nesta pesquisa são fundamentadas na teoria sociocultural de Vygotsky (1978), que enfatiza o papel central da linguagem, da cultura e do ambiente social na formação do sujeito. Segundo essa teoria, a aprendizagem ocorre por meio da participação em atividades culturais, onde os estudantes orientados por indivíduos mais experientes interiorizam os instrumentos necessários para pensar e resolver problemas de forma mais avançada do que conseguiriam sozinhos.

Vygotsky defende que o aprendizado é essencialmente um processo social, no qual o aluno é guiado por um tutor, professor ou parceiro mais experiente, que o auxilia a desenvolver habilidades e conhecimentos dentro da sua zona de desenvolvimento proximal (ZDP). A ZDP é definida como a distância entre o que o aprendiz consegue realizar de forma autônoma e o que ele é capaz de alcançar com apoio, destacando o papel da interação social e da mediação na construção do conhecimento.

> [...] a distância entre o nível de desenvolvimento real, que se costuma determinar através da solução independente de problemas, e o nível de desenvolvimento potencial, determinado através da solução de problemas sob a orientação de um adulto ou em colaboração com companheiros mais capazes (Vygotsky, 1989, p. 97).

A teoria da ZDP de Vygotsky foi desenvolvida para explicar a relação entre aprendizagem e desenvolvimento, especialmente nas crianças. Ao contrário de teorias que

sugerem que o desenvolvimento deveria preceder ou ocorrer simultaneamente à aprendizagem, Vygotsky propôs que a aprendizagem é essencial para impulsionar o desenvolvimento. Assim, o nível de desenvolvimento de um indivíduo pode ser avaliado não apenas observando suas habilidades atuais, mas também identificando o que ele está pronto para aprender com a devida assistência. A ZDP representa o intervalo entre esses dois níveis, descrevendo as funções que estão "em processo de maturação". Experiências de aprendizagem voltadas para a ZDP são as mais eficazes. Vygotsky argumenta que não há benefício em oferecer instrução em áreas onde o domínio já foi alcançado, nem é possível que os indivíduos aprendam conteúdos muito além de sua ZDP (VYGOTSKY, 1989).

O ambiente sociocultural onde a aprendizagem ocorre é um aspecto central da teoria de Vygotsky. Desde o nascimento, os indivíduos aprendem naturalmente por meio da interação com outros, o que impulsiona seu desenvolvimento inicial. Na escola, a instrução é oferecida de forma mais sistemática e proporciona oportunidades adicionais para aprender em colaboração com os colegas. Ao trabalhar em grupo, as crianças podem alcançar resultados que, individualmente, só seriam possíveis com o auxílio de um professor. Dessa forma, tanto as interações entre pares quanto a mediação de adultos fomentam o desenvolvimento, permitindo que a criança se torne progressivamente mais independente e amplie sua ZDP. Os principais conceitos dessa teoria são sintetizados na seguinte passagem dos escritos de Vygotsky:

> Propomos que um aspecto essencial do aprendizado é o fato de ele criar a zona de desenvolvimento proximal; ou seja, o aprendizado desperta vários processos internos de desenvolvimento, que são capazes de operar apenas quando a criança interage com pessoas em seu ambiente e quando em cooperação com seus companheiros. Uma vez internalizados, esses processos tornam-se parte das aquisições do desenvolvimento independente da criança. (Vygotsky, 1989, p. 101).

As ideias centrais de Vygotsky sobre o ensino incluem a ZDP, a internalização das funções psicológicas superiores e a mediação. A ZDP define o potencial de desenvolvimento de um indivíduo, considerando tanto o que ele já é capaz de realizar de forma independente quanto o que pode alcançar com assistência. A internalização das funções psicológicas superiores explica como o desenvolvimento é impulsionado pela aprendizagem, ocorrendo de fora para dentro, à medida que processos sociais são progressivamente interiorizados. A mediação, por sua vez, refere-se ao processo de interação entre o indivíduo e o ambiente sociocultural, por meio do qual ocorre essa internalização. Esses conceitos são fundamentais para o processo de ensino-aprendizagem, pois o educador precisa reconhecer o que o estudante já domina e identificar aquilo que ele ainda pode atingir, com base na mediação

oferecida pelo professor, pelos colegas, pelos recursos utilizados ou pela reflexão sobre experiências anteriores (PREUSS, 2021).

A mediação pode ser compreendida como uma intervenção que atua como intermediária entre o indivíduo e o ambiente ao seu redor. Quando essa intervenção é realizada por outra pessoa, como educadores, colegas mais experientes ou adultos, denomina-se mediação pedagógica. Já quando é feita por meio de signos, sendo a linguagem o mais importante, chama-se mediação semiótica. Essas formas de mediação são interdependentes e podem ocorrer simultaneamente, sem serem exclusivas entre si. Uma intervenção eficaz facilita a interação entre o sujeito e o objeto de conhecimento, promovendo a construção de significado através da criação e uso de signos, o que contribui para o desenvolvimento do aprendiz e favorece o processo de aprendizagem (COSTA, 2006).

A mediação pode ser representada como um processo triádico no esquema (S → X → R), em que "X" é o elemento intermediário que conecta o estímulo (S) à resposta (R). Esse esquema envolve o sujeito da aprendizagem (como um estudante), o objeto de conhecimento (conteúdos específicos) e um mediador, que pode ser um educador ou uma ferramenta. A mediação não se dá como uma simples relação de estímulo resposta, mas como um processo mais elaborado, mediado por instrumentos ou intervenções. Dessa forma, o papel do elemento intermediário altera a interação direta entre o estímulo e a resposta, tornando-a mediada (VYGOTSKY, 1989).

O conhecimento é construído a partir das interações sociais e das condições culturais, sociais e históricas em que o indivíduo está inserido. A forma de pensar e compreender o mundo é moldada pela apropriação do saber compartilhado pela comunidade. Nesse processo, a mediação por meio de signos e instrumentos é fundamental, auxiliando tanto nas ações práticas quanto nos processos psicológicos de aprendizagem. Essas formas de mediação permitem que o indivíduo realize operações cada vez mais complexas sobre os objetos de estudo (VYGOTSKY, 1989).

A interação, a mediação e o uso de instrumentos de mediação ocupam um papel central na teoria de Vygotsky e influenciam diversas outras correntes educacionais que surgiram posteriormente. Na mediação pedagógica, o educador, os pares mais experientes, os recursos metodológicos e os materiais utilizados são essenciais para o desenvolvimento de construtos cognitivos que facilitam a aprendizagem. Além disso, é importante reconhecer o papel das tecnologias, como computadores e recursos multimídia, no processo de ensino e aprendizagem. Ambientes computacionais, quando bem integrados, atuam como ferramentas

poderosas que potencializam a interação entre o aprendiz e o objeto de conhecimento, promovendo a construção ativa do saber (PASSERINO; SANTAROSA, 2000).

Como princípio fundamental de uma pedagogia de base vygotskiana, qualquer conteúdo, seja de ciências humanas, exatas ou biológicas, pode ser ensinado e aprendido por meio de diferentes estratégias pedagógicas, desde que estas promovam interações sociais, nas quais o professor ou outro parceiro mais experiente, que domine o conteúdo, atue como mediador. No entanto, a natureza dessas interações pode variar conforme a disciplina, já que cada área do conhecimento possui suas próprias características e demanda abordagens específicas.

No ensino de programação, é necessário que as interações considerem as dificuldades típicas de alunos em estágio inicial de aprendizagem. Do ponto de vista de uma teoria de inspiração vygotskiana, a discussão das atividades realizadas pelo aluno no processo de aprendizagem de programação é útil, e pode produzir excelentes resultados, desde que adequadamente guiada pelo professor ou por um par de programação mais experiente. Dessa forma, o par de programação pode realizar o papel de mediador, discutindo com o estudante os resultados das tarefas de programação por ele realizadas, buscando auxiliá-lo a realizar um diagnóstico dos eventuais erros cometidos, provendo recomendações adaptativas para cada aluno. Espera-se que a mediação exercida pelo par de programação contribua para ampliar a ZDP do estudante e com isto melhorar suas condições de aprender programação.

2.2.2 Método Socrático

O método socrático fundamenta-se no questionamento dialógico, conduzindo o estudante à reflexão e à formulação de conclusões próprias por meio de perguntas cuidadosamente elaboradas. Derivado da filosofia de Sócrates, seu propósito é estimular o pensamento crítico, a autorreflexão, a descoberta do conhecimento e a resolução de problemas pelo próprio aluno, em vez de fornecer respostas diretas (PAUL; ELDER, 2016).

Segundo Paul e Elder (2016), o questionamento socrático pode ser compreendido como uma abordagem sistemática de formulação de perguntas voltadas ao aprofundamento do pensamento e à investigação crítica de ideias. Diferentemente de perguntas superficiais, esse tipo de questionamento busca explorar conceitos fundamentais, identificar pressupostos e analisar implicações, conduzindo o raciocínio em direção a uma compreensão mais rigorosa e fundamentada. Trata-se de um processo dialógico em que o conhecimento é construído por

meio da reflexão orientada, permitindo ao estudante distinguir entre aquilo que realmente compreende e aquilo que ainda não domina.

No contexto educacional, essa abordagem apresenta diversas vantagens. Ele estimula o desenvolvimento do pensamento crítico, ao incentivar os alunos a analisarem seu próprio raciocínio, questionarem evidências e justificarem suas ideias. Além disso, promove a autonomia intelectual, uma vez que o estudante é conduzido a construir suas próprias conclusões, em vez de apenas receber informações prontas. Esse processo também favorece uma aprendizagem mais profunda e significativa, pois leva os alunos a irem além da memorização, explorando a estrutura do pensamento, reconhecendo erros e reformulando suas ideias com base em argumentos mais consistentes. Dessa forma, o questionamento socrático contribui para a formação de aprendizes mais reflexivos, críticos e capazes de lidar com problemas complexos (PAUL; ELDER, 2016).

Para que o questionamento socrático seja aplicado de forma efetiva, é importante que as perguntas sejam estruturadas de maneira sistemática, contemplando diferentes dimensões do pensamento. Conforme Paul e Elder (2016), essas perguntas podem ser organizadas a partir dos elementos do raciocínio, como propósito, problema, evidências, inferências, suposições e implicações. Nesse sentido, no contexto educacional, e particularmente no ensino de programação, o uso de perguntas pode ser orientado da seguinte forma: perguntas sobre o propósito, que buscam compreender a intenção da solução; perguntas sobre o problema, que visam esclarecer a tarefa ou dificuldade enfrentada; perguntas sobre evidências ou informações, que investigam os dados disponíveis; perguntas sobre o raciocínio, que exploram o processo lógico adotado; perguntas sobre suposições, que identificam ideias implícitas; e perguntas sobre implicações, que analisam as consequências das soluções propostas. A utilização dessas diferentes categorias contribui para aprofundar o pensamento do estudante, permitindo não apenas a correção de erros, mas a compreensão das estruturas subjacentes ao problema.

O método socrático também se destaca como uma estratégia eficaz para o desenvolvimento de habilidades de ordem superior. Por meio de perguntas que desafiam suposições, os alunos são estimulados a analisar, questionar e articular seus próprios pensamentos, identificando possíveis inconsistências e refinando sua compreensão ao longo do processo de aprendizagem (ORYNBASSAROVA; DE LASALA PORTA, 2024).

Entretanto, sua aplicação exige tempo e preparo tanto por parte do aluno, que deve se engajar ativamente com o material, quanto por parte do docente, que precisa elaborar questões

capazes de orientar o diálogo sem antecipar respostas. O ritmo mais lento e exploratório desse processo pode representar um desafio em contextos educacionais marcados por restrições de tempo e pela necessidade de cobrir extensos conteúdos programáticos. O êxito da abordagem depende fortemente da habilidade do professor. Mais do que um transmissor de informações, o professor atua como mediador, interpretando respostas, identificando intenções e conduzindo o diálogo de forma estruturada. A ausência dessa mediação pode levar a discussões superficiais e perda de foco (FAKOUR; IMANI, 2025).

Recentemente, estudos têm demonstrado o potencial da inteligência artificial em simular práticas socráticas de mediação. Tutores socráticos baseados em IA representam uma inovação promissora na educação personalizada, inspirando-se no método socrático para auxiliar a aprendizagem através de conversas interativas. Esses sistemas são projetados com foco na gestão do fluxo de conversação, criação de conteúdo, aplicação de estratégias de ensino e compreensão do aluno, garantindo respostas relevantes e conteúdo envolvente. Favero *et al.* (2024) apresentaram um *chatbot* socrático que melhorou o pensamento crítico dos alunos, enquanto Chukhlomin (2024) demonstrou como a engenharia de *prompts* pode estruturar diálogos educativos mais eficazes. Fakour e Imani (2025) e Goyal *et al.* (2026) reforçam que tutores baseados em IA podem aplicar estratégias socráticas com impacto positivo na aprendizagem.

A aplicação desse método para fomentar o raciocínio crítico no aprendizado de programação revela-se pertinente. No lugar de corrigir um erro de código diretamente ou explicar um conceito de forma expositiva, um chatbot socrático faria perguntas como: *"O que você esperava que este trecho de código fizesse?"*, *"Qual parte da mensagem de erro você não compreende?"*, *"Que outras abordagens você poderia tentar para resolver este problema?"*. Isso incentiva o aluno a depurar seu próprio raciocínio, a identificar suas lacunas de conhecimento e a construir uma compreensão mais profunda e duradoura dos conceitos.

2.2.3 Metodologias Ativas

Diversas estratégias pedagógicas e metodologias podem desempenhar um papel relevante no ensino de programação. Compreender essas abordagens e suas possibilidades de aplicação é fundamental para promover práticas educacionais mais eficazes. Nesse contexto, as metodologias ativas destacam-se por envolverem os alunos em atividades práticas que estimulam a reflexão e a aplicação de conceitos. Essas abordagens mantêm os estudantes

engajados e mentalmente ativos ao longo do processo de aprendizagem, integrando atividades que envolvem a coleta de informações, o pensamento crítico e a resolução de problemas.

As metodologias educacionais abrangem técnicas, procedimentos e estratégias que os professores utilizam para facilitar a aprendizagem e a cognição dos alunos. Essas metodologias são consideradas ativas, quando estão associadas à implementação de práticas pedagógicas centradas nos estudantes, promovendo sua autonomia e criando oportunidades para a aprendizagem por meio da ação e da experiência. Além disso, essas metodologias incentivam o desenvolvimento do pensamento crítico, a reflexão sobre as práticas realizadas, a troca de *feedback*, a aprendizagem colaborativa e o trabalho em grupo, bem como a exploração de atitudes e valores pessoais (BACICH; MORAN, 2017).

Para Bacich e Moran (2017, p. 38),

> [...] a aprendizagem mais profunda requer espaços de prática frequentes (aprender fazendo) e de ambientes ricos em oportunidades. Por isso, é importante o estímulo multissensorial e a valorização dos conhecimentos prévios para "ancorar" os novos conhecimentos.

As metodologias ativas passaram por um contínuo aprimoramento ao longo do século XX e seguem sendo fundamentais para as abordagens contemporâneas de ensino e aprendizagem. Esse progresso se reflete em diversos movimentos e teorias educacionais que, ao longo do século passado, contribuíram para sua evolução. A aprendizagem experiencial, por exemplo, valoriza o envolvimento direto dos alunos com o conhecimento, sustentando que a aprendizagem é mais eficaz quando os estudantes conseguem conectar conceitos teóricos às suas experiências pessoais (KOLB, 1984). Já o Construtivismo, fundamentado nas teorias de Piaget (1970), propõe que os alunos construam ativamente seu próprio aprendizado, rompendo com o modelo tradicional em que o conhecimento é simplesmente transmitido do professor para o aluno. Essa perspectiva foi fundamental para o desenvolvimento das metodologias ativas, que colocam os estudantes como protagonistas de seu próprio processo de aprendizagem. Além disso, a aprendizagem colaborativa, inspirada nas ideias de Vygotsky, ressalta a importância da interação social e do trabalho em grupo para o aprendizado. A troca de ideias e a resolução conjunta de problemas promovem uma compreensão mais profunda e significativa dos conteúdos (VYGOTSKY, 1989).

De modo geral, a aprendizagem ativa ocorre por meio de métodos instrucionais que envolvem os alunos em atividades significativas, incentivando a reflexão sobre o que estão fazendo e aprendendo. Assim, sua principal característica é uma abordagem centrada no aluno, em que docentes e estudantes colaboram e interagem com os conteúdos de forma

dinâmica (PRINCE, 2004).

Com o avanço da tecnologia, as metodologias ativas de aprendizagem estão cada vez mais integradas às ferramentas digitais. A tecnologia desempenha um papel essencial no apoio à aprendizagem ativa, proporcionando aos alunos novas oportunidades de engajamento e interação (BACICH; MORAN, 2017). Essas metodologias visam modernizar a educação, transformando aulas tradicionais e expositivas em experiências de aprendizagem envolventes e significativas. Em um ambiente tecnológico, os estudantes demandam práticas inovadoras para se tornarem profissionais competitivos, utilizando a ciência e a tecnologia como recursos fundamentais para seu desenvolvimento profissional.

Dessa forma, considerando que a aprendizagem ativa é uma parte inerente ao ensino de programação de computadores devido às suas características práticas e aplicadas, observa-se na literatura resultados relevantes decorrentes da aplicação de metodologias ativas nesse contexto. Os estudos de Berssanette e Francisco (2021) e Anicic e Stapic (2022) destacam a aplicação de diversas metodologias ativas no ensino de programação de computadores. Esses pesquisadores enfatizam o uso de abordagens como *gamificação*, aprendizagem baseada em jogos, aprendizagem baseada em projetos, aprendizagem baseada em problemas, sala de aula invertida e programação baseada em pares. No mesmo contexto, Sobral (2021a) conduziu uma revisão sistemática da literatura sobre estratégias pedagógicas no ensino de programação na educação superior, analisando periódicos indexados no Scopus da Elsevier entre 2000 e 2019. As principais metodologias identificadas por Sobral (2021a) incluem programação baseada em pares (OTHMAN *et al*., 2019), aprendizagem baseada em jogos (STEPAHN *et al.*, 2020; KARLING *et al.*, 2021), e aprendizagem baseada em problemas e projetos (COELHO; GUEDES, 2020; CHANG; CHUNG; CHANG, 2020).

De acordo com Calderon, Silva e Feitosa (2023), essas abordagens têm impactos positivos, incluindo melhorias na aprendizagem dos alunos, no desenvolvimento do pensamento crítico, nas atitudes em relação ao aprendizado, além de contribuírem para a redução do insucesso nas disciplinas de programação. As metodologias ativas possibilitam a criação de ambientes de aprendizagem onde os alunos podem construir conhecimento, desenvolver pensamento crítico, refletir sobre as atividades realizadas e explorar atitudes e valores pessoais, enquanto se engajam em práticas de aprendizagem eficazes.

2.2.4 Aprendizagem Baseada em Problemas

A aprendizagem baseada em problemas (ABP) é uma metodologia ativa que tem ganhado destaque por sua abordagem centrada no aluno e por promover um aprendizado ativo e significativo. A ABP envolve a apresentação de problemas complexos e reais aos alunos, que devem buscar soluções aplicando seus conhecimentos e habilidades. Essa metodologia estimula a investigação, a integração de teoria e prática e o desenvolvimento de habilidades de resolução de problemas. Essa metodologia não apenas facilita a compreensão dos conceitos de programação, mas também desenvolve competências essenciais, como o pensamento crítico e a capacidade de resolver problemas, preparando os alunos para desafios futuros no campo da tecnologia.

Savery (2006) define a ABP como um método instrucional e curricular centrado no aluno, que permite aos estudantes realizar pesquisas, integrar teoria e prática, e usar conhecimentos e habilidades para desenvolver soluções para problemas específicos. Essa definição é corroborada por William *et al*. (2019), que a destacam como uma aprendizagem ativa e autodirigida. Na prática, essa metodologia apresenta aos alunos um problema ou situação desafiadora, incentivando-os a aplicar conhecimentos prévios e adquirir novos conhecimentos. Esse processo promove um envolvimento ativo dos estudantes na construção do conhecimento e no desenvolvimento de competências em diversos contextos. Assim, a metodologia proporciona uma aprendizagem autônoma, ao mesmo tempo em que contextualiza e aplica situações reais na programação de computadores (CHANG; CHUNG; CHANG, 2020).

Ilxomovna e Akmalovna (2023) destacam a importância da integração do método de ABP no ensino de linguagens de programação. Os autores enfatizam que essa abordagem, que envolve a resolução de problemas reais ou simulados pelos alunos, promove o desenvolvimento de habilidades práticas, pensamento crítico, colaboração e comunicação.

No contexto do estudo de Chen (2023), o uso da ABP problemas possibilitou que os alunos definissem problemas específicos, identificassem pontos-chave e dividissem problemas maiores, facilitando assim a resolução de problemas e o desenvolvimento do pensamento computacional dos estudantes. Ao combinar design com a ABP, os alunos conseguiram alcançar resultados positivos, como uma compreensão mais profunda das linguagens de programação, melhoria das habilidades práticas e desenvolvimento de

competências essenciais para uma carreira bem-sucedida na área de programação.

2.2.5 Programação em Pares (*Pair Programming*)

Outra metodologia amplamente utilizada no ensino de programação é a programação em pares. Originada na metodologia Extreme Programming (XP), envolve dois programadores que trabalham colaborativamente em um computador, no mesmo projeto, algoritmo ou código, utilizando um único monitor, teclado e mouse. Todas as etapas do processo de desenvolvimento, incluindo design, depuração e teste, são realizadas em conjunto, não se limitando apenas à codificação. Nessa metodologia, um dos programadores assume o papel de "piloto", controlando o teclado e o mouse para digitar o código, enquanto o outro atua como "navegador", observando erros, discutindo abordagens de design alternativas e oferecendo sugestões. Os papéis são alternados regularmente, e o código produzido por um membro do par é revisado por ambos antes de ser oficialmente aceito como parte do programa (WILLIAMS *et al.*, 2000; OTHMAN *et al*., 2019).

Essa abordagem tem sido adotada como uma estratégia de ensino colaborativo, especialmente em cursos introdutórios de programação. Estudos empíricos indicam que a aplicação de programação em pares em ambientes educacionais pode estimular a motivação e o interesse pela aprendizagem, além de contribuir para o desenvolvimento de habilidades cognitivas e de resolução de problemas (BECK; CHIZHIK, 2013). Evidências mais abrangentes são apresentadas por Umapathy e Ritzhaupt (2017), que, por meio de uma meta-análise envolvendo 18 estudos e 3.308 estudantes, identificaram efeitos positivos significativos da programação em pares no desempenho em atividades práticas de programação (efeito médio a alto) e em exames (efeito pequeno a médio), além de um aumento de aproximadamente 8,3% nas taxas de aprovação em comparação ao trabalho individual.

A prática de programação em pares ganhou ampla aceitação desde seu surgimento na década de 90, devido às suas vantagens em relação à programação individual. Esse método de aprendizagem colaborativa demonstrou ser bem-sucedido, e seu potencial continua sendo uma área de pesquisa relevante. Em comparação com a programação individual, a programação em pares oferece vantagens significativas, como a produção de código original com menos erros, maior qualidade e rapidez na criação, aprendizado mútuo entre os programadores, construção de confiança e aprimoramento das habilidades em equipe. Além disso, essa abordagem

melhora a atividade cerebral, proporciona maior satisfação aos programadores, e há evidências de seus efeitos a longo prazo nos resultados de aprendizagem em diferentes áreas (BANIĆ; KONECKI; KONECKI, 2023).

A prática de programação em pares tem sido associada ao compartilhamento de conhecimento e experiência, proporcionando aos alunos um foco mais acentuado nos detalhes durante o trabalho colaborativo. Além disso, a programação em pares contribuiu para a melhoria das práticas e habilidades de programação dos alunos. Os participantes mostraram maior concentração, níveis elevados de confiança e um aprimoramento nas habilidades de trabalho em equipe. Além disso, a programação em pares ajudou a reduzir as frustrações quando os códigos individuais não atendiam às expectativas. Em particular, Othman *et al*. (2019) demonstram que essa abordagem contribui significativamente para o desenvolvimento do pensamento lógico em estudantes iniciantes.

A programação em pares baseia-se nas teorias socioconstrutivistas de Vygotsky, que ressaltam a importância das interações sociais e da colaboração no processo de aprendizagem. Segundo Vygotsky (1978), as interações sociais são fundamentais para o aprendizado, e o desenvolvimento do indivíduo resulta da relação com o mundo e com as pessoas ao seu redor. Dessa forma, os indivíduos constroem conhecimento ao interagir ativamente com outros e compartilhar suas compreensões. A programação em pares oferece um ambiente de apoio que promove essa construção colaborativa de conhecimento, permitindo que os programadores aprendam uns com os outros, desafiem suposições e resolvam problemas coletivamente.

O estudo de Dos Santos e Cury (2023) apresenta diferentes benefícios associados à programação em pares:

- Compartilhamento de conhecimento e aprendizagem: ao trabalharem juntos, os indivíduos podem compartilhar suas percepções e estratégias de resolução de problemas, enriquecendo a experiência de aprendizagem para ambos os participantes. Programadores menos experientes se beneficiam do aprendizado com seus colegas mais experientes, o que acelera a curva de aprendizado e aprimora as habilidades de programação;
- Qualidade do código e design de software: a programação em pares promove a qualidade do código através da revisão contínua do código. Enquanto o “navegador” analisa ativamente o código escrito pelo “motorista”, pode identificar problemas, sugerir melhorias e garantir a adesão aos padrões de codificação. Esse processo reduz

a probabilidade de erros, melhora a legibilidade do código e resulta em um *software* melhor projetado. O esforço coletivo e a responsabilidade compartilhada também contribuem para a qualidade geral e a facilidade de manutenção da base de código;

- Detecção e depuração de erros: a natureza colaborativa da programação em pares facilita a detecção precoce de erros e a depuração eficaz. Com dois programadores envolvidos ativamente, problemas podem ser identificados e resolvidos mais rapidamente do que se estivessem trabalhando sozinhos. A discussão conjunta sobre o código permite detectar erros lógicos, identificar casos extremos e depurar o código de maneira mais eficiente, resultando em software mais robusto e confiável;
- Dinâmica e comunicação de equipe aprimoradas: ao trabalharem juntos, os programadores desenvolvem habilidades interpessoais mais fortes, como escuta ativa, *feedback* construtivo e comunicação eficaz. Essa abordagem incentiva discussões abertas, o compartilhamento de diferentes perspectivas e a construção de confiança entre os membros da equipe, levando a uma colaboração mais eficiente, maior produtividade e um ambiente de trabalho mais positivo.

Apesar das inúmeras vantagens associadas à programação em pares, sua adoção no ensino ainda enfrenta desafios. Estudos recentes indicam que a eficácia dessa abordagem depende de diversos fatores, incluindo dificuldades de comunicação e coordenação entre os participantes, especialmente em contextos distribuídos, além da dependência de ferramentas tecnológicas adequadas (BANIĆ; KONECKI; KONECKI, 2023; XU; CORREIA, 2024). Diferenças no nível de conhecimento prévio entre os parceiros também podem impactar negativamente a colaboração, podendo levar à desmotivação do participante menos experiente ou à predominância do mais experiente durante a atividade.

Além disso, aspectos relacionados à dinâmica de equipe, como a compatibilidade entre os pares e o nível de engajamento individual, desempenham um papel fundamental na eficácia da programação em pares. Problemas como participação desigual, comportamento de “carona” e dificuldades na alternância de papéis ainda são relatados na literatura, podendo comprometer os resultados da aprendizagem (XU; CORREIA, 2024). Soma-se a isso o fato de que a programação em pares pode demandar maior esforço e coordenação entre os participantes, o que, em alguns casos, pode reduzir a percepção de eficiência individual. Dessa forma, devido à complexidade de coordenar dois programadores para trabalhar de maneira sincronizada, a adoção generalizada desse método em ambientes educacionais permanece um desafio.

O surgimento de ferramentas de IA generativa oferece novas oportunidades para aprimorar a experiência de aprendizagem dos estudantes no modelo de programação em pares (pareamento aluno+IA). Durante a colaboração aluno+IA, em vez de absorver passivamente as informações da IA, os alunos devem se envolver ativamente com a interação contínua com a IA enquanto a IA coleta e analisa os dados dos alunos para formar uma compreensão precisa da condição de aprendizagem dos alunos (LIU; LI, 2024). Embora a investigação específica sobre a combinação de IA e programação em pares ainda seja limitada, estudos indicam que, ao utilizar ferramentas de IA como um par virtual, os alunos podem participar de conversas interativas, receber *feedback* personalizado e aprender com a vasta base de conhecimento do sistema de IA (DOS SANTOS; CURY, 2023).

A programação em pares com IA é uma abordagem que combina as habilidades dos alunos com as capacidades das ferramentas de IA, visando aumentar a eficácia, a criatividade e a qualidade no aprendizado de programação. A ideia é aproveitar as competências complementares entre humanos e IA para melhorar o processo educacional, oferecendo *feedback* contínuo e suporte personalizado. Apesar do crescente interesse pelo uso de IA como parceira no ensino de programação, ainda são escassos os estudos empíricos que avaliem seu impacto na aprendizagem, na motivação dos alunos e na qualidade do código produzido. Essa lacuna ressalta a necessidade de pesquisas sistemáticas sobre como a programação em pares com IA pode influenciar esses aspectos no contexto educacional (CHEN, 2024).

Recentemente, um conjunto significativo de pesquisas vem investigando o uso de recursos de Inteligência artificial, dentre elas os modelos de linguagem de grande escala (*Large Language Models* - LLMs) para auxiliar no processo de ensino e aprendizado em diferentes áreas, incluindo a programação. Considerando que este é um recurso utilizado nesta pesquisa, a próxima seção será dedicada a detalhar o uso de IA generativa na educação, com ênfase no ensino de programação.

2.3 Inteligência Artificial na Educação

No cenário atual, marcado por rápidas transformações e um fluxo contínuo de informações, a educação precisa se adaptar para atender as demandas do século XXI. Tecnologias como inteligência artificial (IA), realidade virtual e aumentada, internet das coisas e aprendizado de máquina, têm contribuído para a criação de ambientes de aprendizagem mais interativos, dinâmicos e centrados no estudante, favorecendo o

desenvolvimento de habilidades como pensamento crítico, resolução de problemas e colaboração (LAMATTINA, 2023).

A IA é uma área de pesquisa da Ciência da Computação voltada ao desenvolvimento de algoritmos, modelos e sistemas capazes de planejar tarefas para alcançar metas, realizar tarefas de forma autônoma e executar tarefas cognitivas comumente associadas à mente humana. Conforme a definição da OECD recomendada pelo Conselho de Inteligência Artificial (OECD, 2023, p. 7):

> Um sistema de IA é um sistema baseado em máquina que, para objetivos explícitos ou implícitos, infere, a partir das informações que recebe, como gerar resultados como previsões, conteúdos, recomendações ou decisões que podem influenciar ambientes físicos ou virtuais. Diferentes sistemas de IA variam nos seus níveis de autonomia e adaptabilidade após a implantação.

A presença da IA tem se tornado cada vez mais integrada ao cotidiano. Atividades aparentemente simples, como pesquisas na internet, interações em plataformas de mídia social, uso de aplicativos de navegação ou escuta de músicas, agora envolvem algoritmos de IA que atuam silenciosamente em segundo plano, otimizando a experiência do usuário e transformando a maneira como essas interações ocorrem. No campo da educação, a aplicação da IA, embora não seja uma novidade, está se tornando cada vez mais comum e relevante, com potencial para gerar impactos substanciais (MELLO *et al*., 2023).

Estudos sobre o uso da IA na educação remontam à década de 1960, evidenciando o interesse de longa data e a investigação contínua neste campo. Um exemplo pioneiro é o *chatbot* Eliza, criado por Joseph Weizenbaum em 1964, capaz de interagir com humanos e reconhecido como o primeiro chatbot utilizado para fins educacionais (WEIZENBAUM, 1966). Outro marco na mesma década foi o SCHOLAR, desenvolvido por Jaime Carbonell em 1968 , um programa inovador que respondia automaticamente a perguntas sobre geografia da América do Sul em linguagem natural e avaliava as respostas dos alunos. O SCHOLAR foi um dos primeiros modelos do que hoje são conhecidos como Sistemas Tutores Inteligentes (CARBONELL, 1969).

Inicialmente o campo de IA na educação utilizou-se de bases de conhecimento e sistemas especialistas, resultando em aplicações focadas e específicas para determinados domínios. No entanto, à medida que as técnicas de aprendizado de máquina e de processamento de linguagem natural evoluíram, as aplicações educacionais de IA adquiriram maior complexidade e amplitude, passando a abranger diversas disciplinas, metodologias e tarefas acadêmicas, como avaliação, *feedback* e desenvolvimento de habilidades. A

intersecção entre IA e educação abrange vários subcampos incluindo mineração de dados educacionais, análise de aprendizagem e aprendizagem personalizada, entre outras. Estes subcampos contribuem com diversas perspectivas e metodologias para a integração da IA em contextos educacionais (MELLO *et al*., 2023).

Wang *et al*. (2024) realizaram uma revisão sistemática da literatura em artigos publicados entre 1984 e junho de 2022 sobre o uso de inteligência artificial na educação, visando identificar as principais categorias de aplicações exploradas nesse campo. Foram destacadas quatro categorias primárias, cada uma com suas subcategorias correspondentes: (1) Aprendizagem adaptativa e tutoria personalizada, (2) Avaliação e gerenciamento inteligentes, (3) Criação de perfil e previsão, e (4) Tecnologias ou produtos emergentes. Entre todas as aplicações, os aplicativos de aprendizagem adaptativa e tutoria personalizada foram os mais estudados, representando 40% dos artigos na amostra, seguidos por aplicativos de avaliação e gerenciamento inteligentes (24,8%), criação de perfil e previsão (20%) e produtos emergentes em educação (15,2%).

Atualmente, os recentes avanços na IA generativa influenciaram significativamente todos os campos relacionados à IA na educação, proporcionando novas oportunidades para as instituições de ensino melhorarem a qualidade da sua experiência de aprendizagem a partir de diferentes perspectivas, como avaliação, *feedback* e envolvimento, entre outras (GAŠEVIĆ; SIEMENS; SADIQ, 2023). Neste contexto, a IA generativa é imensamente promissora na definição do futuro da educação, introduzindo novas abordagens para otimizar as experiências de aprendizagem e facilitar o crescimento educativo.

2.3.1 Inteligência Artificial Generativa

A IA generativa é um tipo de inteligência artificial desenvolvida para criar conteúdos, como imagens, música e textos, a partir da identificação de padrões em grandes volumes de dados, utilizando técnicas de aprendizado profundo. Diferentemente de sistemas tradicionais, que apenas analisam ou classificam informações, esses modelos são capazes de gerar conteúdo original, muitas vezes dando a impressão de possuir criatividade própria. Essa tecnologia tem sido amplamente adotada em diversas áreas, desde assistentes virtuais até a geração de imagens e textos (OECD, 2026).

Em novembro de 2022, a OpenAI lançou o ChatGPT, uma ferramenta baseada em modelos de linguagem de grande escala (*Large Language Models* - LLMs), que utilizam

técnicas avançadas de aprendizado supervisionado e por reforço (OPENAI, 2022). O ChatGPT é desenvolvido com base no modelo *Generative Pre-trained Transformer* (GPT), que opera a partir da previsão probabilística de sequências de palavras, sendo capaz de gerar textos coerentes e contextualizados a partir de uma entrada textual (FLORIDI; CHIRIATTI, 2020). Essa abordagem permite ao ChatGPT completar frases, parágrafos ou até mesmo artigos inteiros a partir do contexto fornecido como entrada. O modelo foi treinado com grandes volumes de dados coletados da internet e em diversos idiomas, o que lhe possibilita realizar tarefas complexas (OPENAI, 2022). Com uma ampla gama de funcionalidades, incluindo a produção de textos e documentos, o ChatGPT rapidamente se popularizou entre estudantes e profissionais, tornando-se um marco na disseminação da IA generativa.

Baseada em tecnologias como aprendizado profundo e redes neurais, a IA generativa analisa grandes volumes de dados para identificar padrões e gerar novos conteúdos. Exemplos de IA generativa incluem redes adversariais generativas (GANs), *autoencoders* variacionais e modelos generativos pré-treinados, como o GPT. Esses sistemas podem criar diversos tipos de conteúdo, como texto, imagens e áudio, e em alguns casos, conseguem até simular aspectos da criatividade humana (MELLO *et al.*, 2023).

As tecnologias que sustentam a IA generativa pertencem à família de IA conhecida como aprendizado de máquina (*Machine Learning* - ML), que emprega algoritmos para promover a melhoria contínua e automática de seu desempenho com base em dados. Um dos tipos de ML que tem impulsionado muitos dos avanços recentes da IA, como o reconhecimento facial, são as redes neurais artificiais (RNAs), inspiradas no funcionamento do cérebro humano e nas conexões sinápticas entre neurônios (UNESCO, 2023). A IA generativa utiliza redes de aprendizagem profunda para identificar padrões e relações significativas em grandes conjuntos de dados, o que lhe permite gerar artefatos originais em diversas formas, como imagens, textos, áudios, músicas, vídeos, entre outros (JOVANOVIC; CAMPBELL, 2022).

Tanto as tecnologias de IA generativa de texto quanto as de imagem são fundamentadas em um conjunto de tecnologias de IA que já estão acessíveis aos pesquisadores há vários anos. O ChatGPT, por exemplo, emprega um modelo de transformador generativo pré-treinado (GPT), enquanto a IA generativa de imagem geralmente faz uso das redes generativas adversariais (GANs). O Quadro 1 apresenta as principais técnicas utilizadas na IA generativa (UNESCO, 2023).

Quadro 1 - Principais técnicas utilizadas em IA generativa

<table>
<tr><td colspan="2">Aprendizado de máquina (ML)</td><td>Um tipo de IA que utiliza dados para aprimorar automaticamente seu desempenho.</td></tr>
<tr><td colspan="2">Rede neural artificial (RNA)</td><td>Um tipo de ML inspirado na estrutura e funcionamento do cérebro humano (por exemplo, as conexões sinápticas entre neurônios).</td></tr>
<tr><td rowspan="3">IA generativa de texto</td><td>Transformadores de propósito geral</td><td>Um tipo de RNA que é capaz de focar em diferentes partes de dados para determinar como elas se relacionam entre si.</td></tr>
<tr><td>Modelos de Linguagem de Grande Escala (LLM)</td><td>Um tipo de transformador de propósito geral que é treinado com grandes quantidades de dados textuais</td></tr>
<tr><td>Transformador generativo pré-treinado (GPT)</td><td>Um tipo de modelo de linguagem de grande escala (LLM) que é pré-treinado em quantidades ainda maiores de dados, o que permite ao modelo capturar as nuances da linguagem e gerar textos coerentes e de acordo com o contexto.</td></tr>
<tr><td rowspan="2">IA generativa de imagem</td><td>Redes generativas adversariais (GANs)</td><td rowspan="2">Tipos de rede neurais usados para geração de imagens.</td></tr>
<tr><td>Autoencoders variacionais (VAEs)</td></tr>
</table>

Fonte: Adaptado de UNESCO (2023)

A popularização recente da IA generativa é atribuída ao ChatGPT, um modelo de linguagem de grande porte (LLM). Esses modelos são sistemas de aprendizado de máquina que utilizam técnicas de aprendizado profundo para processar e compreender a linguagem natural. Treinados com grandes volumes de dados textuais, aprendem a identificar padrões e relações entre elementos do idioma (ZHAO *et al*., 2023).

Os LLMs operam com base em arquiteturas do tipo sequência a sequência, gerando novos conteúdos a partir de um texto de entrada. Isso significa que o modelo depende de um *prompt* textual inicial para produzir respostas, proporcionando uma experiência dinâmica e interativa. A geração do conteúdo ocorre por meio da análise estatística das distribuições de palavras, *pixels* ou outros elementos dos dados, permitindo identificar e reproduzir padrões recorrentes, como a probabilidade de determinadas palavras sucederem outras (UNESCO, 2023).

Apesar dos avanços recentes, os LLMs se fundamentam em conceitos desenvolvidos ao

longo do século XX, como redes neurais. As abordagens mais atuais se baseiam em arquiteturas do tipo *Transformer*, que utilizam mecanismos de atenção para modelar dependências no texto (VASWANI *et al.*, 2017). Além disso, técnicas de incorporação de palavras (*word embeddings*) criam representações numéricas usando vetores para capturar o significado das palavras, e, nesse espaço vetorial, palavras com significados semelhantes tendem a estar localizadas próximas umas das outras, permitindo capturar relações semânticas entre elas.

A arquitetura *Transformer* é um modelo de aprendizado profundo projetado para lidar com tarefas de sequência a sequência. Sua principal inovação é a capacidade de considerar diversos fatores críticos durante a modelagem. O *Transformer* leva em conta a ordem do texto, os mecanismos de desambiguação de palavras e atribui maior importância aos conceitos-chave, resultando em uma compreensão contextual mais aprimorada. Esse avanço na compreensão do modelo melhora significativamente sua capacidade de produzir resultados mais precisos e relevantes (ZHAO *et al.*, 2023).

A introdução dos mecanismos de atenção causou uma verdadeira revolução no campo do aprendizado profundo. A atenção permite que os modelos foquem dinamicamente em partes relevantes dos dados de entrada, de forma análoga à maneira como os humanos prestam atenção a certos aspectos de uma cena visual ou conversa. Esse foco seletivo é essencial em tarefas onde o contexto é fundamental, como na compreensão da linguagem ou no reconhecimento de imagens. Nos *Transformers*, os mecanismos de atenção são usados para ponderar a influência de diferentes *tokens* de entrada na geração de uma saída, permitindo um avanço significativo no desempenho dos modelos (ZHAO *et al.*, 2023).

Além da geração de textos, imagens e vídeos, a IA generativa tem sido amplamente aplicada à geração de código-fonte. Esses modelos são treinados com grandes volumes de código-fonte ou são LLMs aprimorados com dados de treinamento adicionais específicos para programação, possibilitando a geração automática de código. Diversas ferramentas foram desenvolvidas com esse objetivo, como o Deepmind AlphaCode (LI *et al*., 2022), o Amazon CodeWhisperer (DESAI; DEO, 2022) e o CodeBERT (FENG *et al.*, 2020), além de modelos mais recentes, como o CodeGemma, do Google (CODEGEMMA TEAM *et al*., 2024), Code Llama, da Meta (ROZIÈRE *et al*., 2023), DeepSeek Coder (DEEPSEEK-AI *et al.*, 2024), IBM Granite Code (MISHRA *et al*., 2024) e Qwen Coder (QWEN *et al*., 2024), entre outros.

Além desses, existem modelos de linguagem de propósito geral que, embora não sejam exclusivamente voltados à programação, foram otimizados para tarefas de codificação, como

Claude Sonnet e Claude Opus, da Anthropic (ANTHROPIC, 2024), e Gemini Pro, do Google (GOOGLE, 2025).

Em 2021, a OpenAI lançou o Codex (CHEN *et al.*, 2021), uma versão do GPT-3 treinada em bilhões de linhas de código provenientes de repositórios públicos, especialmente do GitHub. O Codex possui familiaridade tanto com linguagens naturais quanto com linguagens de programação, sendo mais proficiente em inglês e *Python*, mas também capaz de trabalhar com outras linguagens naturais, como o espanhol, e linguagens de programação como *Go*, *JavaScript*, *Perl*, *PHP, Ruby, Shell, Swift* e *TypeScript*.

Estudos de Finnie-Ansley *et al*. (2022) indicam que o Codex é capaz de resolver problemas de programação introdutórios com desempenho superior ao de um aluno médio, posicionando-se no quartil superior ao responder às mesmas questões de exames. Além disso, Chen *et al.* (2022) observaram que o desempenho do modelo na geração de código correto a partir de descrições em linguagem natural melhora significativamente quando ele também gera casos de teste, os quais auxiliam na seleção das soluções mais adequadas.

Além da geração de código a partir de descrições, esses modelos também são capazes de realizar tarefas como completude, tradução, correção, resumo e explicação de código. Atualmente, eles são utilizados como base para assistentes de programação baseados em IA, como, por exemplo, o GitHub Copilot[1], Tabnine[2] e Cursor[3], todos com funcionalidades de sugestão automática de código.

Embora a maioria dessas ferramentas tenha sido inicialmente desenvolvida para uso profissional, o GitHub Copilot, criado em parceria entre o GitHub e a OpenAI, destaca-se como uma solução acessível também para iniciantes. Utilizando modelos derivados do Codex, o Copilot sugere trechos de código em tempo real à medida que o usuário digita. Lançado oficialmente em junho de 2022, está disponível como extensão para ambientes de desenvolvimento integrados (IDEs), como o Visual Studio Code e o JetBrains. Descrito como "seu par de programação de IA", o GitHub Copilot faz referência à prática de programação em pares, uma metodologia amplamente utilizada no ensino de programação (LIU; LI, 2024). A Figura 2 ilustra o funcionamento de seus principais recursos.

[1] GitHub Copilot: https://github.com/features/copilot/
[2] Tabnine: https://www.tabnine.com/
[3] https://cursor.com/

Figura 2 - Funcionamento dos recursos do GitHub Copilot.



Fonte: Liu e Li (2024)

Essas ferramentas têm o potencial de transformar a forma como os usuários interagem com sistemas computacionais, especialmente no contexto da programação, ao oferecer suporte contínuo durante o desenvolvimento de soluções. No entanto, seu uso no contexto educacional exige atenção, uma vez que a geração automática de código pode levar à redução do esforço cognitivo do estudante e à dependência excessiva da ferramenta.

Nesse sentido, torna-se relevante investigar abordagens que utilizam a IA não apenas como ferramenta de apoio, mas como mediadora do processo de aprendizagem, alinhada a estratégias pedagógicas que incentivem a participação ativa do estudante. Essa perspectiva aproxima o uso da IA generativa do conceito de programação em pares, no qual a interação desempenha papel central no desenvolvimento do conhecimento.

2.3.2 Aplicações da IA Generativa na Educação

A IA generativa tem o potencial de transformar a educação, oferecendo benefícios capazes de impactar o ensino e a aprendizagem. Com seu uso, a educação pode se tornar mais personalizada, acessível e eficaz. Entre os principais benefícios destacam-se a adaptação do conteúdo às necessidades individuais dos alunos, o aumento do engajamento, o fornecimento de suporte educacional personalizado e a adequação das avaliações ao nível de habilidade de cada estudante.

Uma das maneiras mais relevantes da IA generativa na educação é a personalização do ensino. Por meio de algoritmos de aprendizado de máquina, a IA é capaz de analisar dados sobre o desempenho e as preferências dos alunos, ajustando o conteúdo e as atividades

conforme suas necessidades. Essa abordagem proporciona uma experiência de aprendizagem personalizada, auxiliando no desenvolvimento de habilidades e na superação de lacunas de conhecimento. Além disso, assistentes virtuais facilitam o acesso a recursos e informações, auxiliando os alunos na busca por materiais educacionais relevantes e atualizados, bem como *feedback* direcionado (LAMATTINA, 2023).

Diversas plataformas educacionais já incorporam esses recursos. A OpenEdX, desenvolvida por Harvard e MIT, oferece recomendações personalizadas de cursos e trajetórias de aprendizagem adaptativas, com base na análise dos dados dos alunos (OPENEDX, 2024). Ao analisar esses dados de forma inteligente, a plataforma identifica os pontos fortes e as áreas de melhoria dos alunos, sugerindo cursos que se alinham aos seus interesses e objetivos individuais.

De forma semelhante, o Coursera, uma plataforma de aprendizagem online, utiliza algoritmos de IA para avaliar o progresso dos estudantes, identificando seus pontos fortes e fracos, e ajustando dinamicamente seus percursos de aprendizagem, permitindo que se concentrem nas áreas que demandam maior atenção (IKONOMOU, 2023).

A Khan Academy também emprega IA para analisar o desempenho, o progresso e as preferências de aprendizagem dos alunos criando trajetórias de aprendizagem personalizadas com recomendações de conteúdos e exercícios específicos adaptados às necessidades individuais. Mais recentemente, a plataforma lançou o Khamigo, um assistente baseado em LLMs, capaz de analisar respostas, fornecer *feedback* imediato e apoiar o processo de aprendizagem de forma adaptativa (KHAN ACADEMY, 2024; KHANMIGO, 2024).

Além da personalização, a IA contribui para o aumento do engajamento e para o suporte educacional contínuo. Ferramentas como c*hatbots* educacionais e assistentes virtuais permitem a interação em linguagem natural, respondendo dúvidas, fornecendo explicações adicionais e orientando os estudos. A tutoria virtual, viabilizada por esses sistemas, tem se consolidado como uma estratégia eficaz para oferecer suporte individualizado e tornar o processo de aprendizagem mais interativo e acessível (LAMATTINA, 2023).

Os sistemas de tutoria inteligentes (STI) também desempenham papel relevante nesse contexto. Os STI são projetados para simular experiências de tutoria personalizadas e individuais, utilizando IA para se adaptar às necessidades de aprendizagem de cada aluno, diagnosticar o desempenho, fornecer *feedback* e recomendar intervenções. Essas soluções permitem respostas rápidas e precisas aos estudantes, enquanto os professores podem se

concentrar em aspectos pedagógicos mais complexos (AVCI *et al*., 2025).

Outra área emergente é o uso da IA para análise de emoções e sentimentos dos alunos durante o processo de aprendizagem, por meio da interpretação da linguagem ou de expressões faciais. Essa abordagem possibilita identificar necessidades emocionais e oferecer suporte mais adequado, contribuindo para uma experiência educacional mais completa (LAMATTINA, 2023).

A IA também tem contribuído significativamente para o processo de avaliação. Sistemas de avaliação adaptativa ajustam o nível de dificuldade das atividades conforme o desempenho do estudante, proporcionando uma análise mais precisa de suas habilidades. A plataforma Tutoria, por exemplo, utiliza IA para sugerir tags relevantes para educadores durante a avaliação de questões abertas e gerar *feedback* automático com base nas respostas dos alunos, oferecendo uma camada adicional de suporte ao ensino (MELLO *et al.*, 2022).

Além disso, a automatização de tarefas administrativas, como correção de exercícios e atendimento a dúvidas frequentes, contribui para a otimização do tempo dos professores e alunos (LAMATTINA, 2023). Outra aplicação da IA na educação é a análise preditiva, que utiliza técnicas estatísticas e de aprendizado de máquina para examinar dados educacionais históricos e prever resultados futuros (MELLO *et al.*, 2023). Na educação, a análise preditiva pode ser empregada para prever o desempenho dos alunos, a taxa de abandono, o nível de envolvimento emocional e cognitivo, além de identificar indicadores de crescimento e alunos em risco. A Arizona State University, por exemplo, utiliza o sistema *eAdvisor* para identificar estudantes em risco e possibilitar intervenções mais rápidas e eficazes (EADVISOR, 2024).

Apesar das inúmeras possibilidades, é fundamental considerar os desafios associados ao uso da IA na educação. Questões relacionadas ao acesso desigual às tecnologias podem ampliar disparidades educacionais, especialmente em contextos com menor infraestrutura digital. Além disso, sistemas baseados em IA podem reproduzir vieses presentes nos dados de treinamento, influenciando a qualidade e a neutralidade das respostas geradas. Também se destacam preocupações quanto à privacidade dos dados dos estudantes e aos impactos no desenvolvimento socioemocional, considerando a possível redução da interação humana no processo educativo (OECD, 2026).

Evidências emergentes sugerem que, embora as ferramentas de IA generativa de uso geral possam melhorar o desempenho dos alunos em tarefas, elas não necessariamente promovem ganhos de aprendizagem. A delegação de tarefas cognitivas a *chatbots* de uso

geral pode gerar riscos de preguiça metacognitiva e desinteresse, prejudicando a aquisição de habilidades a longo prazo. Diversos estudos indicam que, embora os alunos com acesso a essas ferramentas produzam resultados de maior qualidade do que seus colegas, essa vantagem desaparece, e, em alguns casos, se inverte, em avaliações nas quais o acesso à IA é removido. Em contrapartida, ferramentas de IA generativa educacionais, projetadas ou utilizadas com um propósito pedagógico intencional, tendem a apresentar melhorias mais sustentadas na aprendizagem (OECD, 2026).

Nesse contexto, destacam-se iniciativas como o projeto europeu *AI Pioneers,* responsável pelo desenvolvimento do Suplemento de IA ao Quadro Europeu de Competência Digital para Educadores (DigCompEdu), que descreve as competências necessárias a professores e formadores para o uso de IA na educação. Esse suplemento busca complementar e atualizar o DigCompEdu, incorporando competências críticas relacionadas à IA e reconhecendo seu impacto nos processos de ensino e aprendizagem. Dessa forma, enfatiza a necessidade de que os educadores estejam preparados para compreender, avaliar e utilizar essas tecnologias de maneira eficaz e responsável (BEKIARIDIS, 2024).

Por fim, destaca-se que, embora a IA ainda seja pouco compreendida no contexto educacional, ela envolve competências fundamentais do século XXI, como pensamento crítico, colaboração e inovação. Nesse sentido, Vicari *et al.*, 2022 desenvolveram uma proposta de experiência curricular relacionada à IA, com o objetivo de oferecer aos educadores do Ensino Médio um conjunto de atividades voltadas à reflexão e à aprendizagem nas dimensões cognitiva, ética, física, artística, social e afetiva. O documento propõe a organização de competências e habilidades, além de fomentar o debate sobre o uso da IA na educação. A publicação integra o projeto Capacitação em IA para o Desenvolvimento do Pensamento Computacional nas Escolas Públicas do Rio Grande do Sul (CIARS), uma iniciativa pública de extensão universitária voltada à pesquisa e desenvolvimento de materiais para o ensino de IA nas escolas.

A IA pode contribuir para a melhoria do aprendizado quando utilizada com um propósito pedagógico claro ou quando as estratégias de ensino são reformuladas para se adaptarem à sua disponibilidade. Ao integrar a experiência docente ao processo de design, as ferramentas de IA generativa podem ampliar a capacidade de ensino dos professores, gerando benefícios que superam aqueles alcançados isoladamente por humanos ou sistemas de IA. O desenvolvimento conjunto dessas ferramentas com professores e usuários finais é, portanto, essencial para garantir seu valor educacional (OECD, 2026).

2.3.3 Desafios da IA Generativa na Educação

Apesar dos inúmeros benefícios, a incorporação da IA generativa na educação envolve desafios relevantes, especialmente no que se refere ao acesso, à privacidade e à proteção dos dados dos estudantes. A utilização dessas tecnologias pressupõe o tratamento de grandes volumes de informações, o que levanta questões éticas e legais relacionadas ao armazenamento, uso e compartilhamento de dados pessoais. Nesse contexto, a coleta e a análise de dados educacionais exigem cuidados rigorosos, de modo a evitar usos indevidos, vazamentos ou acessos não autorizados.

Assim, a privacidade dos estudantes e a segurança das informações devem ser tratadas como prioridades no ambiente educacional, em conformidade com legislações específicas, como a Lei Geral de Proteção de Dados (LGPD), no Brasil, e o Regulamento Geral sobre a Proteção de Dados (GDPR), na União Europeia. Dessa forma, torna-se essencial a adoção de medidas robustas de segurança e governança de dados, garantindo que o uso da IA ocorra de maneira ética, responsável e alinhada à proteção dos direitos dos estudantes (GUIMARAES JUNIOR *et al.*, 2025).

Outra preocupação relevante é o potencial de preconceito e discriminação nos sistemas de IA. Como os algoritmos são treinados com dados existentes, eventuais vieses presentes nestes dados podem ser reproduzidos e até e amplificados pelos sistemas, resultando em classificações tendenciosas ou recomendações inadequadas. Para mitigar esses efeitos, é necessário adotar práticas responsáveis no treinamento dos modelos, bem como monitorá-los continuamente, visando garantir maior equidade nos resultados. Além disso, questões relacionadas à acessibilidade e à equidade no uso dessas ferramentas também merecem atenção. A dependência de infraestrutura tecnológica e de acesso à internet de qualidade pode gerar exclusão digital, limitando oportunidades educacionais para estudantes de contextos menos favorecidos e ampliando desigualdades preexistentes (ISOTANI *et al.*, 2023).

A redução da interação humana também é uma preocupação associada ao uso de IA na educação (GAŠEVIĆ, 2023). Embora essas ferramentas possam oferecer suporte significativo, a ausência de contato humano direto pode prejudicar o desenvolvimento de competências sociais e emocionais nos alunos. Além disso, há o risco de dependência excessiva da tecnologia, o que pode comprometer a autonomia, a criatividade e o pensamento crítico. Nesse sentido, a IA não deve substituir o papel dos educadores, mas atuar como

complemento, sendo fundamental manter o equilíbrio entre o uso da tecnologia e a mediação humana no processo de aprendizagem (GUIMARAES JUNIOR *et al.*, 2025).

As considerações éticas também são centrais na educação baseada em IA, envolvendo aspectos como explicabilidade dos modelos, consentimento informado e uso responsável de conteúdos e avaliações gerados por IA. Garantir transparência, responsabilidade e ética no uso dessas tecnologias é essencial para preservar a confiança e a integridade do processo educacional (MOLENAAR *et al*., 2025; OECD, 2026).

Dessa forma, o uso da IA na educação deve ser conduzido de maneira responsável, buscando equilibrar seus benefícios com a mitigação de seus riscos, de modo a contribuir para um sistema educacional mais inclusivo, equitativo e ético.

2.3.4 Inteligência Artificial Generativa no Ensino de Programação

As linguagens e ferramentas de programação têm evoluído continuamente para incorporar recursos como autocompletar código, tradução entre linguagens, documentação, depuração e testes (SARKAR *et al*, 2022). Desde a década de 1970, as linguagens de quarta geração foram desenvolvidas com o objetivo de simplificar a programação, evidenciando que a facilitação do processo de escrita de código sempre foi uma meta importante (SOBRAL, 2019).

Nesse contexto, o uso de IA generativa tem ganhado destaque. Ferramentas baseadas em IA atuam como assistentes especializados, sendo capazes de esclarecer conceitos, responder perguntas, identificar erros e explicar o funcionamento de trechos de código. Além disso, podem gerar explicações para códigos pouco documentados e sugerir soluções para tarefas rotineiras, contribuindo para o aumento da produtividade. Essa evolução tem transformado o cenário da programação, promovendo uma colaboração centrada no ser humano, na qual programadores e IA trabalham em conjunto para aprimorar as habilidades de programação (SÁNCHEZ-GORDÓN *et al.,* 2023).

No ensino de programação, essa colaboração refere-se à interação ativa entre estudantes e IA, com o objetivo de tornar o aprendizado mais eficiente e personalizado. Nesse processo, os alunos não apenas recebem informações, mas participam de interações contínuas com a IA, que analisa dados para compreender seu *status* de aprendizagem e oferecer suporte mais adequado (KIM; LEE; CHO, 2022; YAN; NAKAJIMA SAWADA, 2024).

Tanto a colaboração entre aluno e IA quanto a aprendizagem colaborativa entre

estudantes estão fundamentadas nas teorias construtivistas de Vygotsky e Piaget. Vygotsky, por meio da ZDP, destaca o papel do mediador, que pode ser a IA ou outro aluno, em apoiar o aprendiz na realização de tarefas que ele não conseguiria executar sozinho, promovendo a internalização do conhecimento (VYGOTSKY, 1989). Já Piaget (1970) enfatiza a construção ativa do conhecimento, em que o aluno reorganiza suas estruturas cognitivas ao interagir com o ambiente, seja por meio do *feedback* da IA ou da colaboração com outros alunos. Em ambos os casos, as interações geram desafios e conflitos cognitivos que estimulam o desenvolvimento intelectual, levando o aprendiz a ajustar e expandir seus esquemas mentais.

A aprendizagem colaborativa envolve discussões ativas, geralmente em pequenos grupos, partindo da premissa de que o conhecimento não é simplesmente transmitido de professores para alunos, mas construído por meio de interações. Entre suas características estão interações frequentes e de qualidade, responsabilidade individual, interdependência positiva e metas compartilhadas de aprendizado, promovendo a autonomia e reduzindo a dependência excessiva de professores ou outros membros do grupo (DILLENBOURG, 1999; YANG, 2023).

Entretanto, a colaboração entre aluno e IA generativa não deve ser equiparada de forma simplista à interação entre alunos. Diferentemente dos colegas humanos, a IA pode apresentar comportamentos excessivamente ativos ou passivos. Devido à sua capacidade de processar e fornecer grandes quantidades de informação rapidamente, a IA pode dominar a interação, dificultando o estabelecimento de uma parceria equilibrada. Por outro lado, sua atuação depende da iniciativa do aluno, limitando sua participação espontânea no diálogo. Por isso, é fundamental que os estudantes adotem uma postura ativa, conduzindo seus objetivos, estratégias e avaliações de aprendizagem, de modo a preservar sua autonomia (YAN; NAKAJIMA; SAWADA, 2024).

Diversos estudos têm investigado o potencial das ferramentas de IA generativa para aprimorar a experiência de aprendizagem em programação. Yilmaz e Karaoglan Yilmaz (2023), em um estudo de caso de oito semanas, identificaram benefícios como respostas rápidas, apoio à depuração e estímulo ao pensamento, mas também apontaram limitações, como respostas incorretas e possível redução do esforço dos alunos. Em estudos quase experimentais, Yilmaz e Karaoglan Yilmaz (2023) e Sun *et al.* (2024) exploraram a integração da IA generativa no ensino de programação, focando em aspectos como engajamento, pensamento computacional, autoeficácia e motivação. O primeiro estudo utilizou métodos mistos e revelou melhorias significativas no pensamento computacional, na

confiança dos alunos em programação e na motivação para aprender. O segundo estudo comparou a programação assistida pelo ChatGPT com métodos tradicionais de aprendizagem, observando que, embora o ChatGPT aumentasse o engajamento, não houve uma melhora significativa no desempenho de programação.

Hartley *et al.* (2024), ao examinar a eficácia do ChatGPT em áreas como conteúdo instrucional, ferramentas de programação, *feedback* e planejamento, concluíram que o sucesso da IA generativa depende, em grande parte, da competência do usuário. As ferramentas de IA generativa podem oferecer várias opções de código, mas o programador ainda precisa avaliar e decidir quais utilizar, o que exige habilidades como pensamento crítico, resolução de problemas e originalidade. Portanto, é essencial enfatizar a educação em ética e responsabilidade para os programadores (SÁNCHEZ-GORDÓN *et al.*, 2023).

Jalil *et al.* (2023) destacam que uma forma eficaz de integrar a IA na educação é projetar atividades em sala de aula ou laboratórios utilizando assistentes de ensino inteligentes (tutores de IA). Esses tutores podem melhorar a compreensão dos alunos, adaptando-se a seus objetivos, velocidade de aprendizagem e nível de conhecimento, garantindo uma educação mais personalizada e eficaz.

Autores com Becker *et al.* (2023) e Denny *et al.* (2024) destacam em seus trabalhos diversas oportunidades que as ferramentas de IA generativa oferecem para auxiliar no ensino de programação:

- Exemplos de código para aprendizagem: os alunos de programação são geralmente incentivados a praticar escrevendo código para resolver pequenos problemas, as ferramentas de IA podem ser uma alternativa, gerando exemplos e testes, para que os alunos possam verificar seu trabalho durante a prática;
- Variedade de soluções: ferramentas de IA também podem ser usadas para apresentar aos alunos diferentes abordagens para resolver um problema. Embora existam muitas formas de solucionar a maioria dos problemas, os iniciantes nem sempre têm conhecimento dessas variações. Thompson *et al.* (2006) destacam que apresentar variação apropriada no ensino de programação é importante, pois ajuda os alunos a compreender as diferenças entre as diversas maneiras de escrever código. Eckerdal e Thuné (2005) também defendem esse ponto, baseando-se na teoria da variação, ao afirmar que os professores devem fornecer recursos que destaquem as diferentes dimensões da variação dos conceitos em estudo;

- Revisão de código: as abordagens atuais de avaliação em disciplinas de programação priorizam a correção, em vez da qualidade ou estilo do código. Com a capacidade das ferramentas de IA de gerar automaticamente soluções sintaticamente corretas, a avaliação pode-se concentrar nas diferenças entre múltiplas soluções corretas e fazer análises sobre o estilo e a qualidade das soluções;
- Revisão por pares: ferramentas de IA poderiam criar soluções de qualidade variável, que podem ser usadas em tarefas focadas na avaliação da qualidade do código, envolvendo os alunos nos níveis mais elevados da taxonomia de Bloom. Isto pode ser útil para gerar discussões sobre abordagens alternativas e a qualidade das soluções, e fornecer a base para exercícios de refatoração;
- Geração de exercícios: as ferramentas de IA podem criar novos exercícios de programação a partir de um único exemplo, enriquecendo os recursos de aprendizagem;
- Explicações do código: explicações detalhadas de código são um recurso importante para auxiliar os alunos a desenvolver uma compreensão dos conceitos de programação. A IA pode oferecer explicações claras do código, facilitando a assimilação dos conceitos pelos alunos;
- Explicar conceitos algorítmicos com clareza: a construção dos *prompts* inseridos nos modelos de geração de código influencia diretamente seu desempenho. Com o surgimento dos LLMs, a engenharia de *prompt* tornou-se um campo emergente e específico no uso desses modelos. Os alunos podem se concentrar em fornecer descrições mais precisas dos problemas, permitindo que os modelos de geração de código produzam soluções funcionais com maior eficácia;
- Superação do bloqueio do programador: alunos frequentemente enfrentam dificuldades para iniciar ou prosseguir com tarefas de programação. Ferramentas de IA podem auxiliar na geração do código inicial, permitindo que os alunos se concentrem em expandi-lo, direcionando seus esforços para reescrever, refatorar e depurar. Vaithilingam *et al.* (2022) identificaram que o GitHub Copilot ajudou os alunos a iniciar tarefas de programação, gerando o código inicial e oferecendo, assim, a oportunidade de aprimorá-lo, em vez de escrever cada linha desde o início;
- Superação de barreiras tradicionais: iniciantes enfrentam diversas dificuldades ao aprender a programar, e as mensagens de erro do compilador são uma barreira

conhecida ao progresso dos alunos. Pesquisas recentes mostraram que a IA é capaz de traduzir essas mensagens de erro para uma linguagem natural, além de sugerir correções apropriadas com base no código de entrada e nas mensagens de erro.

Com base nas oportunidades oferecidas pela IA generativa, o ensino de disciplinas introdutórias de programação, que geralmente se concentra inicialmente na sintaxe e nos princípios básicos, pode ser aprimorado ao utilizar modelos de geração de código para lidar com tarefas de implementação de baixo nível. Isso permitiria que os alunos focassem em algoritmos de nível superior. Essa abordagem é semelhante ao uso de ambientes de programação baseados em blocos, que simplificam a sintaxe e direcionam a atenção dos alunos para questões algorítmicas. Assim, o ensino poderia priorizar a resolução de problemas e algoritmos, adiando discussões detalhadas sobre sintaxe para um estágio mais avançado (BECKER *et al.*, 2023; DENNY *et al.*, 2024).

Apesar das inúmeras possibilidades e benefícios, a disponibilidade de ferramentas de IA generativa para a geração de código levanta preocupações sobre seu uso de forma a limitar a aprendizagem ou dificultar o trabalho dos educadores. A integridade acadêmica em computação é uma questão complexa, especialmente em um campo que incentiva a reutilização de código e práticas colaborativas. O uso de código gerado automaticamente suscita questões importantes sobre integridade acadêmica e reutilização de código. Becker *et al.* (2023) destacam alguns dos principais desafios:

- Má conduta acadêmica: o uso de ferramentas de IA, capazes de superar alunos em provas de programação e gerar código correto para algoritmos, pode intensificar a desonestidade acadêmica, oferecendo um caminho fácil para estudantes que buscam apenas obter boas notas em vez do aprendizado;
- Atribuição: muitos IDEs permitem o uso de ferramentas de IA para sugestões automáticas durante o desenvolvimento de código, o que levanta questionamentos sobre se o uso de código gerado por IA constitui má conduta acadêmica, especialmente quando o IDE oferece várias formas de assistência. Isso provoca reflexões sobre a atribuição da propriedade intelectual do conteúdo gerado por IA a humanos, além de questionar conceitos como plágio e a interpretação da contribuição intelectual em processos de geração de conteúdo assistidos por máquinas;
- Reutilização e licenciamento de código: o uso de ferramentas de IA pode levantar questões de licenciamento ao gerar novos conteúdos, mesmo quando os dados

utilizados pelos modelos são de acesso público. Códigos disponíveis publicamente frequentemente possuem diferentes licenças que exigem atribuição aos autores originais. Desenvolvedores que utilizam código por meio de ferramentas de IA podem, inadvertidamente, estarem sujeitos a requisitos de licenciamento desconhecidos;

- Adequação para iniciantes: embora a maioria do código utilizado para treinar ferramentas de IA seja de acesso público, ele frequentemente não é apropriado para estudantes iniciantes. Além do viés de qualidade, os estilos de código disponíveis publicamente tendem a ser mais avançados e diferentes daqueles que são mais adequados para quem está começando a aprender programação;
- Segurança: ensinar práticas de codificação seguras tem sido amplamente documentada nos últimos anos. Portanto, a segurança do código gerado por IA é de extrema importância, mesmo em ambientes educacionais;
- Excesso de confiança: ferramentas de IA podem levar a dependência excessiva do usuário. Especialmente com ferramentas integradas em IDEs, iniciantes podem se acostumar rapidamente com soluções sugeridas, o que pode comprometer a compreensão dos problemas;
- Reforçar comportamentos que impedem a aprendizagem: código incorreto sugerido pela IA pode fazer com que os alunos percam a confiança no *feedback* dos IDEs, comprometendo sua aprendizagem e dificultando a interpretação de mensagens de erro.

Diante das potencialidades e desafios discutidos, observa-se que a IA generativa apresenta um cenário promissor para o ensino de programação, especialmente quando utilizada como ferramenta de apoio ao processo de aprendizagem. No entanto, a análise da literatura evidencia que ainda há lacunas importantes, sobretudo no que se refere à adoção de abordagens pedagógicas estruturadas que orientem o uso dessas tecnologias em sala de aula.

Nesse contexto, torna-se relevante investigar como a IA generativa vem sendo utilizada no ensino de programação, bem como identificar tendências, limitações e oportunidades de pesquisa. Assim, o próximo capítulo apresenta uma análise de trabalhos relacionados, com o objetivo de compreender o estado da arte e posicionar a presente pesquisa frente as contribuições existentes na literatura.

3. TRABALHOS RELACIONADOS

O uso de ferramentas de IA generativa é uma área em rápido desenvolvimento, com crescimento significativo de pesquisas sobre sua aplicação na educação. No contexto do ensino de programação, diversos estudos têm investigado como essas tecnologias podem apoiar o processo de aprendizagem. Diante disso, este capítulo apresenta duas revisões de literatura realizadas ao longo do desenvolvimento desta tese, com o objetivo de identificar e analisar investigações que abordam o uso da IA generativa como suporte no ensino de programação.

Inicialmente, foi conduzido um Mapeamento Sistemático da Literatura (MSL), com o propósito de fornecer uma visão geral da área, analisando e classificando os estudos existentes. Esse mapeamento permitiu examinar o estado atual da pesquisa, identificar tendências, desafios e lacunas, além de indicar direções para estudos futuros. Como complemento, um segundo estudo de Revisão Sistemática da Literatura (RSL) foi realizado, com foco em pesquisas empíricas envolvendo alunos no contexto educacional. O objetivo foi responder à seguinte questão: *"Como a Inteligência Artificial Generativa está sendo utilizada para apoiar o ensino de programação?"*, investigando também as estratégias pedagógicas e metodologias de ensino e aprendizagem empregadas.

As seções seguintes apresentam respectivamente esses dois estudos.

3.1 MSL - Panorama Geral Sobre o Uso Inteligência Artificial Generativa no Ensino de Programação

Buscando identificar na literatura as pesquisas que abordam o uso da IA generativa como suporte no ensino de programação, foi realizado um Mapeamento Sistemático da Literatura (MSL), publicado em Silva *et al.* (2024a), com o objetivo de responder à seguinte questão de pesquisa: *"Qual o panorama das pesquisas sobre o uso de IA generativa no ensino de programação?"*. A técnica utilizada foi proposta por Petersen *et al.* (2008), que define o mapeamento sistemático como uma abordagem voltada à obtenção de uma visão geral de uma área de pesquisa, permitindo identificar evidências de pesquisas existentes e sua distribuição.

A busca foi conduzida em abril de 2024, considerando publicações entre janeiro de 2021 e março de 2024, em cinco bases de dados internacionais IEEE *Digital Library*,

ScienceDirect, *Scopus*, *Web of Science* e ACM *Digital Library*, sendo estas consideradas fontes de pesquisa científica internacionais que possuem um amplo conjunto de publicações relacionadas à Informática e Educação.

A busca foi construída com as palavras-chave *"Generative Artificial Intelligence"*, *"GenAI"*, *"Generative AI"*, *"LLM"*, *"AI Code Generators"*, *"Codex"*, *"ChatGPT"*, *"Copilot"*, *"GPT"*, *"Gemini"*, *"OpenAI"*, *"Programming Education"*, *"Introductory Programming"*, *"Programming Instruction"*, *"Programming Learning"*, *"Programming Teaching"* com busca avançada, considerados os estudos e soluções que possuíssem os termos listados ou seus respectivos sinônimos em seus títulos, resumos ou palavras-chave.

Os critérios de inclusão e exclusão foram definidos para que, após os resultados das buscas, fosse possível selecionar os artigos que se enquadram no contexto do estudo. Os critérios de inclusão são: *"Deve mencionar IA generativa, grandes modelos de linguagem ou uma ferramenta específica que utilize essas tecnologias, como GitHub Copilot, ChatGPT, Codex, etc."*, *"Deve ter aplicabilidade direta ao ensino de programação."*, *"Pelo menos 4 páginas (inclusive)"*. Já os critérios de exclusão usados foram: *"Não utiliza IA generativa"*, *"Não se aplica ao ensino de programação"*, *"Menos de 4 páginas"*, *"Publicado antes de 2021"* e *"Acesso fechado"*.

A busca de artigo nas bases de dados elencadas acima, utilizando a *string* de busca, supracitada, resultou em um total de 160 estudos potenciais. Os resultados das pesquisas nas bases de dados foram importados para a ferramenta Parsifal[4], e, com auxílio da mesma, foram excluídos os artigos duplicados. Foram encontrados um total de 70 artigos duplicados, resultando 90 artigos para avaliação. Em seguida, foi realizada a leitura do título e resumo de cada artigo visando atender aos critérios de inclusão e exclusão, resultando na exclusão de 39 estudos e restando 51 para a leitura completa. A Tabela 1 exibe de forma resumida esses resultados de busca.

Tabela 1 - Resultados da Busca

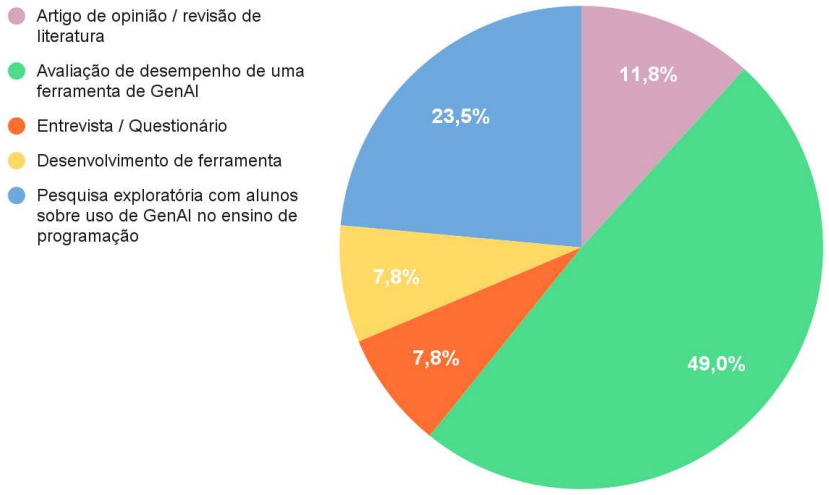
Base	ACM	IEEE	Web of Science	ScienceDirect	Scopus	Total
Quantidade de Estudos retornado na busca	46	12	30	12	60	160
Duplicados	32	9	28	1	0	70
Rejeitados	10	1	0	8	20	39
Analisados	4	2	2	3	40	51

Fonte: elaborado pela autora (2024).

[4] https://parsif.al/

A partir da leitura dos artigos, e objetivando responder à questão de pesquisa: *"Quais são os tipos de estudos e as abordagens utilizadas?"*, foi utilizada a codificação qualitativa para identificar e sintetizar os temas das pesquisas com relação ao objetivo e a abordagem de uso da ferramenta de IA generativa no estudo. Assim, os artigos revisados foram classificados em cinco categorias: (a) avaliação do desempenho, capacidades e limitações das ferramentas de GenAI, (b) desenvolvimento de ferramenta, (c) entrevistas/questionários (d) pesquisa exploratória com alunos usando uma ferramenta de IA generativa no ensino de programação e, (e) artigos de opinião e revisão. Alguns artigos se enquadram em mais de uma categoria, e nesses casos, foram classificados na categoria mais adequada (Figura 3).

Figura 3 - Classificação dos artigos com relação ao objetivo da pesquisa



Fonte: Elaborado pela autora (2024).

Observou-se que a categoria mais representativa foi a de avaliação do desempenho das ferramentas (25 artigos, 49% do total), indicando uma tendência inicial dos estudos em investigar as capacidades e limitações dos modelos de IA generativa, especialmente na geração de código e resolução de problemas de programação (ALI; WIBOWO, 2023; WANG *et al.*, 2023; MORADI DAKHEL *et al.*, 2023; CIPRIANO; ALVES, 2023; DUNDER *et al.*, 2024; FINNIE-ANSLEY *et al.*, 2022, WERMELINGER, 2023). Outros trabalhos analisaram como as ferramentas respondem a diferentes tipos de exercícios de programação, tais como questões de múltipla escolha (SAVELKA et al., 2023a) e problemas de Parsons[5] (REEVES *et al.*, 2023), ou ainda, serem aprovados em questões de provas (SAVELKA *et al.*, 2023b; TORRES, 2023; FINNIE-ANSLEY *et al.*, 2023, AMBROSINI *et al.*, 2023), visando demonstrar o desempenho acadêmico da IA em disciplinas de programação.

[5] Problemas de *Parsons* são uma forma de atividade de programação na qual os alunos são solicitados a organizar fragmentos de código, os quais compreendem a solução do problema, no estilo de um quebra-cabeça.

Nesta categoria de estudos, Finnie-Ansley *et al.* (2022) avaliaram o desempenho do Codex na resolução de questões introdutórias de exames de programação o comparando com as pontuações de alunos que fizeram os mesmos exames. O Codex foi usado para concluir dois conjuntos de questões de teste, que envolveram um total de 23 problemas de programação. O Codex pontuou 15,7/20 (78,5%) no Teste 1 e 19,5/25 (78,0%) no Teste 2. Ao calcular a média das pontuações de ambos os testes, descobriu-se que o Codex ficou em 17º lugar entre 71 alunos, ficando no quartil superior da classe. Isso indica que o Codex superou a maioria dos alunos na resolução de tarefas de programação CS1. Em um estudo subsequente, Finnie-Ansley *et al.* (2023) avaliaram o desempenho do Codex na resolução de questões mais avançadas do exame CS2 (estruturas de dados e algoritmos). Conforme os resultados, o Codex ficou em 66º lugar entre 264 alunos, ficando também no quartil superior da classe. O Codex foi mais preciso na resolução de problemas de programação mais complexos do que a maioria dos alunos.

Além disso, Moradi Dakhel *et al.* (2023) conduziram uma análise comparativa do desempenho do *GitHub Copilot* ao abordar vários problemas de programação. Quando encarregado de implementar algoritmos de classificação simples, como *Bubble Sort*, *Bucket Sort* e *Insertion Sort*, o GitHub Copilot forneceu um código altamente preciso na primeira tentativa (com taxas de correção de 100%, 85,71% e 100%, respectivamente). No entanto, implementar a *Heap Sort*, que envolve criar um *heap* máximo e, em seguida, escrever uma função de classificação, provou ser mais desafiador, pois o GitHub Copilot produziu uma taxa de correção de 0% na primeira tentativa e 9,09% na segunda tentativa, indicando desempenho abaixo do ideal. É evidente que, à medida que a complexidade do algoritmo aumenta, o desempenho de resolução de problemas do GitHub Copilot diminui.

Em alguns estudos, dados quantitativos sobre o desempenho de resolução de problemas das ferramentas foram coletados. Por exemplo, Denny *et al.* (2024) avaliaram o desempenho de resolução de problemas do GitHub Copilot em um conjunto de dados publicamente disponível de 166 problemas de programação. Conforme revelado pela pesquisa, o GitHub Copilot resolveu aproximadamente metade dos problemas na primeira tentativa. Além disso, ele resolveu 60% dos problemas restantes apenas aprimorando ou refinando ainda mais a descrição dos problemas por meio da linguagem natural.

Em geral, segundo os estudos, os LLMs apresentaram um desempenho equivalente, ou melhor, que o dos alunos iniciantes em programação, pelo menos para tarefas de geração de código de exercícios introdutórios. Já em situações de exercícios mais complexos, tais como

questões de múltipla escolha ou problemas de *Parsons* (que são um tipo de atividade utilizada em programação na qual os alunos são solicitados a organizar fragmentos de código, os quais compreendem a solução do problema, como se fosse um quebra-cabeça), os resultados sugerem que estes são mais desafiadores para os LLMs resolverem em comparação com problemas de escrita de código tradicionais. Neste sentido, alguns estudos destacam a importância de considerar que pequenas modificações nas instruções podem afetar o desempenho dos modelos de geração de código.

Além de testar a capacidade de gerar código, muitos estudos se concentraram em analisar os LLMs em atividades de auxílio aos alunos no aprendizado de programação. Estudos como os apresentados em (AZAIZ; DECKARM; STRICKROTH, 2023; PHUNG *et al.*, 2024; KIESLER, LOHR, KEUNING, 2023), trataram de analisar o *feedback* gerado pelo LLM para correção de exercícios de programação. Os estudos (AZAIZ, DECKARM, STRICKROTH, 2023; KIESLER, LOHR, KEUNING, 2023) apresentam resultados que destacam tanto as capacidades quanto às limitações dos LLMs no contexto educacional de programação. Ambos autores indicam que esses modelos de IA conseguem detectar e corrigir certos tipos de erros no código, fornecendo explicações textuais e sugestões para melhorias. No entanto, eles também apontam que os modelos têm dificuldades com erros lógicos e semânticos, além de fornecerem *feedbacks* enganosos ou inadequados em algumas situações.

Em Kiesler, Lohr e Keuning (2023), o ChatGPT teve um desempenho razoável na correção de erros de compilação, mas mostrou limitações com múltiplos erros e erros lógicos, além de fornecer informações enganosas em muitas respostas. Já em Azaiz, Deckarm e Strickroth (2023), o GPT-3.5 obteve uma taxa de acerto de 73% na identificação correta das submissões de estudantes e forneceu *feedback* apropriado em 47% dos casos. No entanto, também cometeu erros e sugeriu alterações que não estavam alinhadas com as instruções fornecidas. Ambos os estudos destacam a importância de orientar os alunos quanto às capacidades e limitações desses modelos de IA, além de desenvolver métodos pedagógicos adequados para o seu uso eficaz.

No estudo apresentado por Phung *et al.*(2024), o objetivo foi comparar diferentes modelos de LLMs para correção e *feedback*. Os resultados mostraram que a combinação de um modelo mais simples, GPT-3.5, para validar dicas de programação geradas por um modelo mais avançado, GPT-4, resultou em uma precisão de aproximadamente 95%, comparável à de tutores humanos. Isso foi alcançado mantendo uma cobertura superior a 70% em três conjuntos de dados reais de programação em *Python*. A pesquisa demonstrou a

eficácia da geração de *feedback* automatizado no ensino de programação, destacando a possibilidade do uso de modelos de IA generativa para avaliações automáticas e análises de aprendizado. No entanto, como limitações de ambos os estudos foi destacada a necessidade de uma investigação empírica em ambiente de sala de aula, analisando a eficácia e usabilidade do *feedback* gerado na perspectiva dos estudantes.

Os estudos apresentados em Leinonen *et al.* (2023), Hellas *et al.* (2023) e Ishizue *et al.* (2024), buscaram avaliar a capacidade das ferramentas de IA generativa em processos de correção e explicação de código, visando identificar quão eficazes os LLMs são na identificação de problemas nos códigos dos estudantes. Em Leinonen *et al.* (2023) os estudantes avaliaram as explicações de código geradas pelo modelo de linguagem GPT-3 como sendo mais fáceis de entender e com resumos mais precisos do código em comparação com as explicações criadas por outros estudantes.

Ainda nesta categoria, foram encontrados estudos que avaliam os LLMs para a geração de exercícios de programação (DOUGHTY *et al*., 2024; SARSA *et al.*, 2022; FAN *et al.*, 2023; JORDAN, LY, SOOSAI RAJ, 2024; FREITAS *et al.*, 2023), cujo objetivo é auxiliar os professores e instrutores, permitindo a geração de uma ampla variedade de exercícios, evitando a repetição de questões e oferecendo aos alunos uma gama diversificada de desafios, podendo personalizar os mesmos conforme as necessidades e níveis de habilidade dos estudantes, e possibilitando a criação rápida e escalável de exercícios para turmas grandes, sem comprometer a qualidade ou a individualização do ensino.

A categoria *"Desenvolvimento de ferramenta"* apresenta 4 estudos (7,8% do total) que tratam do desenvolvimento de aplicações criadas com as APIs de IA generativa. Nos trabalhos de Kwak, Jenkins, Kim (2023) e Wang (2023) são descritos o desenvolvimento de tutores de programação inteligentes capazes de gerar *feedback* automático. Em Karnalim *et al.* (2023) é apresentado o desenvolvimento de um detector de plágio para código assistido por IA generativa, baseado em anomalias de código. Ta, Nguyen, Gottipati (2023) relatam o desenvolvimento de uma ferramenta para geração de exercícios de programação.

Na categoria *"Entrevista/questionário"* também foram classificados 4 estudos. Três destes (LAU; GUO, 2023; SHEARD *et al.*, 2024; HUSAIN, 2024) realizaram entrevistas via questionários com instrutores de programação visando investigar as práticas atuais, preocupações e adaptações planejadas em relação ao uso de ferramentas de geração de código de IA na educação em ciência da computação. A maioria dos instrutores entrevistados reconhece que as ferramentas de IA podem ajudar os estudantes a gerar exemplos de código,

fornecer explicações personalizadas e oferecer um ponto de partida quando estiverem em dúvida. No entanto, expressaram preocupações sobre a dependência excessiva dessas ferramentas, destacando a importância do pensamento crítico no aprendizado de programação.

Nesta categoria, apresenta-se ainda o estudo de Rahman e Watanobe (2023) onde os alunos foram entrevistados, sendo que a maioria demonstrou satisfação com o suporte de programação oferecido pelo LLM, com uma porcentagem significativa de estudantes respondendo positivamente em relação à qualidade das sugestões e assistência recebidas. Em uma das perguntas realizadas, se as sugestões do ChatGPT os ajudavam a resolver problemas de programação, cerca de 86,7% dos alunos responderam "Sim". Isso indica que a maioria dos estudantes participantes da pesquisa estavam recorrendo ao ChatGPT para obter suporte em suas atividades de programação.

Na categoria *"Pesquisa exploratória sobre uso de IA generativa no ensino de programação"* foram classificados 12 (23,5%) estudos que avaliam o uso de ferramentas de IA generativa como apoio aos alunos na aprendizagem de programação. Nesta categoria foram realizados estudos empíricos em sala de aula com grupos de alunos que foram incentivados a utilizar as ferramentas de IA generativa para resolver atividades e exercícios de programação. Nos estudos de Prather *et al*. (2024), Yilmaz e Yilmaz (2023), Silva, Carlos *et al.* (2024), Karnalim *et al.* (2023), Haindl e Weinberger (2024), Yilmaz e Karaoglan Yilmaz (2023), Jing *et al*. (2024) e Sun *et al.* (2024), não foram especificadas a abordagem pedagógica para a intervenção. Os alunos experimentaram uma abordagem direta e interativa de aprendizagem e trabalharam na resolução de problemas de programação, sendo observado como ocorreu a interação desses alunos com o LLM. Já nos estudos de Dos Santos e Cury (2023), Jonsson e Tholander, (2022), Hu, Assadi e Mahroeian (2023) e Banić, Konecki e Konecki (2023), o LLM foi usado como um colega virtual para a programação em pares em cursos de programação de computadores.

Por fim, os estudos exploratórios com alunos mostram que a IA generativa tem sido utilizada como suporte direto na resolução de problemas de programação. No entanto, observa-se que, na maioria dos casos, essas intervenções não são acompanhadas por uma abordagem pedagógica estruturada, sendo utilizadas de forma exploratória. Diante desse panorama, identifica-se uma lacuna importante na literatura: a ausência de investigações que explorem o uso da IA generativa integrada a estratégias pedagógicas bem definidas, especialmente em contextos de sala de aula.

3.2 RSL - IA Generativa no Ensino de Programação: Estratégias Pedagógicas e Metodologias

Como complemento ao MSL, foi realizada uma segunda revisão, publicada em Silva *et al.* (2024b), com foco em estudos empíricos envolvendo alunos em sala de aula. A esta revisão foram incorporadas novas publicações do período de abril a agosto de 2024, além da base de dados nacional SOL[6] da SBC, que não havia sido incluída no mapeamento anterior. O estudo concentrou-se na análise detalhada dos artigos classificados na categoria (d) pesquisas exploratórias com alunos utilizando ferramentas de IA generativa no ensino de programação, visando examinar as estratégias pedagógicas e metodologias de ensino adotadas, bem como os resultados e desafios identificados.

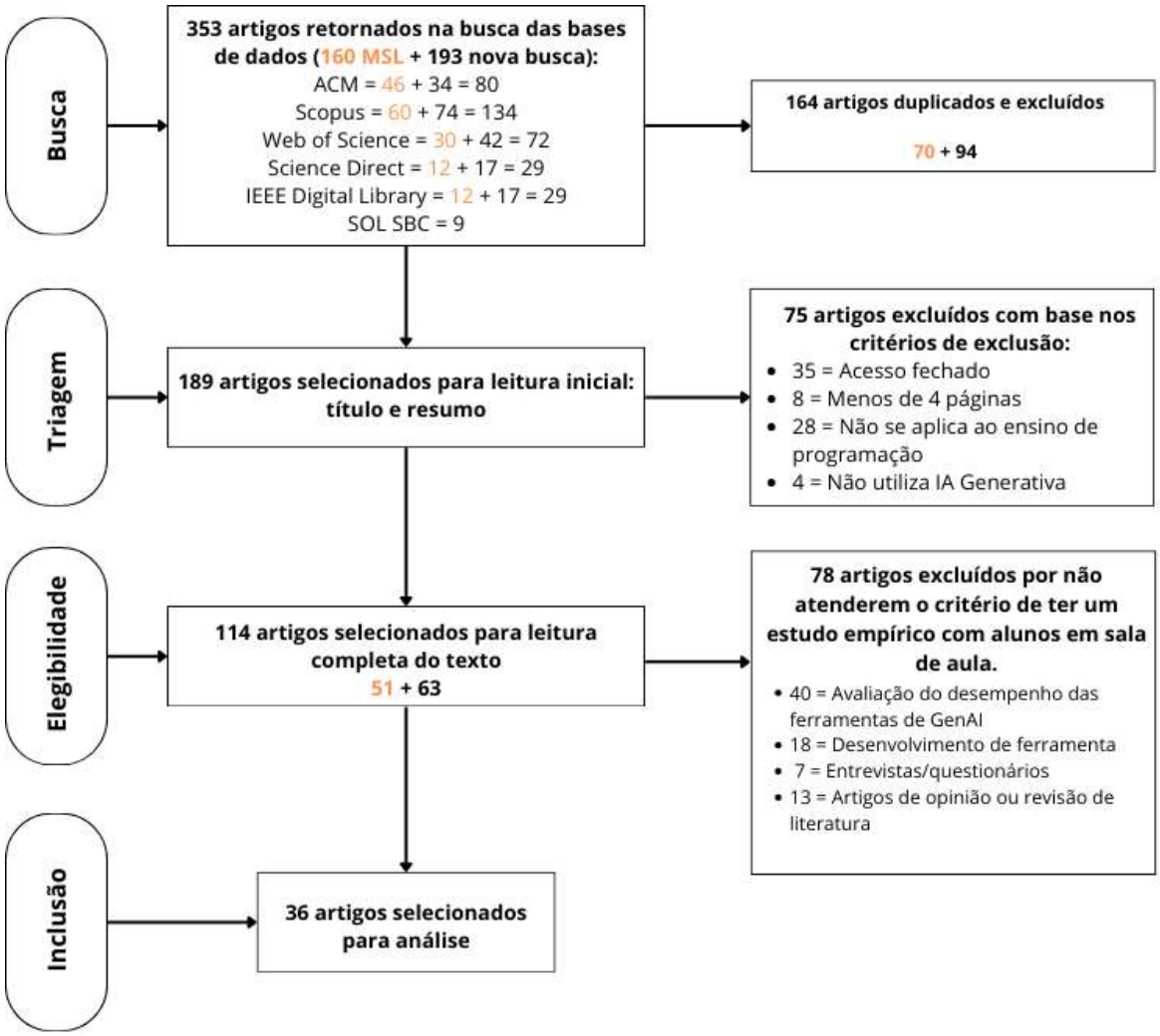
A busca nas bases de dados foi realizada novamente, utilizando a mesma *string* de pesquisa do MSL, com o acréscimo de um novo critério de inclusão: *"Deve incluir um estudo empírico com alunos em sala de aula"*. O principal objetivo dessa revisão foi identificar os estudos mais relevantes que aplicaram o uso de IA generativa no apoio ao ensino de programação em contextos empíricos com estudantes. Para direcionar a investigação foram elaboradas as seguintes questões: QP1: Quais ferramentas de IA generativa estão sendo utilizadas nas experiências de ensino e aprendizagem de programação? QP2: Quais metodologias ou estratégias pedagógicas estão sendo utilizadas em conjunto com essas ferramentas? QP3: Quais são os principais resultados e impactos observados? QP4: Quais são os principais desafios enfrentados nessas experiências?

A busca resultou em 353 estudos, dos quais 160 foram retornados pelo MSL e 193 pela nova busca. Esses resultados foram importados para a ferramenta Parsifal, que auxiliou na exclusão de artigos duplicados, totalizando 164 remoções. Em seguida, realizou-se a leitura dos títulos e resumos, resultando na exclusão de 75 estudos com base nos critérios de exclusão, restando 114 artigos para análise parcial.

Após a leitura parcial dos 114 estudos, 78 foram excluídos por não atenderem ao critério de inclusão que exigia a realização de estudo empírico com alunos em sala de aula. Dessa forma, restaram 36 artigos para leitura completa e análise. A Figura 4 apresenta o processo de busca e seleção dos artigos nas bases de dados.

[6]https://sol.sbc.org.br/

Figura 4 - Processo de busca e seleção dos artigos



Fonte: Elaborado pela autora (2024)

Foram analisadas as ferramentas de IA generativa utilizadas nos artigos revisados, e o ChatGPT foi identificado como o mais recorrente, aparecendo em 72% dos estudos (26 artigos). Nesses trabalhos, os alunos utilizaram o ChatGPT para explorar conceitos de programação, buscando explicações, exemplos e instruções detalhadas para resolver exercícios. A maioria dos estudos utilizou a versão GPT-3.5, enquanto apenas três mencionaram o uso do GPT-4, o que pode ser explicado pelo fato de o GPT-3.5 estar disponível gratuitamente no período analisado, ao passo que o GPT-4 exigia assinatura paga. Além do ChatGPT, foram identificados estudos que utilizaram o GitHub Copilot, presente em seis artigos, o Codex, citado em um artigo, e o uso de ferramentas próprias desenvolvidas foi citado em três artigos. De modo geral, essas ferramentas foram empregadas para auxiliar os alunos na geração de código, correção de erros e fornecimento de *feedback* automático.

Os estudos revisados concentraram-se predominantemente em cursos de graduação,

com exceção de dois trabalhos aplicados ao Ensino Médio, sendo um deles integrado ao curso técnico de informática. Embora nem todos os artigos tenham especificado as disciplinas envolvidas, as mencionadas foram Introdução à Programação (CS1), Programação Orientada a Objetos e Cursos de Extensão em Programação. Alguns experimentos abrangeram múltiplas disciplinas de programação, em diferentes turmas e níveis.

Em relação às linguagens de programação, Python foi a mais utilizada (em 13 estudos), seguida por C/C++ (8 estudos), Java (7 estudos), e JavaScript, C#, além de linguagens de programação web. Em alguns artigos, a linguagem de programação utilizada não foi especificada.

Buscando responder à segunda questão de pesquisa, procurou-se analisar os artigos para identificar as metodologias ou estratégias pedagógicas utilizadas em conjunto com ferramentas de IA generativa no ensino de programação. Nos estudos de Prather *et al.* (2024), Yilmaz e Yilmaz (2023), Silva, Carlos *et al.* (2024), Karnalim *et al.* (2023), Haindl e Weinberger (2024), Yilmaz e Karaoglan Yilmaz (2023), Jing *et al.* (2024), Sun *et al.* (2024), Silva Junior *et al.* (2023) e Lira *et al.* (2024), não foi explicitada uma abordagem pedagógica formal para a intervenção. Nesses casos, os alunos trabalharam diretamente na resolução de problemas de programação, sendo observadas suas interações com a ferramenta. Ainda que não declarada, percebe-se proximidade com a ABP, metodologia em que o foco recai sobre a atuação ativa dos estudantes na resolução de desafios e na construção do conhecimento.

Nos estudos de Dos Santos e Cury (2023), Jonsson e Tholander (2022) e Banić, Konecki e Konecki (2023), a ferramenta de IA generativa foi utilizada como colega virtual por meio da programação em pares, prática em que dois programadores colaboram em um mesmo projeto utilizando um único ambiente de trabalho. Essa metodologia tem sido adotada como estratégia de ensino colaborativo, especialmente em cursos introdutórios de programação.

O estudo de Lelli *et al.* (2024) utilizou a metodologia POGIL em conjunto com o ChatGPT. O POGIL (*Process-Oriented Guided Inquiry Learning*) é uma estratégia colaborativa baseada em investigação guiada e resolução de problemas, na qual os estudantes exploram questões, discutem em grupo e assumem papéis específicos para potencializar a aprendizagem. Essa abordagem incentiva participação ativa, formulação de hipóteses e aplicação de conhecimentos em problemas mais complexos.

Hu, Assadi e Mahroeian (2023) mencionam a utilização de métodos de

desenvolvimento de software, como prototipagem e desenvolvimento orientado a testes (TDD), no contexto da criação de um aplicativo de negócios utilizando C# e apoio do ChatGPT. Ainda que não explicitado pelos autores, percebe-se proximidade com a Aprendizagem Baseada em Projetos, abordagem que envolve os estudantes na investigação e desenvolvimento de soluções mais amplas e contextualizadas.

Para responder à terceira questão da pesquisa, foram analisados os principais resultados e impactos observados nas experiências de ensino e aprendizagem com auxílio de IA generativa. De modo geral, diversos estudos relataram melhoria na resolução de problemas de programação, aumento da precisão nas respostas e redução do tempo de execução das tarefas. Também foram identificados impactos positivos na motivação dos estudantes, no desenvolvimento do pensamento crítico, na consolidação da aprendizagem e na personalização do ensino. Alguns trabalhos ainda destacaram estímulo à criatividade e à experimentação de novas soluções.

Silva Junior *et al.* (2023), por exemplo, destacaram melhoria na resolução de questões de programação, com percentuais variáveis de sucesso conforme o tipo de problema. Lira *et al.* (2024) observaram que o GitHub Copilot influenciou positivamente o percentual de respostas corretas, com 65% de acertos entre os participantes que utilizaram a ferramenta, em comparação a 43% entre aqueles que não a utilizaram. O tempo médio de resolução também foi significativamente menor, sendo inferior à metade do tempo gasto pelo grupo sem apoio da ferramenta. Além disso, todos os participantes conseguiram resolver os problemas com o Copilot, enquanto três estudantes não enviaram resposta em uma das tarefas quando não dispunham do recurso. Esses resultados indicam impacto positivo tanto na precisão quanto na eficiência da resolução de problemas. O estudo de Yilmaz e Yilmaz (2023) também corrobora esses achados, apontando benefícios do uso do ChatGPT no apoio à resolução de problemas, na compreensão de conceitos complexos e no aumento da produtividade durante as atividades de programação.

Ainda entre os principais resultados, destacam-se o aumento da motivação dos alunos no aprendizado de programação, o desenvolvimento do pensamento crítico, a consolidação do conhecimento adquirido e a facilitação da aprendizagem personalizada, adaptada às necessidades individuais. Alguns autores também ressaltam a importância de equilibrar o uso da automação com a intervenção criativa dos estudantes, promovendo uma abordagem mais ampla da aprendizagem. Em determinados casos, dificuldades e imprevisibilidades na interação com a IA foram percebidas como oportunidades para explorar soluções criativas e

gerar novas ideias.

A última questão da pesquisa buscou identificar os principais desafios enfrentados nas experiências de ensino e aprendizagem de programação com apoio de ferramentas de IA generativa. Entre os aspectos mais recorrentes, destacam-se respostas variáveis ou excessivamente complexas (KARNALIM *et al*., 2023; SILVA JUNIOR et al., 2023; LELLI et al., 2024; LIRA et al., 2024; SANTOS; CURY, 2023; PRATHER et al., 2024), comportamento inconsistente dos sistemas, geração de código impreciso ou com erros (JONSSON; THOLANDER, 2022; SUN et al., 2024; YILMAZ; YILMAZ, 2023; HAINDL; WEINBERGER, 2024; DOS SANTOS; CURY, 2023; PRATHER et al., 2024), necessidade de treinamento prévio dos estudantes (JING et al., 2024; LELLI et al., 2024), risco de dependência excessiva, preocupações éticas relacionadas a plágio e integridade acadêmica (HU et al., 2023; SUN et al., 2024; KARNALIM et al., 2023; YILMAZ; YILMAZ, 2023; SILVA, CARLOS et al., 2024; HAINDL; WEINBERGER, 2024; PRATHER et al., 2024), além da necessidade de supervisão docente (SUN et al., 2024; DOS SANTOS; CURY, 2023). Também foi apontada a necessidade de adaptação das metodologias de ensino para garantir o desenvolvimento do pensamento crítico e da capacidade de resolução de problemas, competências que não devem ser substituídas pela IA (HU et al., 2023; BANIĆ; KONECKI; KONECKI, 2023; YILMAZ; KARAOGLAN YILMAZ, 2023; SILVA JUNIOR et al., 2023; SILVA, CARLOS et al., 2024).

Esses pontos evidenciam que a integração de tecnologias baseadas em IA no ensino de programação oferece oportunidades relevantes, mas exige implementação cuidadosa e pedagogicamente orientada para maximizar benefícios e minimizar riscos. A seguir, destacam-se algumas contribuições dos estudos analisados.

Yilmaz e Karaoglan Yilmaz (2023) investigaram o impacto do uso de ferramentas de IA generativa no ensino de programação, com foco no pensamento computacional, autoeficácia em programação e motivação dos alunos. A pesquisa foi conduzida com grupo experimental utilizando ChatGPT e grupo de controle com métodos tradicionais, em delineamento com pré-teste e pós-teste. Os resultados mostraram melhorias significativas no grupo experimental nas três dimensões analisadas. Além disso, o uso do ChatGPT impactou positivamente criatividade, pensamento algorítmico, cooperação, resolução de problemas e habilidades em tarefas de programação. Os autores ressaltam a necessidade de novas pesquisas em diferentes níveis educacionais, linguagens e contextos acadêmicos.

Prather et al. (2024) investigaram como estudantes novatos interagem com o GitHub

Copilot em um curso introdutório de programação (CS1). A pesquisa, baseada em observações e entrevistas, mostrou que, embora os alunos percebam o Copilot como recurso capaz de acelerar a escrita de código, também expressam preocupações quanto à compreensão do código gerado e à dependência da ferramenta. Foram identificados dois padrões de interação: estudantes que guiavam a ferramenta com suas sugestões e estudantes que seguiam sugestões incorretas, resultando em confusão. Além disso, surgiram desafios cognitivos e metacognitivos relacionados à avaliação das respostas geradas. O estudo conclui com reflexões éticas e recomendações de design voltadas a um uso mais crítico e autônomo.

Já a pesquisa de Dos Santos e Cury (2023) explora as percepções iniciais e oportunidades relacionadas à programação em pares com inteligência artificial, com foco no GitHub Copilot. O trabalho investigou como os desenvolvedores utilizam a ferramenta, os desafios encontrados e as oportunidades trazidas ao desenvolvimento de software. Utilizou-se abordagem mista, incluindo análise de fóruns, estudo de caso e pesquisa em larga escala. Entre os resultados, destacam-se aumento de produtividade, melhoria da qualidade do código e desafios ligados à confiança, transparência, proveniência do código gerado e risco de exposição de informações sensíveis.

Rahman e Watanobe (2023) investigaram como estudantes novatos utilizam geradores de código baseados em LLMs para resolver tarefas de programação em disciplinas introdutórias e em contextos individualizados de aprendizagem. O estudo analisou 33 aprendizes, entre 10 e 17 anos, que utilizaram Python com apoio de ferramenta baseada no OpenAI Codex. Os resultados mostram que os estudantes recorrem a esses sistemas principalmente para acelerar a escrita de código e obter ajuda em estruturas condicionais, laços e funções. Entretanto, as instruções fornecidas tendem a ser vagas, o que compromete a qualidade das respostas.

Embora o código gerado fosse geralmente funcional, muitos alunos apresentaram dificuldades em compreendê-lo e adaptá-lo. Foram identificadas quatro abordagens de uso: solicitação completa da solução, uso passo a passo, abordagem híbrida e codificação manual. Os autores observaram que estudantes mais ativos e autônomos tendem a apresentar melhor desempenho em testes de retenção, sugerindo que os efeitos da IA dependem do nível de envolvimento cognitivo do aluno.

Fernandez e Cornell (2024) exploraram as implicações do uso de geradores de código baseados em IA entre estudantes iniciantes em programação. Em experimento controlado com 69 alunos, metade teve acesso ao OpenAI Codex durante tarefas de autoria e modificação de

código em Python. Os participantes que utilizaram a ferramenta apresentaram desempenho superior nas tarefas de autoria, com maior progresso, menos erros, mais correções e menor tempo gasto. Nas tarefas de modificação, porém, os grupos tiveram desempenho semelhante. No teste de retenção, realizado uma semana depois, o grupo experimental apresentou desempenho ligeiramente superior, embora sem diferença estatisticamente significativa. Os autores concluem que essas ferramentas podem beneficiar o desempenho inicial, mas exigem uso orientado para evitar dependência e promover autonomia e compreensão.

Styve, Virkki e Naeem (2024) exploraram a incorporação de ferramentas como GitHub Copilot e ChatGPT no ensino superior de programação. O estudo, realizado com 90 alunos, envolveu projeto prático no qual o GitHub Copilot atuou como par de programação e o ChatGPT foi utilizado como apoio em atividades avaliativas. Os resultados indicaram maior consciência dos estudantes sobre potencialidades e limitações da IA, além de avanços nas práticas de pensamento crítico relacionadas à programação. Os autores defendem que, embora úteis, essas ferramentas exigem postura crítica e responsabilidade dos estudantes quanto à qualidade do software produzido.

De modo geral, os estudos indicam um aumento na motivação e no interesse em aprender programação após a utilização da ferramenta de IA generativa. Os autores também observaram elevação na autoeficácia em programação, especialmente em tarefas que envolvem a criação de partes do código e soluções completas, além de contribuições para o desenvolvimento do pensamento computacional. No entanto, os estudantes destacaram que, embora a IA seja uma ferramenta útil, ela não substitui o papel do professor ou tutor, mas atua como complemento ao processo de aprendizagem. Alguns alunos consideraram as sugestões da IA valiosas, mas relataram dificuldades na compreensão de trechos complexos do código gerado pela IA, especialmente aqueles com menor experiência.

Além disso, surgiram dúvidas quanto à correção do código gerado, evidenciando a necessidade de instruções mais precisas para obter respostas confiáveis. Em algumas situações, os modelos apresentaram limitações, como erros lógicos e semânticos, além de, ocasionalmente, fornecerem informações imprecisas. Ainda assim, os estudantes reconheceram a utilidade da IA para oferecer respostas rápidas, auxiliar na depuração de códigos e estimular o pensamento crítico, embora tenham apontado preocupações relacionadas à dependência excessiva e à dificuldade de interpretação das respostas.

3.3 Diferenciais da Pesquisa

Com base nos estudos analisados, observa-se que a maioria das pesquisas não contempla a adoção de uma metodologia pedagógica estruturada para o uso da IA generativa no ensino de programação. Em grande parte dos trabalhos, a abordagem pedagógica não é explicitada, e os alunos são frequentemente expostos a interações diretas com modelos de linguagem de grande escala, sem orientações claras sobre como integrar essas ferramentas ao processo de aprendizagem. Conforme evidenciado nas seções anteriores, apenas uma parcela reduzida dos estudos analisados incorpora metodologias ativas de forma explícita, evidenciando uma lacuna significativa na integração entre IA generativa e práticas pedagógicas estruturadas.

Diante desse cenário, esta pesquisa se diferencia ao propor uma abordagem estruturada para o uso da IA generativa como par de programação, destacando-se pelos seguintes aspectos:

- Integração com metodologias ativas: utilização de abordagens como aprendizagem baseada em problemas e programação em pares, nas quais a IA atua como mediadora do processo de aprendizagem, promovendo um ensino mais dinâmico e centrado no estudante;
- Foco na interação aluno+IA: investigação de como ocorre a colaboração entre estudante e IA no contexto da programação em pares, analisando seus efeitos sobre o raciocínio lógico, a tomada de decisão e a resolução de problemas;
- Personalização do suporte: desenvolvimento de um assistente adaptado ao nível de conhecimento de alunos iniciantes, oferecendo orientações alinhadas à Zona de Desenvolvimento Proximal (ZDP);
- Aplicação de *guardrails* e engenharia de *prompt*: definição de mecanismos para orientar o comportamento da IA, evitando a entrega de soluções completas e incentivando o desenvolvimento do pensamento crítico;
- Avaliação empírica robusta: utilização de abordagem mista, com aplicação de pré e pós-testes, questionários e análise das interações, permitindo avaliar de forma mais controlada o impacto pedagógico da IA.

Dessa forma, esta pesquisa contribui para o avanço do conhecimento ao propor uma

integração intencional entre IA generativa e práticas pedagógicas, superando abordagens predominantemente exploratórias identificadas na literatura.

A análise dos trabalhos relacionados evidencia que, embora a IA generativa apresente grande potencial para apoiar o ensino de programação, sua utilização ainda ocorre, em grande parte, de forma pouco estruturada, sem integração consistente com princípios pedagógicos. Observa-se, ainda, uma lacuna no desenvolvimento de soluções que considerem explicitamente aspectos como mediação pedagógica, personalização do suporte e promoção do pensamento crítico.

Diante dessas limitações, esta pesquisa propõe o desenvolvimento do AIPairCode, um assistente de programação baseado em IA generativa, projetado para atuar como par de programação pedagógico, alinhado a princípios como a Zona de Desenvolvimento Proximal e o método socrático.

No próximo capítulo, são apresentados os procedimentos metodológicos adotados para o desenvolvimento e a avaliação da pesquisa.

4 PROCEDIMENTOS METODOLÓGICOS

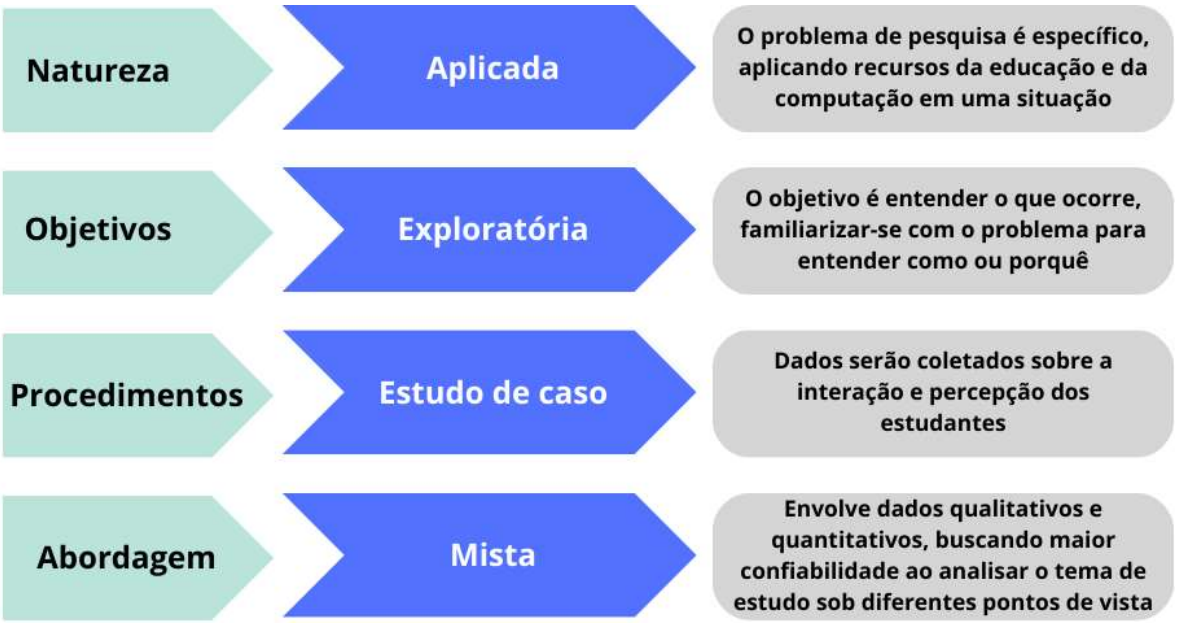
Este capítulo apresenta os procedimentos metodológicos adotados para o desenvolvimento desta pesquisa, incluindo a classificação do estudo, os participantes envolvidos, os instrumentos e procedimentos de coleta e análise de dados, bem como as etapas da investigação.

4.1 Classificação da Pesquisa

Esta pesquisa investiga o uso de ferramentas de Inteligência Artificial generativa como par de programação no ensino de programação, com o objetivo de compreender seu impacto no desempenho, na motivação e no engajamento dos estudantes. Para alcançar esse objetivo, foram conduzidos estudos empíricos em disciplinas introdutórias de programação.

Considerando seus objetivos e características, a pesquisa pode ser classificada conforme apresentado na Figura 5.

Figura 5 - Classificação da pesquisa



Fonte: Elaborado pela autora (2024)

A natureza da pesquisa é caracterizada como aplicada, pois busca gerar conhecimento com aplicação prática voltada à solução de problemas específicos (GIL, 2019). Essa característica se evidencia enquanto o estudo procura contribuir para a compreensão de um problema concreto: as dificuldades enfrentadas pelos alunos na aprendizagem de

programação. Nesse contexto, investiga-se como o uso de uma ferramenta de IA generativa pode atuar como um par de programação, contribuindo para aprimorar o desempenho, a motivação e o engajamento dos estudantes no processo de aprendizagem.

Quanto aos objetivos, a pesquisa possui caráter exploratório, uma vez que busca compreender um fenômeno ainda recente no contexto educacional: o uso de ferramentas de IA generativa no ensino de programação. Estudos exploratórios são apropriados quando o tema investigado ainda apresenta lacunas na literatura ou quando se pretende ampliar a compreensão sobre determinado fenômeno (GIL, 2019).

Em relação aos procedimentos, a investigação foi conduzida por meio de estudos de caso realizados em disciplinas de programação, nos quais foram analisadas as interações dos estudantes com ferramentas de IA generativa, bem como suas percepções sobre o uso dessas tecnologias no processo de aprendizagem. Essa abordagem permite investigar o fenômeno em seu contexto real, possibilitando compreender como ele se manifesta em situações educacionais concretas (PRODANOV; FREITAS, 2013).

Além disso, os estudos empíricos adotaram um delineamento quase-experimental, no qual foram comparadas diferentes condições de uso da ferramenta de IA generativa. Para isso, os estudantes foram organizados em grupos controle e experimental, permitindo analisar o impacto do uso do assistente de programação no desempenho e na motivação dos participantes.

Em relação à abordagem metodológica, este trabalho se caracteriza pela utilização de métodos mistos, combinando dados qualitativos e quantitativos na análise, buscando uma maior confiabilidade ao investigar o tema sob diferentes pontos de vista. A abordagem quantitativa foi empregada na análise do desempenho dos estudantes em avaliações, nos ganhos de aprendizagem e nos resultados de questionários aplicados durante os estudos. Já a abordagem qualitativa foi empregada na análise das interações dos estudantes com a ferramenta de IA e nas respostas abertas dos questionários, buscando compreender suas percepções sobre o uso do assistente de programação.

A combinação dessas abordagens possibilita uma compreensão mais abrangente do fenômeno investigado, permitindo analisar tanto indicadores mensuráveis de aprendizagem quanto aspectos relacionados às experiências e percepções dos participantes.

4.2 Participantes da Pesquisa

Os participantes desta pesquisa são estudantes do curso de Sistemas de Informação, da Universidade Federal de Santa Maria (UFSM), campus de Frederico Westphalen, no estado do Rio Grande do Sul. A instituição foi escolhida por ser uma renomada instituição federal de ensino superior e pela autora da pesquisa atuar como docente nessa instituição, o que possibilitou a realização das intervenções pedagógicas no âmbito das disciplinas de programação do curso.

Ao longo da pesquisa foram conduzidos três estudos empíricos, totalizando a participação direta de 94 estudantes. Todos os participantes eram alunos regularmente matriculados em disciplinas introdutórias de programação do curso de Sistemas de Informação da UFSM, campus de Frederico Westphalen. A quantidade de participantes em cada estudo, bem como aspectos sociodemográficos das amostras são apresentados nas seções 6.1 (Estudo Preliminar), 6.2 (Estudos Final 1) e 6.3 (Estudos Final 2).

Para a realização da pesquisa foi utilizada uma amostra por conveniência, tendo como critério de inclusão a matrícula nas disciplinas de programação alvo do estudo e a disponibilidade dos estudantes de participar voluntariamente.

Todos os participantes foram previamente informados sobre o propósito geral da pesquisa, sem que fossem apresentados os objetivos específicos do estudo, a fim de reduzir possíveis vieses durante a coleta de dados. Também foram esclarecidos quanto ao caráter voluntário de sua participação e a confidencialidade e uso restrito das informações coletadas.

Esses esclarecimentos constam no Termo de Consentimento Livre e Esclarecido (TCLE) (Apêndice I), apresentado aos estudantes antes do início da intervenção didática. O documento visa explicar a pesquisa, proteger os participantes contra possíveis riscos e reafirmar o sigilo e anonimato dos mesmos.

A participação foi voluntária, sendo assegurado aos estudantes o direito de aceitar ou recusar participar da pesquisa. Além disso, foi garantido o anonimato dos participantes, e os dados coletados são utilizados exclusivamente para fins acadêmicos, conforme as diretrizes aprovadas pelo comitê de ética da instituição, assegurando a confidencialidade das informações.

4.3 Fases da Pesquisa

A pesquisa foi organizada em quatro fases principais: (A) Fundamentação, (B) Planejamento e Desenvolvimento, (C) Estudos Empíricos e (D) Análise. Em cada fase diferentes etapas e atividades foram realizadas conforme apresentado na Figura 6.

Figura 6 - Fases da Pesquisa



Fonte: Elaborado pela autora (2024)

Cada uma dessas etapas será detalhada nas seções seguintes.

4.3.1. Fase A: Fundamentação

A primeira fase da pesquisa foi dedicada à fundamentação teórica do estudo, envolvendo a delimitação do problema de pesquisa, a construção do referencial teórico e a análise de trabalhos relacionados. Essas atividades correspondem aos Capítulos 1 (Introdução), 2 (Referencial Teórico) e 3 (Trabalhos Relacionados) desta tese.

Inicialmente realizou-se a delimitação do tema, dos objetivos e da questão de pesquisa, motivada pelas dificuldades observadas pela autora no processo de ensino e aprendizagem de programação. A partir dessas inquietações, foi conduzido um levantamento na literatura com o objetivo de compreender os principais desafios enfrentados por estudantes em disciplinas introdutórias de programação, bem como identificar estratégias pedagógicas e tecnologias que possam contribuir para a superação dessas dificuldades.

Nesse contexto, foram analisados estudos que discutem o insucesso e as dificuldades recorrentes na aprendizagem de algoritmos e programação, destacando-se contribuições de autores como Watson e Li (2014), Bennedsen e Caspersen (2019), Medeiros et al. (2019), Sobral (2021a), Brezolin e Silveira (2021), Morais, 2022, entre outros. Esses estudos apontam que disciplinas introdutórias de programação apresentam elevados índices de dificuldade,

exigindo o desenvolvimento de novas abordagens pedagógicas capazes de apoiar os estudantes durante o processo de aprendizagem.

A construção do referencial teórico contemplou a análise de diferentes perspectivas teóricas relacionadas ao ensino de programação. Foram investigadas teorias de aprendizagem aplicadas ao ensino de programação, com destaque para a teoria sociocultural de Vygotsky (ZDP e mediação) (VYGOTSKY, 1978), método socrático (PAUL; ELDER, 2016), metodologias ativas (BACICH; MORAN, 2017), aprendizagem baseada em problemas (SAVERY, 2006; WILLIAM *et al*., 2019) e programação em pares (OTHMAN *et al.*, 2019).

Além disso, foi realizada uma revisão da literatura sobre Inteligência Artificial generativa, abordando seus fundamentos e sua aplicação na educação, e mais detalhadamente no ensino de programação (ELLIS; SLADE, 2023; VASWANI *et al.*, 2017; ZHAO *et al*., 2023; UNESCO, 2023; LAMATTINA, 2023; MELLO *et al*., 2023; BECKER *et al*., 2023; DENNY *et al.*, 2024).

Para identificar os trabalhos relacionados ao tema desta pesquisa, foram conduzidos dois estudos de revisão da literatura em bases de dados nacionais e internacionais. Inicialmente, foi realizado um Mapeamento Sistemático da Literatura (MSL) com o objetivo de obter uma visão ampla dos estudos na área, analisando e classificando as pesquisas existentes. Os trabalhos selecionados foram organizados em cinco categorias, segundo o foco e a utilização das ferramentas de IA generativa: (a) avaliação do desempenho, capacidades e limitações das ferramentas, (b) desenvolvimento de ferramentas, (c) entrevistas/questionários, (d) pesquisas exploratórias com alunos utilizando ferramentas de IA generativa no ensino de programação, e (e) artigos de opinião e revisão.

Como desdobramento desse mapeamento, foi conduzido um segundo estudo com foco na categoria de pesquisas exploratórias, analisando as estratégias pedagógicas e metodologias de ensino e aprendizagem empregadas nos estudos que envolvem o uso de ferramentas de IA generativa no ensino de programação. Essa análise contribuiu para identificar lacunas na literatura e subsidiar a definição da proposta investigativa desta tese.

4.3.2 Fase B: Planejamento e Desenvolvimento

Na segunda fase da pesquisa foram definidos os instrumentos de coleta de dados, realizados estudos exploratórios iniciais com estudantes e iniciado o desenvolvimento do assistente de programação utilizado nos estudos empíricos.

4.3.2.1 Instrumentos

Inicialmente, foram definidos os instrumentos utilizados para a coleta de dados quantitativos e qualitativos ao longo da investigação. Para a coleta de dados quantitativos, foram aplicados questionários estruturados, pré-testes e pós-testes de conhecimento, além dos registros das atividades de programação desenvolvidas pelos estudantes. Esses instrumentos permitiram acompanhar o progresso dos alunos ao longo da disciplina e comparar seu desempenho antes e após a intervenção com a ferramenta de IA.

Para os dados qualitativos, as interações com a ferramenta de IA foram documentadas e perguntas abertas foram aplicadas aos alunos após o uso da ferramenta, visando coletar dados sobre suas percepções em relação à IA como par de programação, bem como identificar padrões de interação durante a resolução dos exercícios.

A coleta de dados foi realizada em três fases. Na primeira fase, os alunos responderam a questionários iniciais para coletar dados demográficos e avaliar seu conhecimento prévio (pré-teste). Em seguida, na segunda fase, durante o semestre, os alunos utilizaram o assistente de IA como par de programação para resolver problemas e realizar exercícios práticos. O uso da ferramenta foi monitorado por meio de *logs* de interações e acompanhamento das atividades realizadas. Na terceira fase, ao final do semestre, foi aplicado um questionário pós-teste para avaliar a eficácia percebida no aprendizado, um questionário para identificar a motivação, e também foi avaliada a usabilidade do assistente de programação baseado em IA.

Os questionários foram disponibilizados *online* por meio da plataforma Google Formulários e aplicados aos participantes em diferentes momentos da pesquisa. O questionário demográfico teve como objetivo identificar o perfil dos estudantes e sua autoavaliação em relação às temáticas abordadas na disciplina, sendo aplicado no início da intervenção. Esse instrumento (Apêndice II) é composto por 17 perguntas de múltipla escolha e 1 questão aberta, permitindo que os participantes expressem suas opiniões.

Para avaliar a motivação dos estudantes no contexto do uso de uma ferramenta de IA como par de programação, foi utilizado um questionário baseado no *Intrinsic Motivation Inventory* (IMI) (DECI; RYAN, 2000), fundamentado na Teoria da Autodeterminação. O instrumento (Apêndice III), é composto por 25 afirmações avaliadas em escala *Likert*[7] de sete pontos, variando de “1 = Discordo totalmente” até “7 = Concordo totalmente”.

[7] A escala *Likert* é um método proposto, em 1932, pelo cientista social estadunidense Rensis Likert (1903-1981), como uma forma de medir opiniões, percepções e comportamentos.

Além disso, também foi aplicado o questionário *SUS - System Usability Scale* (BROOKE, 1996) com 10 questões avaliadas em escala *Likert* de cinco pontos, variando de "1 = Discordo totalmente" até "5 = Concordo totalmente", cujo o objetivo é avaliar a usabilidade do assistente de programação desenvolvido nesta pesquisa (Apêndice IV).

Nos estudos finais da pesquisa, os participantes foram organizados em dois grupos, controle e experimental, permitindo comparar o impacto do uso do assistente de programação no desempenho dos estudantes. Para ambos os grupos foram aplicados pré-testes e pós-testes ao longo dos experimentos possibilitando analisar o ganho de aprendizagem decorrente das atividades realizadas.

Além dos questionários, os exercícios desenvolvidos pelos estudantes e os registros das sessões de interação com a ferramenta de IA foram analisados como material de coleta de dados, permitindo identificar evidências de aprendizagem relacionadas aos conteúdos de programação.

4.3.2.2 Estudo Preliminar

Como parte desta fase de planejamento, foi realizado um estudo preliminar com o objetivo de investigar as percepções e interações dos alunos ao utilizarem uma ferramenta de IA generativa de propósito geral como par de programação. O estudo buscou compreender como os estudantes utilizam essas ferramentas durante a resolução de diferentes problemas de programação. Entre os objetivos específicos do estudo preliminar, destacam-se:

- Identificar as características presentes nas solicitações de auxílio feitas pelos alunos às ferramentas;
- Analisar os padrões de uso das ferramentas de IA, considerando aspectos como a frequência de interações e a natureza das perguntas formuladas pelos alunos; e
- Identificar a motivação e percepção dos estudantes ao utilizar a IA como par de programação.

Esta etapa da pesquisa foi conduzida entre setembro e outubro de 2024, com estudantes da disciplina de Programação e Estrutura de Dados II do curso de Sistemas de Informação da Universidade Federal de Santa Maria (UFSM), campus de Frederico Westphalen. A disciplina foi realizada de forma presencial e contou com 49 alunos matriculados, dos quais 37 participaram ativamente na atividade de pesquisa.

Os alunos participantes eram estudantes do segundo semestre do curso e estavam iniciando o aprendizado da linguagem de programação C++. Todos haviam tido contato prévio com a linguagem de programação Python no primeiro semestre da graduação.

Para não interferir no andamento regular da disciplina, foi realizada uma intervenção didática composta por uma atividade extracurricular. A atividade incluiu exercícios de programação em C++, abordando tópicos como variáveis, estruturas de decisão e repetição, e *arrays*, organizadas em nível crescente de dificuldade.

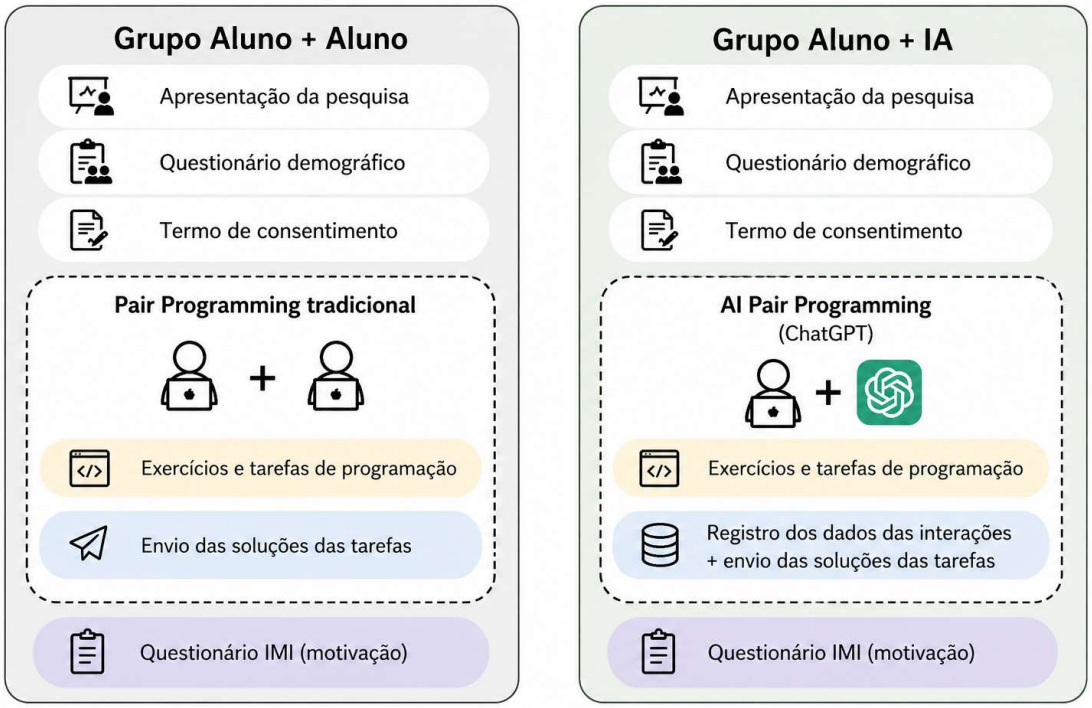
Os exercícios foram elaborados com base na abordagem de aprendizagem baseada em problemas (ABP), na qual os estudantes deveriam interpretar situações-problema e propor soluções utilizando conceitos de programação. Nesse contexto, cada problema exigia que os alunos aplicassem suas habilidades de análise para identificar as necessidades e restrições do problema, selecionassem as estruturas de programação adequadas e implementassem uma solução funcional.

Durante a atividade, foi utilizada a ferramenta ChatGPT como par de programação. Os estudantes foram divididos em dois grupos: um grupo realizou as atividades utilizando um par de programação humano (colega) e o outro utilizou a IA como par de programação. A alocação dos alunos nos grupos foi feita com base nas notas da disciplina de Programação e Estruturas de Dados I cursada no semestre anterior.

As atividades ocorreram em laboratório de informática, com um computador disponível para cada par de programação, o que permitiu o acompanhamento das atividades pela pesquisadora. O Ambiente Virtual de Ensino e Aprendizagem Moodle foi utilizado como plataforma de apoio para disponibilização dos materiais e coleta de dados.

Após a realização das atividades, foi aplicado um questionário para coletar as percepções dos estudantes sobre a intervenção e o uso da ferramenta de IA durante a resolução dos exercícios. As etapas do estudo preliminar são apresentadas na Figura 7.

Figura 7 - Etapas do Estudo Preliminar



Fonte: Elaborado pela autora (2024)

Os resultados obtidos neste estudo preliminar serviram de base para definir os requisitos e implementar o assistente de programação a ser utilizado nos estudos finais, além de contribuir para o para o refinamento do desenho metodológico da pesquisa.

4.3.2.3 Desenvolvimento do Assistente de Programação

Com base nas evidências obtidas na revisão de literatura e no estudo preliminar, foi iniciado o desenvolvimento de um assistente de programação baseado em IA generativa, projetado para atuar como par de programação no contexto educacional.

Conforme discutido na literatura, ferramentas baseadas em modelos de linguagem de grande escala (LLMs), como o *GitHub Copilot* e o ChatGPT, são capazes de gerar, corrigir e explicar código automaticamente. Embora esses assistentes ofereçam suporte significativo em ambientes de desenvolvimento profissional, eles foram treinados com base em vastos volumes de código provenientes de repositórios públicos, que em sua maioria consistem em projetos de grande porte e soluções profissionais, o que pode resultar em soluções excessivamente complexas para estudantes iniciantes.

Diversos estudos apontam que o uso direto dessas ferramentas por alunos em

disciplinas introdutórias de programação pode gerar códigos difíceis de compreender ou adaptar, uma vez que as respostas frequentemente não consideram o nível de conhecimento dos estudantes (KARNALIM *et al.*,2023; SILVA JUNIOR *et al.*,2023; LELLI *et al.*, 2024; LIRA *et al.*, 2024; DOS SANTOS; CURY, 2023; PRATHER *et al.*,2024).

Além disso, os assistentes de programação generalistas não foram projetados especificamente para contextos educacionais, carecendo de mecanismos que promovam o desenvolvimento gradual das habilidades dos estudantes, como a apresentação de soluções progressivas, *feedback* estruturado e explicações adaptadas ao nível de conhecimento do aprendiz.

Diante destas limitações, optou-se pelo desenvolvimento de um assistente personalizado, concebido para atuar como um par de programação pedagógico, alinhado aos princípios da Teoria Sociocultural de Vygotsky, especialmente ao conceito de ZDP. Nesse contexto, o assistente foi projetado para fornecer orientações graduais, explicações conceituais e exemplos parciais de solução, evitando a entrega direta de respostas completas sempre que possível.

Assim, o desenvolvimento do assistente teve como objetivo criar uma ferramenta educacional capaz de atuar como mediador pedagógico, ajustando-se às necessidades específicas de alunos iniciantes. Esse tipo de ferramenta tem o potencial de preencher lacunas que as soluções generalistas não conseguem abordar, oferecendo uma experiência mais alinhada aos objetivos de ensino de programação e favorecendo a construção de conhecimento de forma incremental e acessível.

Os detalhes do desenvolvimento do assistente de programação proposto nesta tese são apresentados no capítulo 5.

4.3.3. Fase C: Estudos Empíricos

A terceira fase da pesquisa correspondeu à implementação dos estudos empíricos no contexto das disciplinas de programação. Nessa etapa foram realizadas as intervenções pedagógicas que permitiram avaliar o uso do assistente de programação baseado em IA como par de programação no processo de aprendizagem.

Foram realizados dois estudos finais com estudantes do curso de Sistemas de Informação da Universidade Federal de Santa Maria (UFSM), campus de Frederico

Westphalen. As intervenções ocorreram em duas disciplinas de programação: Tópicos Especiais em Programação: Introdução a Programação Web com JavaScript e Programação e Estrutura de Dados II, ofertadas no primeiro e segundo semestre de 2025, respectivamente.

Para a realização dos estudos foi utilizada uma amostra por conveniência composta por estudantes matriculados nas disciplinas mencionadas anteriormente. Em ambos os estudos os alunos foram organizados em dois grupos: um grupo experimental, que utilizou o assistente de programação baseado em IA durante as atividades práticas, e um grupo de controle, que realizou as mesmas atividades sem o uso da ferramenta. Essa divisão permitiu comparar os resultados obtidos com a estratégia pedagógica proposta em relação ao método tradicional de ensino.

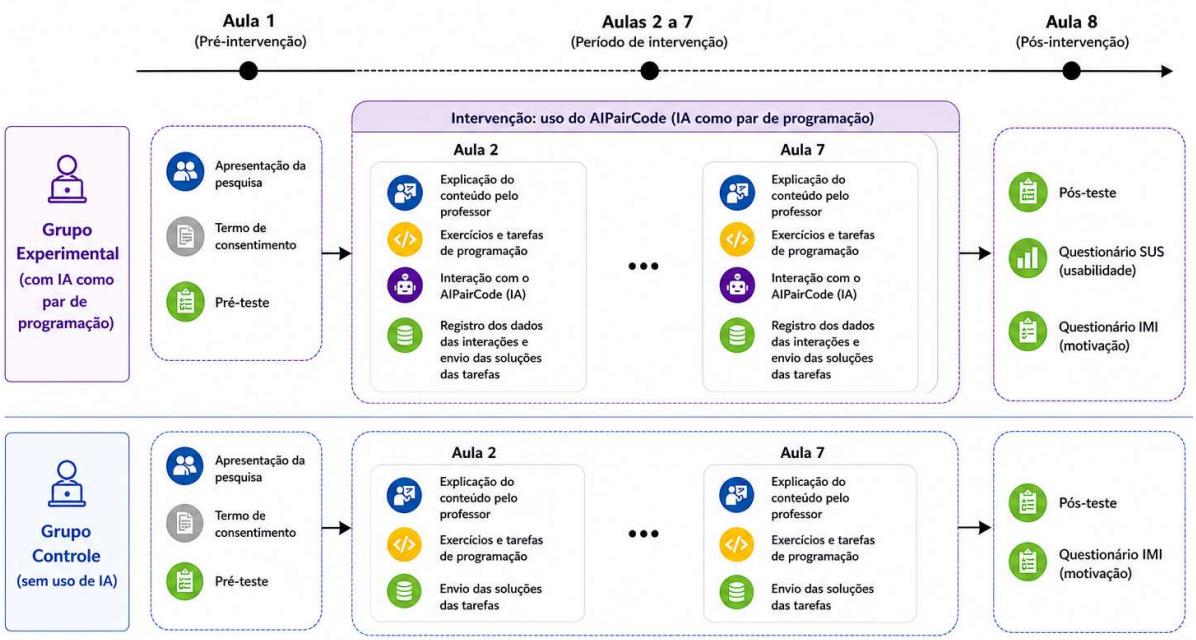
Embora cada estudo possui características específicas relacionadas à disciplina e aos conteúdos abordados, as intervenções seguiram um roteiro geral semelhante. Inicialmente, os estudantes foram informados sobre os procedimentos da pesquisa e convidados a participar voluntariamente da investigação. Em seguida, foi aplicado um pré-teste com o objetivo de avaliar o nível de conhecimento prévio dos participantes.

Durante o período de intervenção, os estudantes realizaram atividades práticas de programação ao longo das aulas da disciplina. No grupo experimental, essas atividades foram desenvolvidas com o apoio do assistente de IA, que atuou como um par de programação, oferecendo orientações, sugestões e *feedback* durante a resolução dos exercícios. As interações dos alunos com o assistente foram registradas automaticamente por meio de *logs* do sistema, permitindo acompanhar o processo de uso da ferramenta.

Ao final do período de intervenção, foi aplicado um pós-teste com o objetivo de avaliar o desempenho dos estudantes após a realização das atividades, bem como um questionário de motivação baseado no *Intrinsic Motivation Inventory* (IMI). No grupo experimental, também foi aplicado um instrumento adicional para avaliar aspectos relacionados ao uso da ferramenta: o *System Usability Scale (SUS)*, utilizado para avaliar a usabilidade do assistente.

A Figura 8 apresenta o desenho geral da intervenção realizada no estudo, indicando a sequência das atividades desenvolvidas ao longo das aulas para os grupos experimental e de controle, bem como os momentos de aplicação dos instrumentos de coleta de dados.

Figura 8 - Roteiro básico dos estudos do grupo experimental e controle



Fonte: Elaborado pela autora (2024)

De forma geral, a coleta de dados no grupo experimental envolveu:

- Aplicação do questionário demográfico;
- Aplicação do pré-teste para avaliação do conhecimento prévio dos estudantes;
- Realização das atividades práticas de programação com apoio do assistente de IA;
- Registro automático das interações dos alunos com a ferramenta;
- Coleta das soluções dos exercícios desenvolvidos pelos estudantes;
- Aplicação do pós-teste ao final da intervenção;
- Aplicação de questionários adicionais para avaliar motivação e usabilidade da ferramenta.

No grupo de controle, os estudantes participaram das mesmas atividades práticas de programação, porém sem o uso do assistente de IA. Nesse caso, a coleta de dados incluiu:

- Aplicação do questionário demográfico;
- Aplicação do pré-teste;
- Realização das atividades de programação propostas na disciplina;
- Coleta das soluções dos exercícios desenvolvidos pelos estudantes;
- Aplicação do pós-teste ao final da intervenção;

- Aplicação do questionário para avaliar motivação.

A descrição detalhada dos procedimentos específicos de cada estudo, bem como dos instrumentos utilizados e das etapas de coleta de dados, é apresentada no capítulo 6.

4.3.4. Fase D: Análise dos Dados

Na quarta e última fase, os dados coletados ao longo do estudo foram organizados e analisados. A análise de dados de uma pesquisa consiste em extrair os significados mais relevantes de textos e imagens obtidos por meio dos instrumentos de coleta de dados. Esse processo envolve preparar, organizar, aprofundar, representar e interpretar os dados para a sua análise. Considerando a abordagem metodológica de métodos mistos adotada nesta tese, foram realizadas análises quantitativas e qualitativas, posteriormente integradas por meio de triangulação (CRESWELL, 2012).

No que se refere a análise quantitativa, foram inicialmente realizadas análises por meio de estatística descritiva dos dados obtidos nos instrumentos aplicados aos participantes. Foram calculadas medidas como média, mediana e desvio padrão, com o objetivo de caracterizar o desempenho dos estudantes nos pré-testes e pós-testes realizados durante os estudos. Para avaliar a evolução do desempenho dos participantes ao longo das intervenções, foi calculado o ganho de aprendizagem, obtido a partir da diferença entre os resultados do pós-teste e do pré-teste. Esse indicador permitiu analisar o progresso individual e coletivo dos estudantes ao longo das atividades realizadas.

A escolha pela análise do ganho de aprendizagem foi adotada como estratégia para comparar a evolução dos estudantes ao longo da intervenção, uma vez que os participantes apresentavam níveis iniciais de conhecimento distintos. Dessa forma, o uso do ganho permitiu avaliar a variação no desempenho dos alunos ao longo do período analisado, possibilitando uma comparação mais adequada entre os grupos.

Além da análise descritiva, foram aplicados testes estatísticos inferenciais com o objetivo de verificar diferenças estatisticamente significativas entre os resultados obtidos pelos grupos. Inicialmente, foram verificados os pressupostos estatísticos de normalidade e homogeneidade de variâncias. Para avaliar a normalidade da distribuição dos dados foi utilizado o teste de *Shapiro–Wilk*, enquanto a homogeneidade das variâncias entre os grupos foi verificada por meio do teste de *Levene*. Quando os pressupostos para a aplicação de testes paramétricos foram atendidos, foi utilizado o teste *t de Student* para amostras independentes

para comparar os resultados entre os grupos controle e experimental. Nos casos em que os pressupostos não foram satisfeitos, foram utilizados testes não paramétricos equivalentes, como o teste *U de Mann–Whitney*. Além disso, foram calculadas medidas de tamanho de efeito, como o *d de Cohen* e o coeficiente *r*, permitindo estimar a magnitude das diferenças observadas entre os grupos.

Também foram analisados os dados provenientes dos questionários aplicados aos participantes ao longo da pesquisa. Para avaliar a motivação dos estudantes foi utilizado um questionário adaptado baseado no *Intrinsic Motivation Inventory (IMI)*, cujos resultados foram analisados por meio de estatística descritiva e testes comparativos entre os grupos, considerando as diferentes dimensões do instrumento. Da mesma forma, os dados obtidos por meio do *System Usability Scale (SUS)* foram analisados com o objetivo de avaliar a usabilidade percebida do assistente de programação pelos participantes do grupo experimental.

Além das análises relacionadas ao desempenho e aos questionários, também foram examinados os registros das interações dos estudantes com o assistente de programação ao longo das atividades. Essas interações foram registradas automaticamente por meio de *logs* do sistema e posteriormente organizadas para análise. Inicialmente, foi realizada uma análise quantitativa da frequência de uso da IA por cada aluno, buscando identificar quantas vezes a ferramenta foi utilizada, durante quanto tempo e quantas interações foram realizadas por exercício.

Paralelamente às análises quantitativas, também foi realizada uma análise qualitativa dos dados coletados. Os registros das interações foram analisados por meio da técnica de Análise de Conteúdo, proposta por Bardin (2016). Esse procedimento envolveu três etapas principais: a pré-análise, que consiste na organização e leitura inicial do material coletado; a exploração do material, na qual os dados são codificados e categorizados; e o tratamento e interpretação dos resultados, etapa em que são identificados temas e padrões recorrentes nas solicitações dos participantes.

A partir dessa análise de conteúdo foi feita a categorização das solicitações feitas pelos estudantes ao assistente, considerando o tipo de apoio buscado durante a resolução dos exercícios de programação. Em seguida, foram analisados os tipos de respostas fornecidas pela IA, buscando identificar as estratégias adotadas pelo assistente ao interagir com os estudantes. A partir dessa categorização, foram calculadas as frequências absolutas e relativas das diferentes categorias identificadas, permitindo mapear padrões de interação e

compreender de que forma o assistente foi utilizado durante o processo de resolução de problemas.

As respostas abertas dos questionários também foram analisadas qualitativamente por meio da análise de conteúdo. A aplicação dessa técnica permitiu identificar as percepções dos estudantes sobre o uso da IA como par de programação, bem como compreender como os participantes avaliaram o suporte oferecido pela ferramenta durante as atividades. Essa análise possibilitou interpretar aspectos da experiência de uso que não poderiam ser plenamente capturados apenas por meio de indicadores quantitativos.

Por fim, os resultados das análises quantitativas e qualitativas foram integrados por meio de triangulação dos dados. Esse procedimento permitiu relacionar diferentes fontes de evidência da pesquisa, incluindo desempenho acadêmico, padrões de interação com o assistente e percepções dos estudantes sobre o uso da ferramenta. A triangulação possibilitou identificar convergências e divergências entre os resultados obtidos por diferentes instrumentos e métodos de análise, contribuindo para uma compreensão mais abrangente do uso da IA como par de programação e para a avaliação do artefato proposto na pesquisa, permitindo analisar seu impacto no processo de aprendizagem e na experiência dos estudantes.

As análises estatísticas foram realizadas com auxílio do software *Jamovi*[8], enquanto os dados qualitativos foram organizados e analisados com apoio de planilhas eletrônicas e do software *ATLAS.ti*[9]. Todos os dados coletados durante a pesquisa foram armazenados em ambiente digital seguro sob responsabilidade da pesquisadora, sendo mantidos de forma anônima e confidencial e utilizados exclusivamente para fins acadêmicos. Os registros permanecerão armazenados pelo período de cinco anos e posteriormente serão excluídos.

Com a definição dos procedimentos metodológicos, dos instrumentos de coleta de dados e das técnicas de análise adotadas nesta pesquisa, o capítulo seguinte apresenta o desenvolvimento do assistente de programação baseado em inteligência artificial utilizado nos estudos finais desta tese.

[8] https://www.jamovi.org/
[9] https://atlasti.com/

5. DESENVOLVIMENTO DO ASSISTENTE DE PROGRAMAÇÃO AIPAIRCODE

Para atender a um dos objetivos desta pesquisa, foi desenvolvida uma ferramenta computacional denominada AIPairCode, cujo nome deriva da combinação de *Artificial Intelligence* (AI), *Pair Programming* e *Code*, refletindo sua proposta de atuar como um par de programação baseado em inteligência artificial. O sistema foi projetado para acompanhar os estudantes durante a resolução dos exercícios e atividades práticas de programação, oferecendo suporte orientado que favorece a compreensão conceitual, o desenvolvimento do raciocínio lógico e a autonomia na resolução de problemas.

O desenvolvimento do assistente foi orientado por princípios pedagógicos derivados da teoria sociocultural de Vygotsky, especialmente os conceitos de zona de desenvolvimento proximal (ZDP) e mediação (VYGOTSKY, 1978). Também foram incorporadas estratégias associadas à dinâmica da programação em pares (WILLIAMS *et al.*, 2000) e ao uso de questionamentos inspirados no método socrático, como forma de estimular o pensamento crítico (PAUL; ELDER, 2016). Assim, o sistema foi planejado não apenas como uma ferramenta de geração de respostas, mas como um mediador pedagógico voltado à promoção da autonomia, da reflexão e da construção gradual do conhecimento.

A ZDP orientou o desenho das interações, priorizando a oferta de apoio gradual que possibilitasse ao estudante avançar além de seu desempenho independente, mobilizando processos cognitivos em desenvolvimento e preservando seu papel ativo na resolução dos problemas. O método socrático foi incorporado de forma simplificada, por meio de perguntas investigativas, incentivo à formulação de hipóteses e proposição de pequenos desafios intermediários, estimulando o raciocínio do aluno.

Complementarmente, a alternância entre os papéis de navegador e piloto, inspirada na prática de programação em pares, reforçou a dimensão colaborativa do processo. Em determinados momentos, o assistente adota a postura de navegador, instigando a análise conceitual e estratégica, enquanto em outros assume o papel de piloto, oferecendo trechos curtos de código, sugestões de correção ou explicações pontuais.

A concepção do AIPairCode também buscou incorporar princípios de *scaffolding* (andaime) educacional, oferecendo diferentes níveis de suporte ao estudante durante o processo de resolução de problemas. Em vez de fornecer diretamente a solução completa de um exercício, o assistente foi configurado para apresentar dicas progressivas, explicações

conceituais e exemplos parciais, incentivando o estudante a participar ativamente da construção da solução.

O escalonamento da ajuda foi estruturado em níveis progressivos de mediação:

- Nível 1 (Orientação e Diagnóstico): o assistente limita-se a fornecer uma orientação conceitual, identificando o problema e oferecendo explicações gerais ou localizadas, como indicar a linha de código com erro ou apontar a ausência de inicialização de uma variável. Nessa etapa, a ênfase recai sobre questionamentos breves, inspirados no método socrático, que incentivam o estudante a refletir sobre seu próprio código e a propor caminhos de solução.
- Nível 2 (Pistas Estruturadas): caso o estudante não consiga avançar ou solicite apoio adicional, o assistente passa a oferecer descrições de algoritmos, fluxos lógicos ou representações em pseudocódigo. O objetivo é reduzir a sobrecarga cognitiva, sem retirar do estudante a responsabilidade de implementar a solução.
- Nível 3 e 4 (Exemplos de Código): somente após sucessivas tentativas sem êxito ou a partir de um pedido explícito do estudante é que o sistema atinge esse nível, no qual são apresentados exemplos de código. Esses exemplos se limitam a trechos curtos e localizados (geralmente de três a cinco linhas), sempre restritos ao ponto em discussão, e nunca contemplam a solução completa do programa.

A progressão entre os níveis é guiada por dois critérios principais: as tentativas realizadas pelo estudante sem alcançar avanço técnico e os pedidos explícitos por exemplos mais objetivos. Essa lógica assegura que o suporte seja responsivo às necessidades individuais, mas evita que a ferramenta se torne apenas um fornecedor de soluções prontas.

Para que esta estratégia pedagógica fosse possível do ponto de vista técnico, o AIPairCode foi estruturado em uma arquitetura modular, integrada ao ambiente de desenvolvimento do aluno e capaz de acompanhar suas atividades em tempo real. A implementação desse mecanismo de mediação envolveu o uso de *guardrails* e engenharia de *prompt* para limitar o tipo de resposta da IA, além de técnicas de *Retrieval-Augmented Generation* (RAG) para fundamentar base de conhecimento da ferramenta com os materiais didáticos da disciplina. Os detalhes dessa arquitetura, bem como os componentes de registro de dados para a pesquisa e as interfaces de interação, são descritos nas seções a seguir.

5.1 Arquitetura Geral do Sistema

O AIPairCode é um assistente de programação, baseado em APIs de Inteligência Artificial generativa, projetado para apoiar estudantes no processo de aprendizagem de linguagens de programação, atuando como um par de programação. Seu funcionamento é fundamentado em abordagens pedagógicas como programação em pares, método socrático e ZDP.

O assistente interage com o aluno durante a resolução de problemas, auxiliando na interpretação de enunciados, explicação de conceitos e trechos de código, identificação de erros e proposição de sugestões que estimulam pensamento crítico.

Para viabilizar a experiência pedagógica proposta e permitir a coleta de dados para esta pesquisa, a interface do sistema foi estruturada em três módulos principais, complementares e integrados entre si:

1. Extensão para Visual Studio Code (VS Code): representa uma das interfaces diretas do estudante com o assistente, permitindo o uso da ferramenta no próprio ambiente de desenvolvimento. A extensão inclui uma área de *chat* integrada ao editor de código, leitura automática do enunciado inserido no arquivo, envio do código para análise da IA, sugestões *inline* (quando no modo piloto) e botões de ações rápidas, como explicar trecho, corrigir trecho e sugerir próximo passo, acompanhando o processo de programação em tempo real.
2. Módulo Web do Estudante: oferece um ambiente centralizado para organização e acompanhamento das atividades. Permite criar e registrar exercícios, utilizar uma interface de *chat* integrada a um editor de código web, enviar o código atual para análise da IA e realizar testes automatizados com correção. Além disso, possibilita acessar mini-desafios e exercícios extras gerados dinamicamente, visualizar o histórico completo de interações, receber diagnósticos automáticos sobre dificuldades recorrentes, baixar materiais da disciplina usados como base (RAG) e acompanhar gráficos de progresso e desempenho.
3. Módulo Web do Professor: funciona como uma central de acompanhamento pedagógico. Neste painel, o docente pode visualizar métricas de interação dos alunos, identificar dificuldades mais frequentes por tema, analisar desempenho por exercício e por período, baixar relatórios em formato CSV, acompanhar estatísticas agregadas da

turma, enviar os materiais da disciplina utilizados no RAG e monitorar uso da ferramenta pelos alunos.

A integração entre esses três módulos ocorre por meio de uma infraestrutura organizada em três camadas principais: as interfaces (extensão *VS Code*, módulo web do estudante e módulo web do professor), o servidor *backend* e o banco de dados em nuvem.

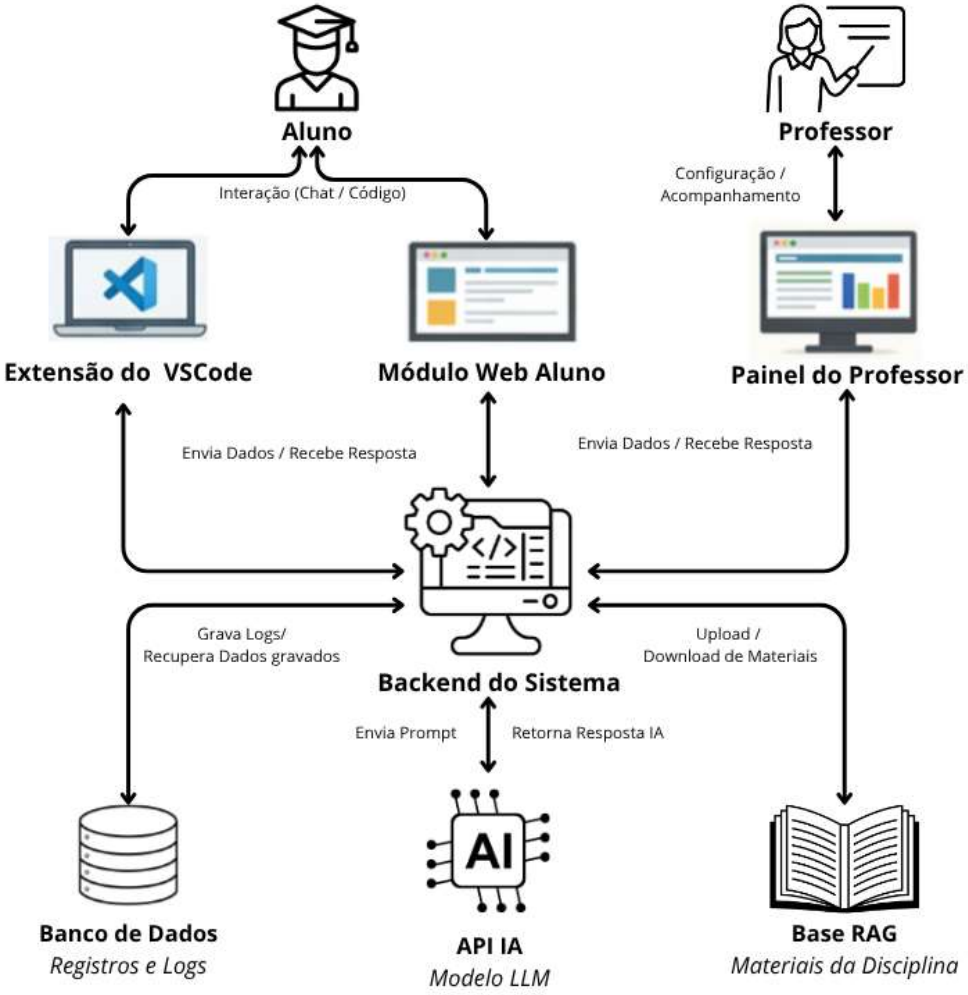
As camadas de interface foram desenvolvidas utilizando Next.js e React no *frontend*, com interface implementada em TypeScript e Tailwind CSS. O *backend* foi desenvolvido em Node.js utilizando o *framework* Express, sendo integrado ao Firebase Authentication, Firestore Database e Firebase Storage para gerenciamento de autenticação, armazenamento de dados e arquivos da aplicação.

O *backend*, atua como elemento central da arquitetura intermediando as comunicações entre os diferentes módulos do sistema. Essa camada é responsável por receber as requisições dos usuários, processar o contexto das interações, aplicar os mecanismos de engenharia de *prompt* e *guardrails* e encaminhar as solicitações ao modelo de IA. Além disso, gerencia a lógica de negócio do sistema, incluindo controle de sessões, definição do papel do assistente (Navegador ou Piloto), cálculo do nível de suporte (ZDP) e registro das interações realizadas. O *backend* também se conecta à *Assistants API* da OpenAI para o processamento das interações e ao Firebase para autenticação e armazenamento dos dados.

A camada de persistência de dados foi implementada utilizando os serviços Firebase *Authentication,* Firestore *Database* e Firestore *Storage*, responsáveis por gerenciar o acesso dos usuários, armazenar de forma estruturada os exercícios, metadados e históricos de interação, além de armazenar os arquivos utilizados na base de conhecimento (RAG). Esses dados são fundamentais tanto para o funcionamento do sistema quanto para a análise posterior das interações no contexto da pesquisa.

A Figura 9 apresenta a arquitetura geral do AIPairCode, evidenciando os módulos do sistema, a comunicação entre os componentes e o fluxo de dados.

Figura 9 - Arquitetura Geral do AIPairCode



Fonte: Elaborado pela autora (2025)

O fluxo de dados do sistema é centrado na interação do estudante, que inicia o processo ao enviar uma dúvida ou trecho de código por meio da extensão no VS Code ou do módulo web. Essas requisições são encaminhadas ao *backend,* que atua como gerenciador central, processando a solicitação, definindo o papel atual da IA (navegador ou piloto) e enviando a requisição para a *Assistants* API da OpenAI.

A IA processa a solicitação considerando as instruções definidas via engenharia de *prompt* e consultando os materiais didáticos previamente configurados (RAG). A resposta gerada é então retornada ao *backend*, que realiza a validação por meio de *guardrails* antes de encaminhá-la à interface de origem do estudante.

Simultaneamente, o *backend* realiza o registro dessa interação no banco de dados (Firestore), armazenando os metadados que posteriormente alimentam os painéis de acompanhamento nos módulos do professor e do estudante. Esse fluxo garante o

funcionamento integrado do sistema, a proteção das credenciais da API, o respeito às diretrizes pedagógicas e a rastreabilidade dos dados coletados.

A partir da arquitetura apresentada, foram definidas as tecnologias e estratégias responsáveis pela implementação do assistente de programação. A seguir são descritos os principais recursos utilizados no desenvolvimento do AIPairCode, com destaque para a integração com a *Assistants* API da OpenAI, bem como o uso de técnicas de engenharia de *prompt*, *guardrails* e recuperação de informações (RAG), que viabilizam o comportamento do sistema.

5.1.1 Assistants API

No desenvolvimento do assistente, foi utilizada a *Assistants API da OpenAI*[10], que possibilita a criação de assistentes de IA integrados a aplicações personalizadas. Essa API permite configurar instruções específicas, utilizar diferentes modelos, ferramentas e arquivos para responder às solicitações dos usuários. No contexto do projeto, suas funcionalidades foram exploradas para personalizar o comportamento do assistente de acordo com os objetivos pedagógicos estabelecidos.

A *Assistants API* oferece suporte a diferentes ferramentas, como interpretador de código, pesquisa em arquivos e chamadas de função, que podem ser utilizadas tanto pela interface da OpenAI quanto por meio de integração direta via código. Neste trabalho, optou-se pela integração direta via código, permitindo maior controle sobre o fluxo de interação, sobre o comportamento do assistente e sobre a adaptação das respostas à metodologia adotada.

A arquitetura da API permite configurar modelos com instruções que definem a personalidade e as capacidades do assistente. Um dos recursos mais relevantes para este trabalho é o suporte a *threads* persistentes, que funcionam como sessões de conversa capazes de armazenar o histórico das interações entre o assistente e o usuário, mantendo o contexto de forma automática ao longo do tempo. Dessa forma, não é necessário reenviar todo o histórico de mensagens no *prompt* a cada nova interação, reduzindo o volume de dados transmitidos, a latência das respostas e os custos de processamento. No contexto educacional, essa característica permite que o assistente acompanhe a evolução do estudante durante as atividades, adaptando seu suporte conforme o progresso e as necessidades individuais.

Além disso, a API oferece suporte ao gerenciamento e manipulação de arquivos em

[10] https://developers.openai.com/api/docs/assistants

diferentes formatos. Os assistentes podem receber arquivos durante a configuração ou no decorrer das interações, criar arquivos e referenciá-los nas respostas. Para isso, a API se organiza em torno de objetos principais: o Assistente, que representa a IA personalizada; a *Thread*, que corresponde à sessão de conversa entre o assistente e o usuário; a *Mensagem*, que contém o conteúdo trocado (texto, imagens, arquivos); a *Execução* (*Run*), que é a chamada do assistente para processar mensagens e utilizar modelos e ferramentas; e o *Passo de Execução*, que detalha cada ação realizada no processo de geração da resposta.

A escolha pela *Assistants* API da OpenAI foi motivada pelas necessidades específicas de implementação do AIPairCode, especialmente no que se refere à gestão do contexto das interações, à personalização do comportamento do assistente e à integração com uma arquitetura composta por múltiplos módulos. O suporte nativo a sessões persistentes (*threads*) permitiu manter o histórico das interações de forma estruturada, sem a necessidade de implementação manual de mecanismos de gerenciamento de contexto.

Outro fator relevante foi a integração nativa com mecanismos de recuperação de informações por arquivos e bases de conhecimento, aspecto importante para a implementação da estratégia de RAG utilizada no sistema. Isso possibilitou incorporar materiais didáticos da disciplina ao processo de geração das respostas do assistente, mantendo alinhamento com os conteúdos trabalhados em aula.

Dessa forma, a *Assistants* API mostrou-se adequada aos requisitos do sistema, permitindo a implementação de um assistente com comportamento controlado, integrado a diferentes interfaces e capaz de sustentar as estratégias de mediação pedagógicas propostas nesta pesquisa.

Embora existam outras APIs e *frameworks* para desenvolvimento de aplicações com LLMs, muitas dessas soluções exigem a implementação manual de mecanismos de gerenciamento de contexto, persistência das interações e integração com bases de conhecimento. Nesse sentido, a *Assistants* API apresentou-se, na época da implementação do sistema (2024), como uma solução integrada que reduziu a complexidade de desenvolvimento e possibilitou maior foco na definição das estratégias pedagógicas do assistente.

Ressalta-se que, após o desenvolvimento deste trabalho, a OpenAI anunciou a descontinuação gradual da *Assistants* API, com substituição por novos mecanismos de interação. Essa mudança, no entanto, não impacta os resultados da pesquisa, uma vez que a tecnologia utilizada atendeu plenamente aos requisitos do sistema no período de sua

implementação e utilização do AIPairCode. A adaptação e migração para as novas bibliotecas e APIs já está sendo desenvolvida como continuidade e trabalho futuro do projeto.

5.1.2 *Retrieval-Augmented Generation* (RAG)

O *Retrieval-Augmented Generation* (RAG) constitui uma abordagem que combina a capacidade generativa de modelos de linguagem de grande escala (LLMs) com a recuperação de informações provenientes de bases de dados externas, previamente validadas e confiáveis. Essa técnica surge como uma resposta às limitações dos modelos puramente generativos, os quais, em geral, dependem exclusivamente do conhecimento contido em seus parâmetros, podendo incorrer em lacunas ou imprecisões.

O funcionamento do RAG ocorre em duas etapas complementares. Primeiramente, um módulo de recuperação localiza, em uma base de conhecimento definida, informações pertinentes à consulta formulada pelo usuário. Em seguida, o modelo de linguagem utiliza tanto a consulta quanto os trechos recuperados como contexto para gerar a resposta. Essa integração possibilita que o conteúdo produzido pela IA esteja ancorado em fontes conhecidas, reduzindo a probabilidade de alucinações ou de informações desatualizadas (SHAHADE; DESHMUKH, 2024).

No contexto educacional, e especificamente no ensino de programação, a base de conhecimento utilizada pelo RAG pode ser constituída por recursos como a documentação oficial da linguagem de programação, materiais didáticos da disciplina, tutoriais, exemplos de código e conteúdos elaborados pelo docente. Esse processo de curadoria assegura que as respostas geradas pela IA sejam não apenas tecnicamente corretas, mas também contextualizadas em relação ao conteúdo trabalhado em sala de aula.

No AIPairCode, a abordagem de RAG foi implementada a partir da utilização de materiais didáticos fornecidos pelo professor por meio do módulo web do docente. Esses materiais, que podem incluir conteúdos teóricos das aulas, exemplos e orientações da disciplina, são armazenados no Firebase *Storage* e registrados no banco de dados (Firestore), compondo uma base estruturada de conteúdos da disciplina. Após o envio, os arquivos passam por um processo de indexação, sendo integrados à base de conhecimento utilizada pelo assistente, o que permite sua recuperação durante as interações com os estudantes.

Durante o uso do sistema, quando o estudante realiza uma solicitação, o *backend* chama o *Assistants* API que por sua vez aciona o mecanismo de recuperação, identificando trechos

relevantes desses materiais previamente indexados. Essas informações são então incorporadas ao contexto da requisição, permitindo que a resposta gerada esteja alinhada aos conteúdos trabalhados na disciplina.

Além da utilização do RAG para fundamentar as respostas do assistente, o sistema também emprega essa base de conhecimento para apoiar o estudante de forma proativa. Ao iniciar um novo exercício, o AIPairCode utiliza o enunciado fornecido pelo aluno para identificar, por meio da IA, os materiais mais relevantes dentro da base disponível, recomendando conteúdos de apoio que possam auxiliar na resolução do problema. Essa funcionalidade contribui para orientar o estudante no uso de recursos adequados, promovendo maior autonomia e alinhamento com o percurso pedagógico da disciplina.

5.1.3 Engenharia de *Prompt*

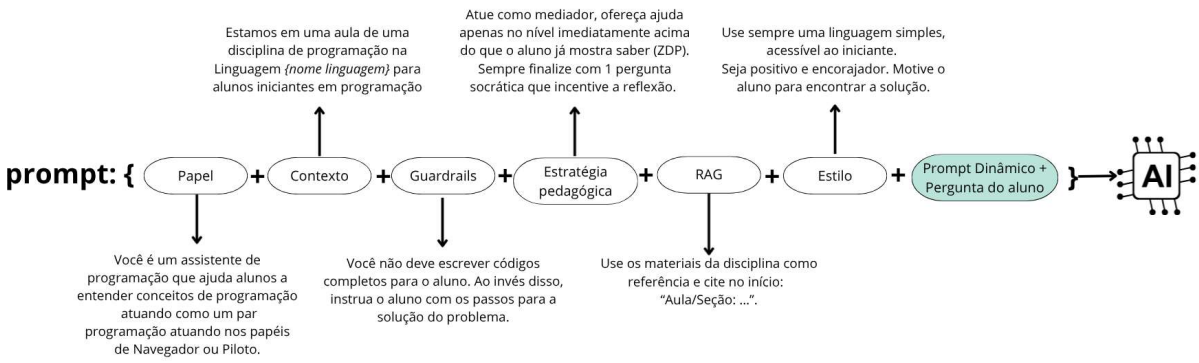
Um *prompt* é a forma pela qual as instruções são fornecidas a modelos de IA generativa, sendo fundamental para orientar o tipo de resposta ou ação produzida. A engenharia de *prompt* consiste em um conjunto de técnicas utilizadas para projetar e estruturar essas instruções, garantindo que as respostas do modelo sejam adequadas a contextos específicos. No contexto educacional, essa abordagem permite alinhar o comportamento da IA aos objetivos de aprendizagem, favorecendo respostas mais direcionadas, coerentes e pedagogicamente adequadas (MOLLICK; MOLLICK, 2023; SCHULHOFF *et al.*, 2024).

Além disso, a engenharia de *prompt* permite a personalização do nível de complexidade das respostas (SCHULHOFF *et al*., 2024). Por meio de instruções adaptativas, o assistente pode ajustar a profundidade das explicações de acordo com o nível de compreensão do estudante, permitindo uma abordagem mais gradual e acessível. Esse aspecto é particularmente relevante no ensino de programação para iniciantes, que podem se sentir sobrecarregados diante de soluções completas ou explicações excessivamente complexas.

No AIPairCode, a engenharia de *prompt* foi utilizada como o principal mecanismo para definir o comportamento do assistente enquanto par de programação, estabelecendo diretrizes para a forma, o conteúdo e a intencionalidade das respostas geradas. Ela foi implementada em duas camadas complementares: um *Prompt* base, configurado estaticamente na *Assistants* API, e um *Prompt* dinâmico, gerado e injetado pelo *backend* a cada interação. O objetivo não foi apenas obter respostas corretas, mas estruturar interações que estimulassem o raciocínio do estudante, promovendo a construção progressiva do conhecimento.

O *prompt* base do sistema estabelece as regras fundamentais de conduta da IA, configurada para atuar como um par de programação voltado a estudantes iniciantes. A Figura 10 ilustra a estrutura conceitual que orienta a formulação dessas instruções, garantindo que o modelo assuma a identidade pedagógica adequada, respeite restrições e mantenha um estilo encorajador.

Figura 10 - Estrutura de construção do *prompt* base



Fonte: Elaborado pela autora (2025)

As principais diretrizes estabelecidas neste escopo global englobam:

- Restrição de código completo: a IA é instruída a não fornecer respostas que contenham soluções completas, devendo orientar o próximo micropasso coerente. Essa diretriz está diretamente relacionada ao componente de *guardrails* apresentado na Figura 10, no qual o assistente é orientado a não resolver integralmente o problema, mas a apoiar o estudante na construção progressiva da solução.
- Alternância de papéis: o assistente inicia as interações como Navegador, operando com raciocínio lógico e pseudocódigo, e a mudança para o papel de Piloto ocorre sob condições específicas. Essa dinâmica está associada ao componente *papel* do prompt, permitindo simular a prática de programação em pares e ajustar o nível de intervenção conforme o progresso do aluno.
- Mediação socrática: a IA é instruída a evitar perguntas fechadas de "sim" ou "não", priorizando perguntas de checagem, previsão, comparação, explicação e metacognição. Essa diretriz está vinculada ao componente de estratégia pedagógica da Figura 10, no qual o assistente atua como mediador, incentivando a reflexão e respeitando a ZDP do estudante.
- Estrutura fixa de resposta: para garantir consistência e clareza nas interações, foi definido um formato estruturado de resposta. Cada resposta do assistente é organizada

em quatro elementos principais: (i) uma análise breve da situação atual; (ii) a indicação de etapas ou um pequeno trecho de código acompanhado de explicação; (iii) a definição de uma próxima ação concreta a ser realizada pelo estudante; e (iv) uma pergunta final de caráter socrático. Essa estrutura favorece a continuidade do diálogo, estimula o raciocínio ativo e evita respostas excessivamente extensas ou desorganizadas.

Além dessas diretrizes, conforme ilustrado na Figura 10, o *prompt* incorpora elementos adicionais que contribuem para o comportamento pedagógico do assistente. O contexto define o cenário educacional e o perfil do estudante, enquanto o uso de RAG permite que o assistente utilize materiais da disciplina como base para suas respostas. O componente de estilo orienta o uso de linguagem simples, acessível e encorajadora, reforçando o papel do assistente como mediador do processo de aprendizagem. Por fim, o *prompt* dinâmico, garante a adaptação contínua das respostas ao contexto da atividade.

Um dos principais elementos estruturantes dessas instruções é a alternância de papéis inspirada na programação em pares, distinguindo o Navegador e o Piloto. No papel de Navegador, o assistente atua na análise do problema, auxiliando na interpretação do enunciado, na identificação de erros e na elaboração de estratégias de solução, sem fornecer diretamente código. Já no papel de Piloto, o assistente pode fornecer pequenos trechos de código, limitados a blocos curtos e focados em um único conceito, sempre acompanhados de explicações e orientações para continuidade. A transição entre esses papéis ocorre de forma controlada pelo *backend*, considerando o comportamento do estudante ao longo da interação.

Além disso, o sistema integra a abordagem de RAG à engenharia de *prompt*. Os materiais didáticos da disciplina, previamente selecionados e indexados, são utilizados como base de conhecimento para enriquecer o contexto da geração de respostas. Dessa forma, quando o assistente introduz novos conceitos ou exemplos, ele pode fundamentar suas explicações em conteúdos alinhados às aulas, garantindo maior consistência pedagógica e reduzindo a ocorrência de informações imprecisas.

No entanto, a principal diferenciação na arquitetura do *prompt* do AIPairCode ocorre no *backend*, onde é criada uma camada de *prompt* dinâmico. A cada interação, o sistema constrói um contexto atualizado que incorpora múltiplos elementos: o enunciado do exercício, o estado atual do código desenvolvido pelo aluno e os dados do progresso do aluno. Essa composição permite que a IA evite repetições, mantenha coerência com a solução em desenvolvimento e direcione a resposta para o próximo passo relevante.

A adaptação do *prompt* é realizada por meio do mecanismo de *Run Instructions* da *Assistants* API, que permite sobrescrever dinamicamente as instruções do modelo a cada interação. Esse mecanismo viabiliza a implementação de estratégias de *scaffolding* (andaimes) baseadas no desempenho real e imediato do aluno. A Figura 11 apresenta um trecho representativo do código-fonte do *backend*, evidenciando como essas diretrizes pedagógicas são inseridas dinamicamente no *prompt* a cada interação.

Figura 11 - Exemplo de inserção dinâmica de contexto pedagógico no *backend*

```
  const RUN_INSTRUCTIONS = `
# Override deste run (reforço)
${papelHint}  // Injeta a diretriz: NAVEGADOR ou PILOTO
${modeHint}   // Injeta a restrição de concisão (tamanho da resposta)
${nivelHint}  // Injeta regras dos níveis de suporte (L1, L2, L3 ou L4)
${categoriaHint}  // Injeta regras baseadas nas categorias de solicitação (CORRECAO, ERRO, SINTAXE, EXPLICACAO, ESTRATEGIA, REFATORACAO)
${tentativaHint}  // Injeta regras baseadas no nº de tentativas sem sucesso do aluno
- FOCO: mantenha toda a resposta no tema "${focusTerm}" (objetivo: "${objectiveKey}").
- Evite repetir o enunciado ou o que já está implementado.
${CONTINUATION_HINT}

${formatoHint}
- Teste rápido: gere **exatamente 1 pergunta socrática** seguindo este BRIEFING (não copie o texto; escreva a pergunta final em 1 linha):
  "${briefingSocratico}"
- Em PILOTO: se houver código, **após o bloco** escreva **uma única linha** começando com "Por quê:" explicando em até 15 palavras
o que o trecho faz e o próximo passo.
`.trim();

const papelHint =
  papelDecidido === 'navegador'
    ? '- PAPEL: NAVEGADOR (NÃO mostrar código real; converta trechos em pseudocódigo claro).'
    : '- PAPEL: PILOTO (máx. 1 bloco e até 5 linhas de código).';

const modeHint = direct ? '- MODO: direto, objetivo (2–5 frases).' : '- MODO: conciso (2–5 frases).';

const tentativaHint =
  attempts <= 2
    ? '- TENTATIVAS 1–2: use perguntas socráticas + pistas, sem exemplo de código.'
    : attempts === 3
      ? '- TENTATIVA 3: ofereça um mini-exemplo (≤5 linhas) APENAS do conceito faltante (no PILOTO); no NAVEGADOR, descreva em pseudocódigo.'
      : '- TENTATIVAS 4+: forneça micro-trecho com LACUNAS (____) para o aluno completar; mantenha-se no escopo do conceito atual.';

const nivelHint =
  sessao.nivel === 'L1'
    ? '- NÍVEL L1 (altíssimo suporte): mais guiado; checklists curtos; se der exemplo, mínimo possível.'
    : sessao.nivel === 'L2'
      ? '- NÍVEL L2 (suporte moderado): 1 pergunta socrática + exemplo curto só após erro repetido.'
      : sessao.nivel === 'L3'
        ? '- NÍVEL L3 (suporte leve): foco em "por quê" + 1 pista; só exemplo se o aluno pedir.'
        : '- NÍVEL L4 (autonomia): checklist de validação e referências, sem exemplos.';
```

Fonte: Elaborado pela autora (2025).

Como demonstrado na Figura 11, o sistema insere no *prompt* variáveis, incluindo:

- O papel atual do assistente (Navegador ou Piloto);
- O nível de suporte pedagógico (ZDP);
- O número de tentativas realizadas pelo estudante;
- O tipo de solicitação identificada (como correção, explicação ou estratégia);
- O foco conceitual da interação, definido a partir do objetivo da tarefa.

A classificação automática da solicitação do estudante permite ajustar a forma da resposta, diferenciando, por exemplo, intervenções voltadas à depuração de código, explicações conceituais ou orientação estratégica. Esse mecanismo contribui para tornar a

assistência mais contextualizada e adequada à necessidade específica do aluno.

Outro aspecto importante é a consideração do progresso do estudante ao longo da interação. O sistema analisa o estado atual do código e identifica padrões de evolução ou estagnação, permitindo adaptar o nível de suporte oferecido. Essa abordagem possibilita uma mediação mais precisa, evitando tanto o excesso de ajuda quanto a ausência de suporte em momentos críticos.

A arquitetura também incorpora um mecanismo de geração de perguntas socráticas contextualizadas. Em vez de utilizar perguntas genéricas, o sistema analisa o enunciado, a dúvida do aluno e o código desenvolvido para identificar elementos como estrutura de dados, operação envolvida, casos mínimos e possíveis erros. Com base nessas informações, o assistente gera perguntas alinhadas ao método socrático, explorando aspectos como raciocínio, evidências, suposições e implicações das soluções propostas. Para operacionalizar essas estratégias pedagógicas, o sistema utiliza movimentos socráticos derivados dessas categorias teóricas, como previsão, explicação, comparação, verificação e decomposição, promovendo a reflexão ativa e a construção gradual do conhecimento pelo estudante.

Além disso, a adaptação do suporte é realizada com base em dois eixos principais:

- Escalonamento por nível de proficiência (ZDP): o sistema ajusta o grau de detalhamento das respostas, variando de orientações guiadas até níveis de maior autonomia. Como por exemplo: o nível L1 exige um suporte alto (guiado com uso de passos ou etapas); o nível L2 oferece suporte moderado; o nível L3 foca em questionar as ações, reduzindo os exemplos; e o nível L4 foca na autonomia, exigindo que o aluno valide seu próprio código apresentando pequenos exemplos de código.
- Escalonamento por tentativas: o *backend* monitora quantas vezes o aluno tentou resolver a mesma etapa. Nas tentativas 1 e 2, a IA usa apenas perguntas socráticas e passos ou etapas. Na tentativa 3, é liberado um exemplo. A partir da tentativa 4, o *prompt* instrui a IA a gerar trechos de código com lacunas, forçando o aluno a preencher a lógica correta em vez de simplesmente copiar uma resposta pronta.

O desenvolvimento da engenharia de *prompt* ocorreu de forma iterativa, a partir da análise das interações dos estudantes com o sistema. Versões iniciais, baseadas apenas em instruções gerais, resultaram em respostas excessivamente longas ou diretas. A partir desse diagnóstico, foram incorporados mecanismos de controle, como a definição explícita de papéis, a limitação do tamanho dos trechos de código, a adaptação baseada em tentativas e o

uso de instruções dinâmicas em tempo de execução.

Por fim, o *prompt* completo gerado é então enviado ao modelo, que retorna uma resposta estruturada conforme os princípios pedagógicos estabelecidos, garantindo que o aluno receba uma orientação mais direcionada e que possa promover o aprendizado ativo.

5.1.4 Inserção de *Guardrails* no AIPairCode

A implementação de um assistente de programação baseado em IA exige mecanismos de controle que garantam que as respostas geradas estejam alinhadas aos objetivos pedagógicos. Nesse contexto, a incorporação de *guardrails* torna-se essencial para assegurar que a experiência de aprendizado seja eficaz e promova o desenvolvimento de habilidades de programação. Esses mecanismos têm como objetivo restringir a forma como as respostas são geradas, assegurando que os alunos não recebam soluções completas, mas sim orientações e dicas que incentivem a sua própria aprendizagem (AYYAMPERUMAL; Ge, 2024).

Um dos principais objetivos dos *guardrails* é evitar que os alunos se tornem excessivamente dependentes do assistente para a resolução de problemas. Quando o modelo fornece códigos completos, há um risco de que os estudantes não se envolvam ativamente no processo de resolução de problemas, o que pode prejudicar seu aprendizado, conforme relatado como principais dificuldades nos estudos de Hu *et al.*, 2023, Sun *et al.*, 2024, Karnalim *et al.*, 2023; Yilmaz e Yilmaz, 2023, Silva *et al.*, 2024a, Haindl e Weinberger, 2024 e Prather *et al.*, 2024. Assim, os *guardrails* devem ser configurados para estimular a reflexão crítica, incentivando os alunos a pensar sobre a lógica por trás da programação e a desenvolver suas habilidades de codificação de maneira autônoma.

O AIPairCode foi desenvolvido para fornecer trechos de código ou pseudocódigo, em vez de soluções completas, desafiando os alunos a completarem as partes restantes. Os *guardrails* foram implementados como um sistema multicamadas, atuando em diferentes níveis da arquitetura para garantir o controle efetivo das respostas geradas.

Na camada de *prompt*, as restrições são definidas explicitamente por meio de instruções que orientam o comportamento do modelo. Essas diretrizes incluem a proibição de fornecimento de código completo e a obrigatoriedade de respostas estruturadas em micropassos. Tais elementos são inseridos tanto no *prompt* base quanto nas instruções dinâmicas, garantindo que o modelo receba, a cada interação, orientações claras sobre os limites e objetivos da resposta.

Na camada de *backend,* o sistema implementa mecanismos adicionais responsáveis por validar e ajustar as respostas antes de serem exibidas ao estudante. Entre as principais estratégias adotadas, destacam-se:

- Limitação do tamanho de blocos de código: no papel de Piloto, os trechos de código são restritos a pequenos fragmentos (até 5 linhas), focados em um único conceito;
- Supressão de código no papel de Navegador: caso a IA gere código indevidamente, o *backend* remove automaticamente esses trechos;
- Quebra de respostas excessivas: blocos de código ou explicações muito longas são reduzidos, indicando omissões intencionais;
- Controle de formato: o sistema garante que a resposta siga a estrutura definida (análise, orientação, próxima ação e pergunta socrática).

A figura 12 apresenta um trecho de código para exemplificar o uso de *guardrails* no *backend*.

Figura 12 - Exemplo de implementação de *guardrails* no *backend*



```
//----------------- Guardrails: NAVEGADOR --------------------------
export function bloquearCodigoNavegador(resposta: string): string {
  let out = String(resposta || '');
  out = out.replace(/```([\s\S]*?)```/g, (_m, inner) => toPseudoFromCodeBlock(inner));
  out = out.replace(/`([^`\n]{1,120})`/g, '[$1]');

  const linhas = out.split(/\r?\n/);
  const result: string[] = [];
  let buffer: string[] = [];
  const flush = () => {
    if (buffer.length) {
      const pseudo = toPseudoFromCodeLines(buffer);
      result.push(pseudo || '- descreva os passos em bullets');
      buffer = [];
    }
  };
//----------------- Guardrails: PILOTO -------------------------
export function limitarQtdBlocos(resposta: string, maxBlocks = 1): string {
  let count = 0;
  return String(resposta || '').replace(/```[\s\S]*?```/g, (blk) => {
    count++;
    return count <= maxBlocks ? blk : ````// bloco omitido para evitar solução completa````;
  });
}

export function restringirTamCodigo(resposta: string, maxLines = 5): string {
  return String(resposta || '').replace(/```([\s\S]*?)```/g, (m, inner) => {
    const lines = String(inner).split(/\r?\n/);
    const first = (lines[0] || '').trim();
    const hasLangHeader = /^[a-zA-Z0-9_+\-]+$/.test(first);
    const langHeader = hasLangHeader ? first + '\n' : '';
    const codeLines = hasLangHeader ? lines.slice(1) : lines;
    if (codeLines.length <= maxLines) return m;
    const capped = codeLines.slice(0, maxLines).join('\n') + '\n// ... restante por sua conta';
    return '````' + langHeader + capped + '````';
  });
}
```

Fonte: Elaborado pela autora (2025)

Esses controles garantem que, mesmo quando a IA não segue completamente as instruções do *prompt*, a resposta final apresentada ao estudante permaneça dentro dos limites definidos.

Além disso, o sistema também implementa verificações adicionais para identificar padrões indesejados nas respostas geradas. Mecanismos monitoram a quantidade de interrogações por mensagem evitando respostas excessivamente longas ou dispersas, verificam a alternância de papéis com base na sequência de interações, detecção de respostas repetitivas por meio da comparação com interações recentes e validação de conteúdo vazio ou inconsistente, evitando a exibição de respostas incompletas.

Considerando que LLMs podem ocasionalmente desrespeitar instruções, o AIPairCode incorpora um mecanismo de *fallback* automático. Quando a resposta gerada apresenta problemas, como estrutura incorreta, excesso de código, repetição ou inconsistência, o sistema intercepta a saída e realiza uma nova requisição ao modelo com instruções mais restritivas. Esse processo ocorre de forma transparente para o usuário, garantindo que apenas respostas válidas e adequadas sejam exibidas. O *fallback* atua, portanto, como uma camada adicional de segurança, reforçando os *guardrails* previamente definidos.

Na camada de *interface*, os *guardrails* são reforçados por meio de elementos visuais e interacionais. A identificação do papel atual do assistente (Navegador ou Piloto), por exemplo, é apresentada de forma explícita ao estudante, contribuindo para a compreensão do tipo de ajuda fornecida em cada momento. Essa transparência favorece a construção de um modelo mental adequado sobre o funcionamento do sistema, evitando interpretações equivocadas e promovendo uma interação mais consciente.

Os *guardrails* foram refinados de forma iterativa, a partir da análise das interações dos estudantes com o sistema. O registro das perguntas, respostas e padrões de uso permite identificar situações em que o assistente fornece suporte excessivo ou insuficiente, possibilitando ajustes contínuos nas regras de controle.

Adicionalmente, foi estabelecido um sistema de *feedback,* no qual os estudantes podem sinalizar respostas erradas, contribuindo para o aprimoramento contínuo da ferramenta, que foi complementado por orientações aos alunos sobre a importância de revisar as respostas geradas pela IA. Esse processo favorece a confiabilidade do sistema e fortalece seu papel como mediador do aprendizado.

Por fim, as limitações técnicas da IA também foram consideradas durante o desenvolvimento do assistente, incluindo aspectos relacionados ao custo computacional, tempo de resposta e escalabilidade do sistema. Atualmente, o AIPairCode utiliza a API da OpenAI com o modelo GPT-4.1 mini, escolhido por apresentar um equilíbrio adequado entre

desempenho, velocidade e custo de utilização. O modelo possui suporte a contexto de até 1 milhão de *tokens* e foi otimizado para tarefas de programação e seguimento de instruções, características relevantes para o contexto educacional da aplicação.

A escolha do modelo também considerou a necessidade de sustentar múltiplas interações contínuas entre estudantes e assistente durante as atividades práticas, e o GPT-4.1 mini apresentou-se como uma alternativa viável com custo aproximado de US$ 0,40 por milhão de tokens de entrada e US$ 1,60 por milhão de tokens de saída, valores inferiores aos de modelos maiores da mesma família.

Cabe destacar que a comparação entre diferentes modelos não constituiu objetivo desta pesquisa e a escolha esteve relacionada principalmente à viabilidade técnica e operacional do sistema, buscando garantir estabilidade, custo compatível com o volume de interações e desempenho adequado às estratégias implementadas no AIPairCode. Além disso, a arquitetura do sistema foi desenvolvida de forma modular, permitindo a substituição ou atualização do modelo utilizado conforme a evolução das APIs e o surgimento de modelos mais avançados ou mais eficientes em termos computacionais e financeiros.

5.2 Módulos do AIPairCode

A partir da arquitetura geral apresentada na Seção 5.1, esta seção detalha os principais módulos de interface que compõem o sistema AIPairCode, destacando as formas de interação com o usuário, as funcionalidades disponíveis e os casos de uso associados a cada componente. O sistema foi estruturado em três módulos que se complementam e se integram: a extensão para Visual Studio Code, o Módulo Web do Estudante e o Módulo do Professor .

5.2.1 Extensão para o Visual Studio Code

A extensão do AIPairCode para o Visual Studio Code (VS Code) é uma das principais interfaces de interação entre o estudante e o assistente de programação. Ela possibilita que o aluno utilize a ferramenta diretamente no editor de código, acompanhando o processo de programação em tempo real e disponibilizando um conjunto de funcionalidades que garantem um fluxo contínuo de aprendizagem. Essa integração foi desenvolvida com o objetivo de evitar a necessidade de alternar entre diferentes ferramentas favorecendo a continuidade do raciocínio durante a resolução de problemas. Essa interface foi utilizada pelos alunos durante as aulas de laboratório nos estudos finais 1 e 2.

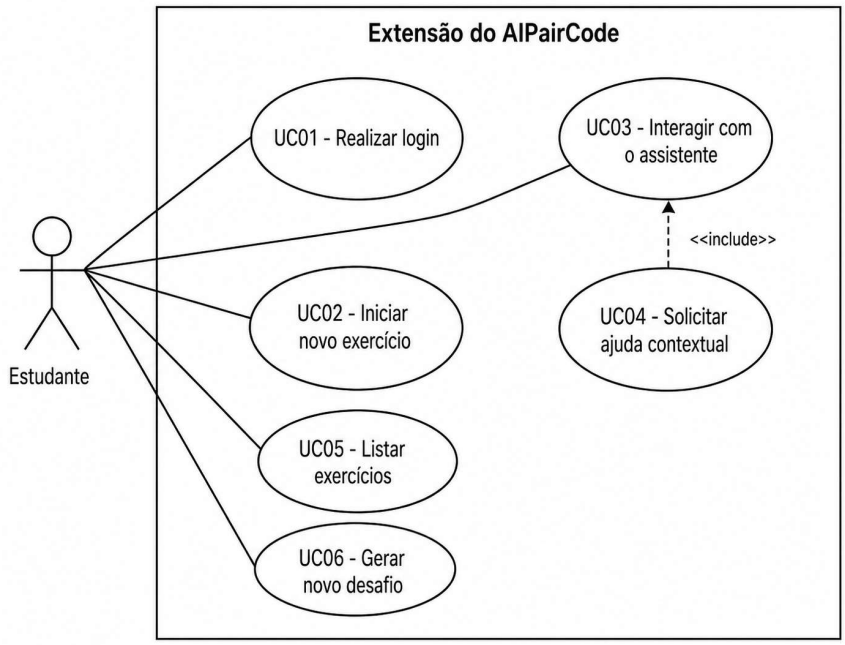
Inicialmente, foram identificadas as principais funcionalidades de interação do estudante com o sistema. O estudante deve realizar autenticação por meio de e-mail e senha, garantindo segurança no acesso e rastreabilidade das interações. A partir disso, pode iniciar novos exercícios diretamente no editor, interagir com o assistente por meio de mensagens ou envio de código, solicitar diferentes tipos de ajuda contextual, acessar exercícios anteriores e gerar novos desafios a partir das atividades em andamento.

Os principais casos de uso associados à extensão do AIPairCode são:

- UC01 - Realizar login: o estudante autentica-se na extensão com seu e-mail e senha para iniciar o uso do sistema.
- UC02 - Iniciar novo exercício: o estudante pode criar um novo exercício de programação inserindo o enunciado do problema direto no editor de código ou através de uma janela modal.
- UC03 - Interagir com o assistente: o estudante envia perguntas ou trechos de código e recebe respostas da IA.
- UC04 - Solicitar ajuda contextual: o estudante pode utilizar os botões de apoio (“Como começar?”, “Explica esse trecho”, “Corrige esse trecho”) para se comunicar com a IA.
- UC05 - Listar exercícios: o estudante pode acessar os exercícios desenvolvidos anteriormente e retomar as interações com os mesmos.
- UC06 - Gerar novo desafio: o estudante solicita para a IA a criação de novos problemas relacionados ao exercício atual e a IA retorna com enunciados de novos exercícios para serem desenvolvidos.

A Figura 13 apresenta o diagrama de casos de uso da extensão do AIPairCode, sintetizando as principais interações do estudante com o sistema no contexto do Visual Studio Code.

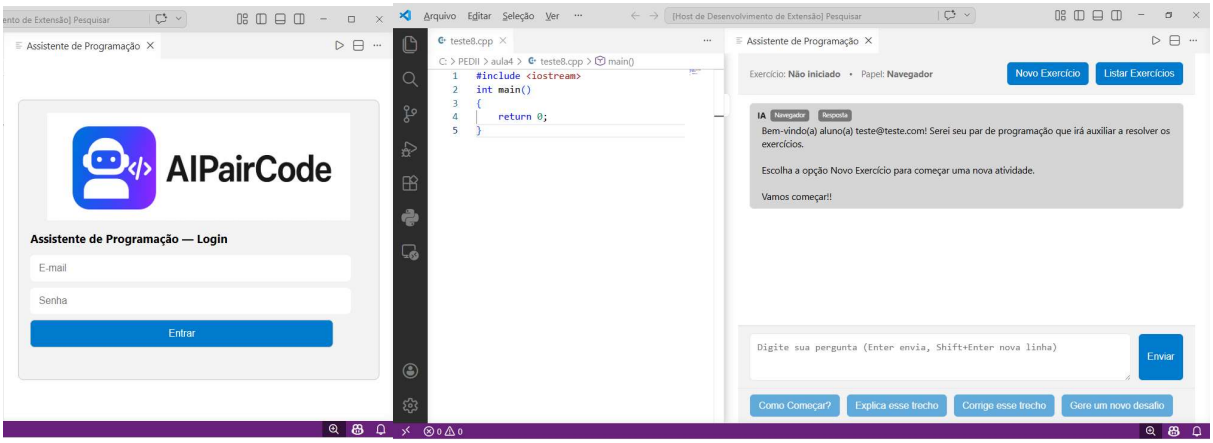
Figura 13 - Diagrama de casos de uso da extensão do AIPairCode



Fonte: Elaborado pela autora (2025)

Do ponto de vista da experiência do usuário, foram desenvolvidas telas específicas para cada etapa da interação. A tela de *login*, implementada na *webview* da extensão, apresenta um *design* simples, com a identidade visual do assistente e campos para inserção de e-mail e senha. Uma vez autenticado, o estudante tem acesso à interface principal da extensão, organizada em uma janela lateral do VS Code, onde ocorrem as interações com o assistente por meio de um *chat*. A Figura 14 apresenta as telas de login e a interface inicial da extensão.

Figura 14 - Telas de login e interface inicial da extensão

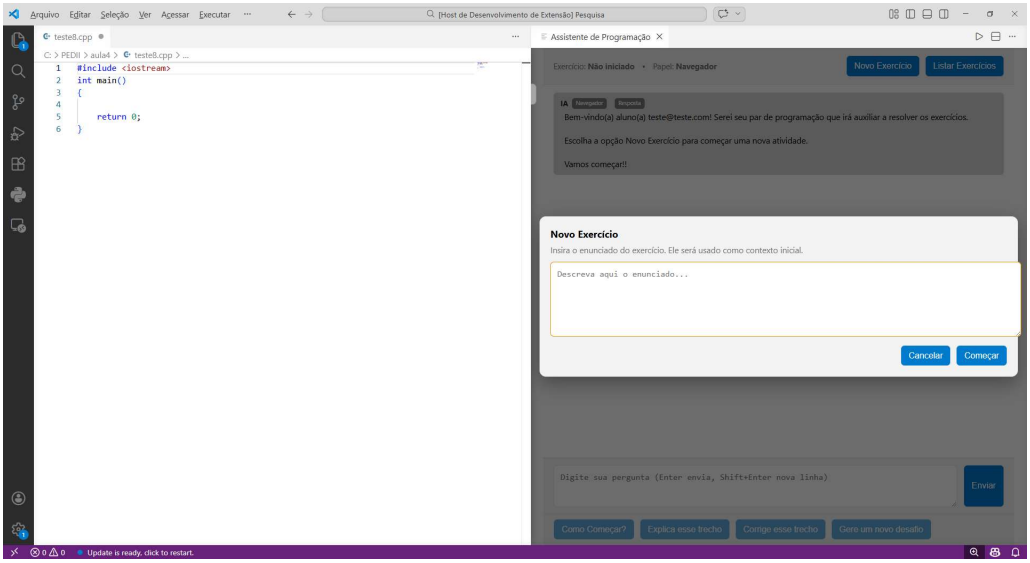


Fonte: Elaborado pela autora (2025)

A partir dessa interface, o estudante pode iniciar novos exercícios, registrando o enunciado diretamente no código (por meio de comentários no início do arquivo) ou através

de uma janela *modal*, além de retomar atividades anteriores armazenadas no sistema. A criação de um novo exercício pode ser realizada por meio do botão correspondente, conforme ilustrado na Figura 15.

Figura 15 - Iniciar novo exercício na extensão



Fonte: Elaborado pela autora (2025)

Após o início de um novo exercício o aluno pode iniciar a interação com o assistente por meio do *chat*, enviando suas perguntas e recebendo as respostas da IA. A interface do *chat* diferencia visualmente as perguntas do estudante e as respostas da IA, contribuindo para a clareza do diálogo e reforçando a percepção de acompanhamento personalizado.

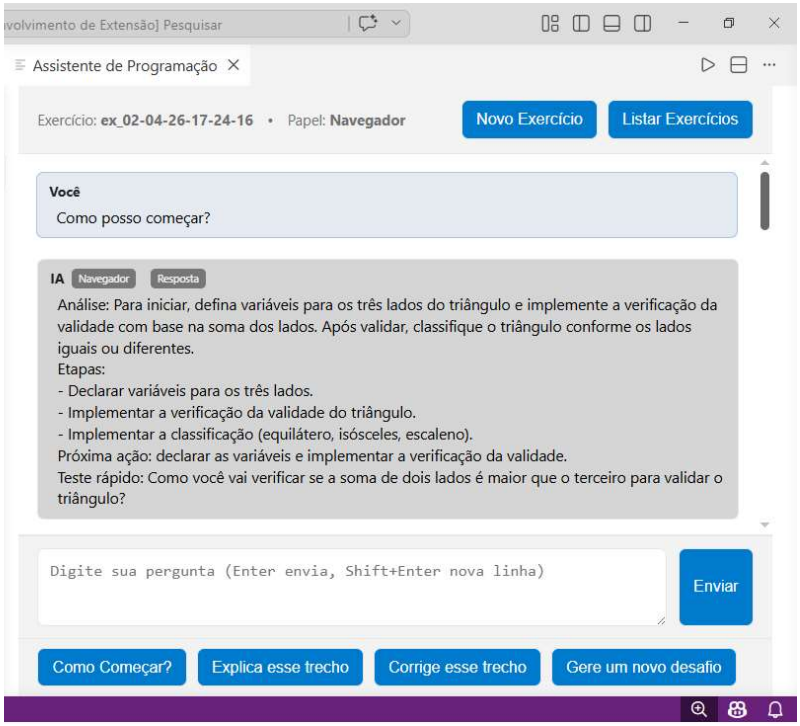
Durante a interação, o assistente atua de forma dinâmica, alternando automaticamente entre os papéis de navegador e piloto, conforme o contexto da atividade. No papel de navegador, a IA prioriza explicações, questionamentos e orientações conceituais, enquanto no papel de piloto fornece pequenas intervenções no código, sempre de forma controlada e sem apresentar soluções completas. Essa alternância é gerenciada pelo *backend* e orientada pelas estratégias definidas na engenharia de *prompt* e pelos mecanismos de restrição implementados, garantindo que o suporte oferecido esteja alinhado aos objetivos pedagógicos do sistema.

Durante cada interação, o sistema envia automaticamente ao assistente não apenas a pergunta do estudante, mas também o estado atual do código em desenvolvimento. Esse processo permite que a IA acompanhe continuamente a evolução da solução, oferecendo respostas mais contextualizadas e alinhadas ao que o estudante já produziu.

Localizados na área inferior da interface, abaixo do *chat*, estão disponíveis os botões de ações rápidas, que têm a função de orientar e apoiar a comunicação entre o aluno e a IA, funcionando como atalhos pedagógicos. Cada um deles foi desenhado para estimular diferentes tipos de interação, assim o aluno não depende apenas de elaborar suas próprias perguntas, mas conta com sugestões que o ajudam a refletir sobre o código em análise e a construir gradualmente sua compreensão.

O botão *"Como começar?"* atua como um disparador inicial, sendo especialmente útil quando o aluno não sabe como iniciar a resolução do exercício. Neste caso, o assistente apresenta orientações sugerindo etapas iniciais ou levantando questões que auxiliam na construção do raciocínio. Um exemplo dessa interação é apresentado na Figura 16.

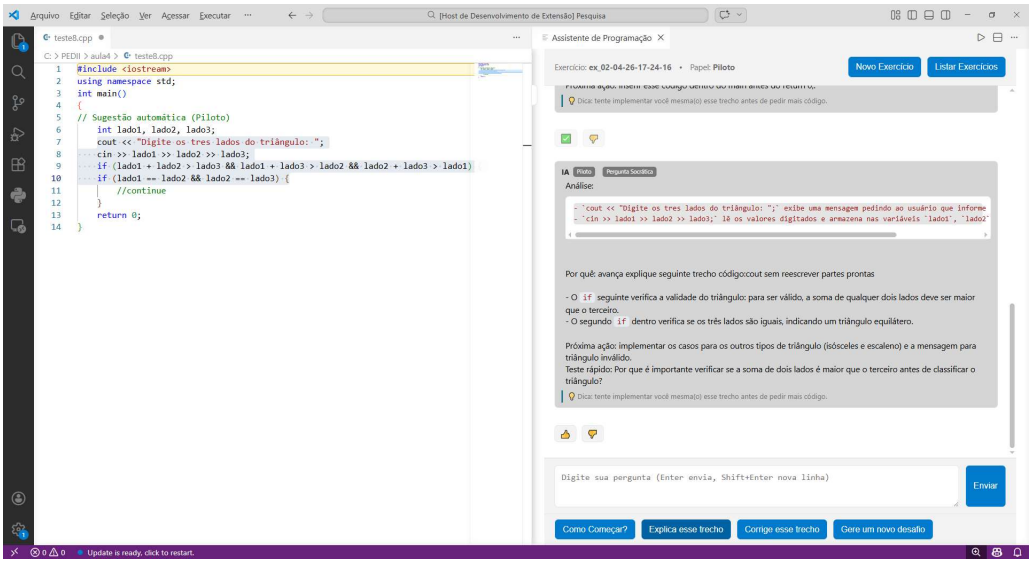
Figura 16 - Uso do botão *"Como começar?"*



Fonte: Elaborado pela autora (2025)

O botão *"Explica esse trecho"* permite ao aluno solicitar esclarecimentos sobre partes específicas do código. O aluno deve selecionar um trecho de código e clicar nesse botão, o assistente explicará todo o funcionamento deste trecho de código de forma simples e didática. Essa função está diretamente ligada à ideia de mediação, em que o assistente age como um par mais experiente, traduzindo a complexidade técnica para um nível acessível e apoiando a internalização de novos conceitos (VYGOTSKY, 1978). Figura 17, exemplo do botão *Explica esse trecho"*.

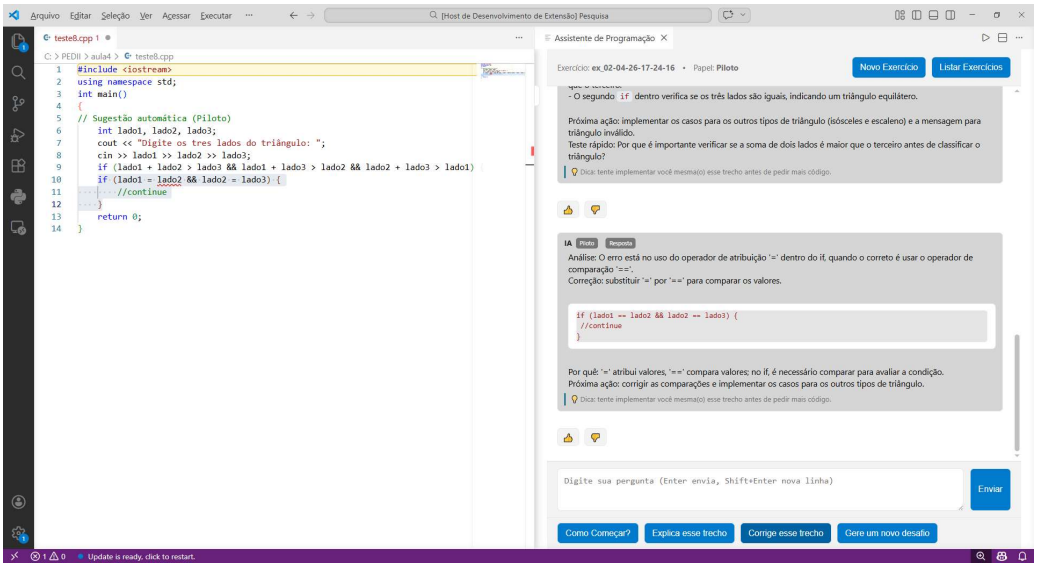
Figura 17 - Exemplo do botão *"Explica esse trecho"*



Fonte: Elaborado pela autora (2025)

Já o botão *"Corrige esse trecho"* é pensado para situações em que o estudante suspeita que seu código está incorreto ou incompleto. Da mesma forma que o botão anterior, o aluno seleciona um trecho de código com erro, e a IA explica a ele como deve fazer a correção. Nesse caso, a IA não substitui a autoria do aluno, mas fornece *feedback* localizado, sugerindo alterações ou destacando possíveis erros. Tal recurso fortalece o papel da avaliação formativa e dialogada, essencial em contextos de aprendizagem ativa. A Figura 18 apresenta um exemplo de uso desse botão.

Figura 18 - Exemplos de uso do botão *"Corrige esse trecho"*



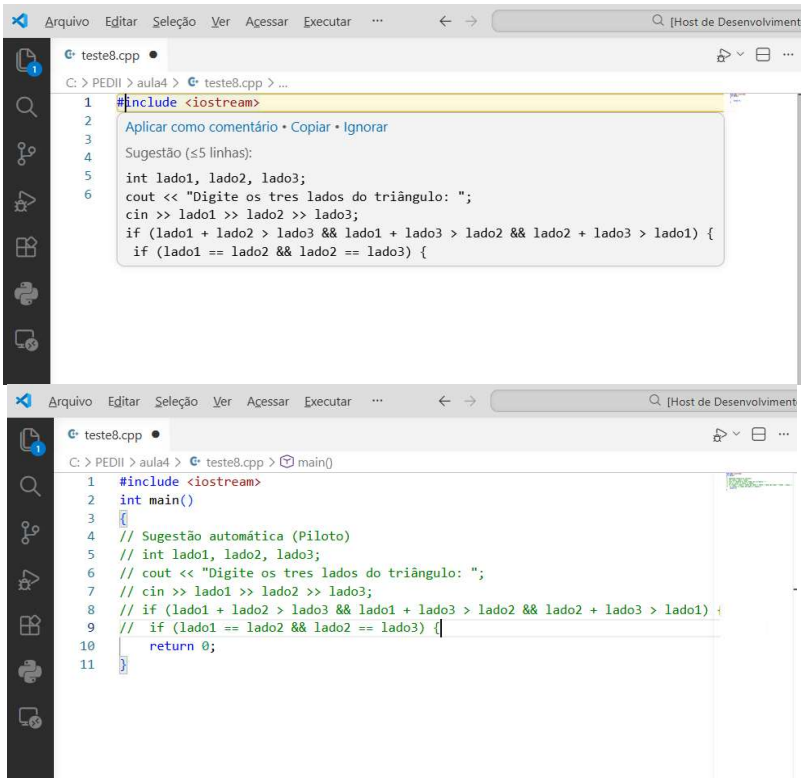
Fonte: Elaborado pela autora (2025)

Por fim, o botão *"Gere um novo desafio"* tem como objetivo ampliar gradualmente a dificuldade da atividade, oferecendo uma tarefa adicional ou uma variação do exercício em andamento. Essa funcionalidade contribui para manter o engajamento e a motivação dos alunos, ao mesmo tempo em que permite ao assistente ajustar a complexidade dos problemas ao nível de desenvolvimento do estudante.

Desse modo, os botões não atuam apenas como recursos de interface, mas como componente de mediação pedagógica, estruturando a interação com o assistente e promovendo diferentes tipos de apoio ao longo do processo de aprendizagem.

No modo piloto, além da resposta apresentada no *chat*, o assistente pode também sugerir intervenções diretamente no código do estudante. Essas sugestões são inseridas no editor como comentários, destacando trechos relevantes ou indicando possíveis modificações, e o aluno pode aceitá-las ou recusá-las. Esse mecanismo amplia a integração entre o assistente e o ambiente de programação, permitindo que o suporte ocorra de forma mais prática e situada no contexto do código. A Figura 19 apresenta essa funcionalidade.

Figura 19 - Sugestões de correções *inline* no editor



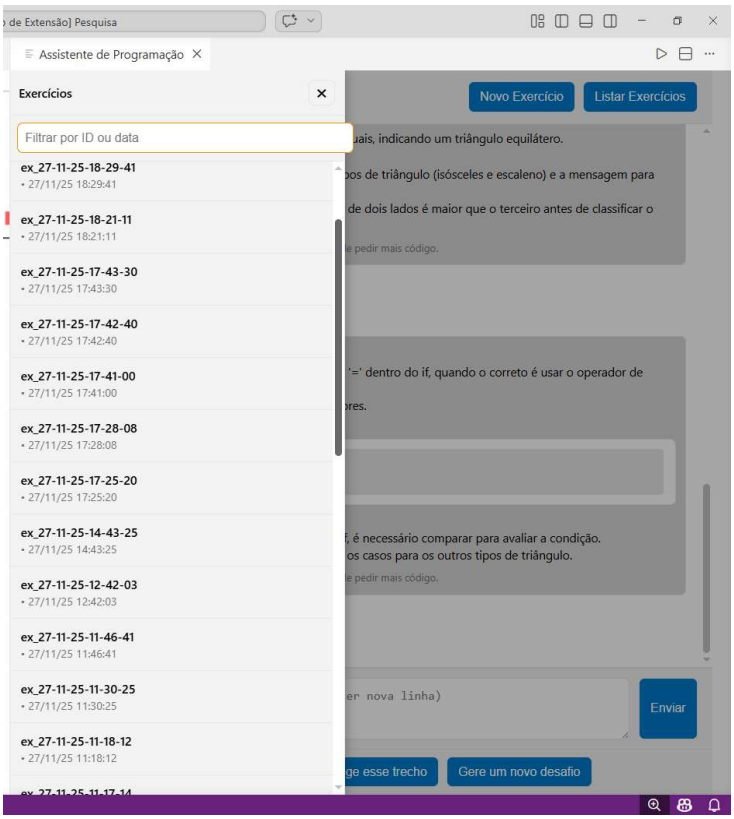
Fonte: Elaborado pela autora (2025)

Adicionalmente, o sistema permite que o estudante avalie cada resposta do assistente por meio de um mecanismo de *feedback*, representado por ícones de avaliação positiva e

negativa. Essa funcionalidade possibilita ao aluno expressar sua percepção sobre a utilidade da resposta recebida, contribuindo tanto para o aprimoramento do sistema quanto para a análise das interações no contexto da pesquisa. Fornecer *feedback* é opcional, e, se o aluno enviar uma avaliação, uma mensagem de confirmação (*"Obrigado pelo feedback!"*) será exibida.

Além das interações em tempo real, a extensão também disponibiliza uma funcionalidade de listagem de exercícios, que exibe todos os trabalhos salvos e possibilita a retomada do histórico de interações com um exercício selecionado. A tela de listagem de exercícios é apresentada na Figura 20.

Figura 20 - Tela de listagem de exercícios



Fonte: Elaborado pela autora (2025)

No contexto da pesquisa, todas as interações realizadas na extensão são registradas no banco de dados, incluindo informações como identificador do usuário, enunciado do exercício, perguntas, respostas da IA, papel assumido pelo assistente e dados de data e hora da ação realizada. Esse registro possibilita a análise detalhada do comportamento do sistema e da evolução dos estudantes ao longo das atividades.

De modo geral, a extensão para o VS Code foi projetada para integrar o assistente de

programação diretamente ao ambiente de desenvolvimento do estudante, proporcionando uma experiência contínua e contextualizada. As funcionalidades implementadas buscam equilibrar suporte técnico e mediação pedagógica, favorecendo a reflexão, a autonomia e o desenvolvimento progressivo das habilidades de programação.

5.2.2 Módulo Web do Estudante

O módulo web do estudante foi desenvolvido como um ambiente complementar e alternativo à extensão do VS Code, com o objetivo de ampliar o acompanhamento do processo de aprendizagem e oferecer recursos adicionais de apoio ao estudante durante a resolução de exercícios. Essa interface foi utilizada pelos alunos nas atividades extraclasse do estudo final 2.

Diferentemente da extensão, que está diretamente integrada ao editor de código, esse módulo dispensa a instalação de softwares, compiladores ou configurações adicionais, sendo acessado diretamente por meio de um navegador web, mediante autenticação com e-mail e senha. Nesse ambiente, além da resolução dos exercícios com o apoio da IA, foram incorporados recursos extras que possibilitam ao aluno acompanhar seu progresso, revisar conceitos, acessar materiais da disciplina e praticar atividades de reforço.

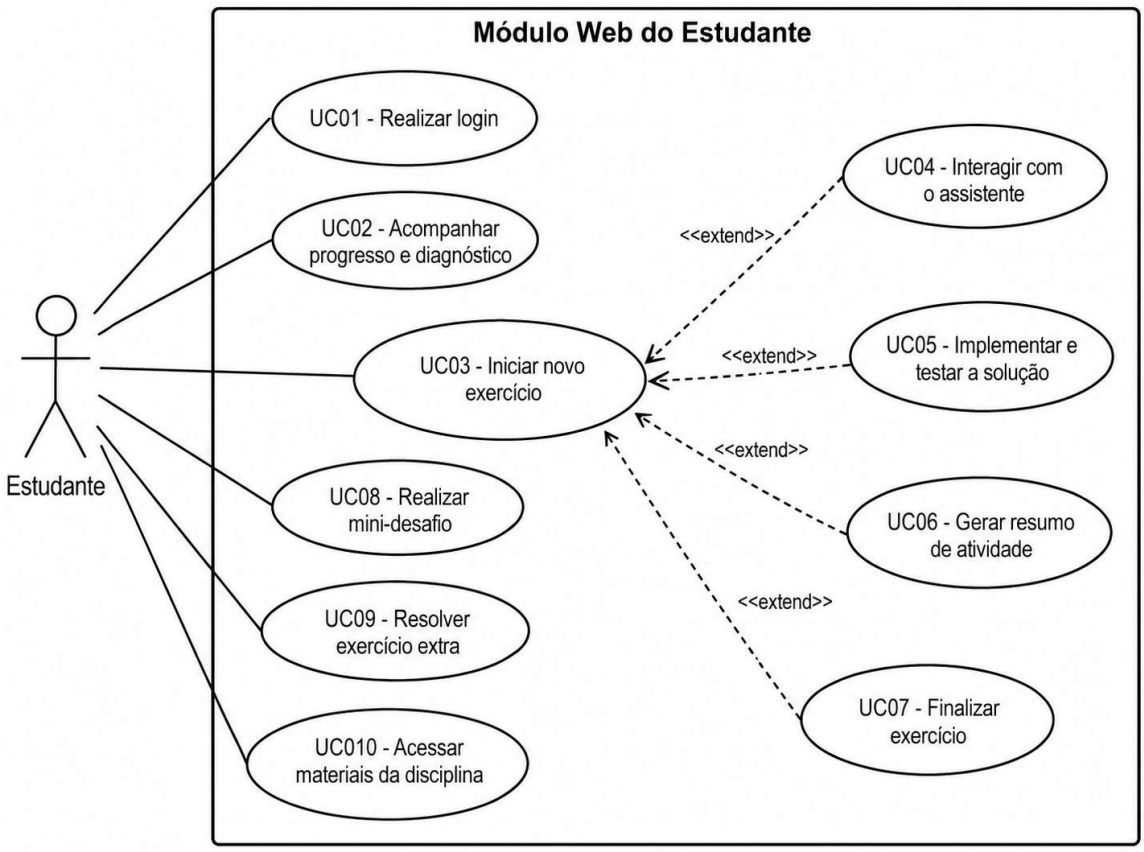
Os principais casos de uso associados ao módulo web do estudante, são apresentados a seguir:

- UC01 - Realizar login: o estudante autentica-se na extensão para iniciar o uso do sistema.
- UC02 - Acompanhar progresso e diagnóstico: o estudante visualiza o painel de inicial (Home), contendo indicadores de interações, principais dificuldades e sugestões de estudo.
- UC03 - Iniciar novo exercício: iniciar uma nova atividade a partir da inserção do enunciado em uma janela modal.
- UC04 - Interagir com o assistente: enviar perguntas ou trechos de código e recebe respostas da IA.
- UC05 – Implementar e testar a solução: o estudante programa no editor integrado, podendo compilar e testar sua solução com apoio da IA.

- UC06 - Gerar resumo de atividade: solicitar ao assistente uma síntese dos conceitos e aprendizados do exercício.
- UC07 - Finalizar exercício: marcar a atividade como concluída.
- UC08 - Realizar mini-desafio: resolver atividades curtas voltadas a habilidades específicas identificadas como deficitárias.
- UC09 - Resolver exercício extra: o estudante solicita a geração de um novo exercício baseado em PBL relacionado a um tema específico.
- UC010 - Acessar materiais da disciplina: visualizar ou realizar o *download* de materiais indexados na base RAG.

A Figura 21 ilustra os principais casos de uso associados ao módulo web do estudante, evidenciando as funcionalidades disponíveis e suas relações no contexto de uso da plataforma.

Figura 21 - Diagrama de casos de uso do módulo web do estudante

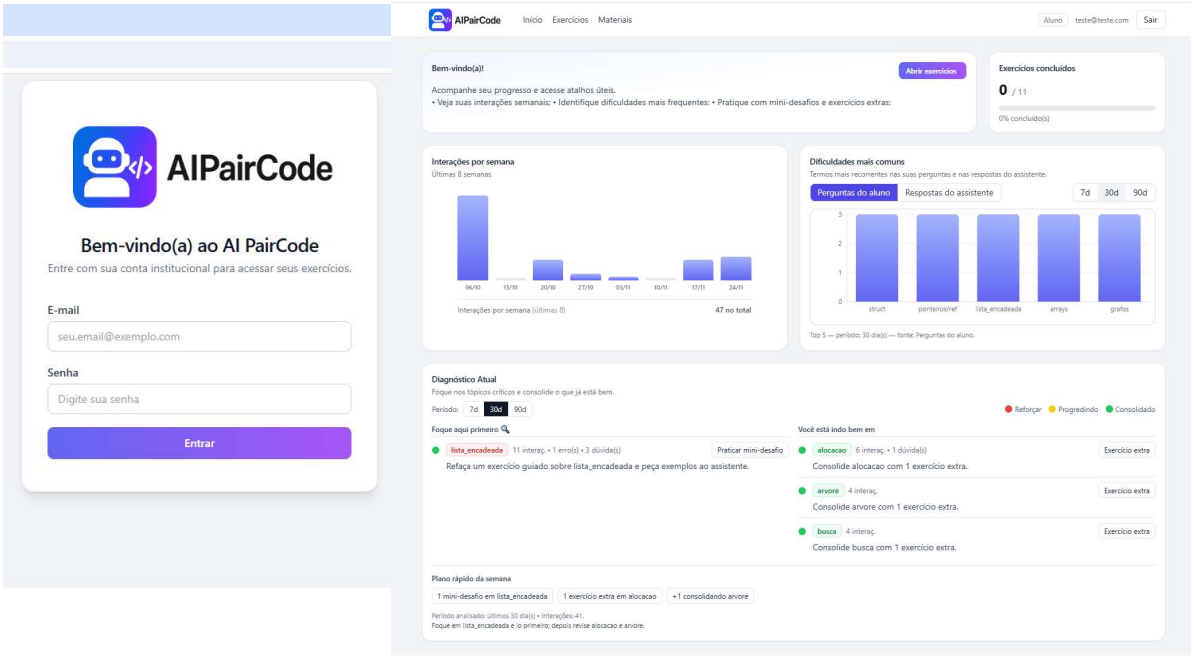


Fonte: Elaborado pela autora (2025)

Após a autenticação no sistema, o estudante é direcionado à página inicial (*Home*), onde são exibidas informações resumidas sobre seu progresso. Na parte superior da página é apresentada a quantidade total de exercícios desenvolvidos com o auxílio do assistente, o

número de exercícios concluídos e um gráfico que ilustra o volume de interações semanais com a IA nas últimas oito semanas. A Figura 22 ilustra a tela de login e a tela inicial do sistema.

Figura 22 - Página de login e interface inicial do módulo web do estudante



Fonte: Elaborado pela autora (2025)

Um dos destaques da página inicial é a apresentação das dificuldades mais comuns, evidenciando os termos e conceitos mais perguntados pelo aluno e também os mais recorrentes nas respostas do assistente em recortes temporais de 7, 30 ou 90 dias. Esses dados são gerados a partir da análise das interações registradas entre o aluno e o assistente, permitindo uma visão mais ampla do processo de aprendizagem. A figura 23 apresenta em detalhes os gráficos de interações por semana e das dificuldades mais frequentes.

Figura 23 - Gráficos de interações por semana e dificuldades mais frequentes do aluno

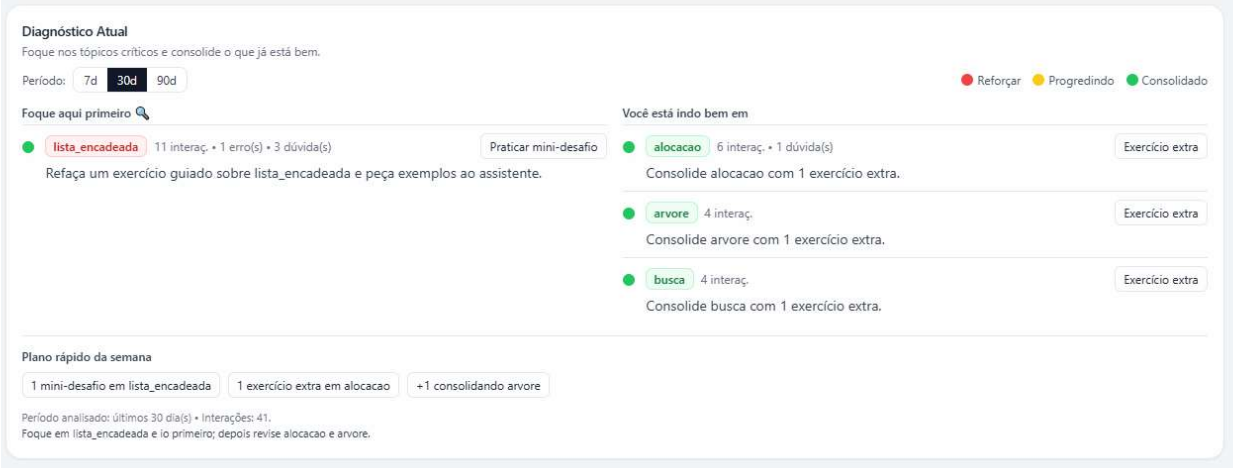


Fonte: Elaborado pela autora (2025)

Outro componente central do módulo web é o diagnóstico de aprendizagem, que classifica os conteúdos em níveis de domínio, como *"Reforçar"*, *"Progredindo"* e *"Consolidado"* (Figura 24). Essa classificação é realizada com base nos exercícios e interações realizados pelo aluno em determinado período onde o sistema identifica os tópicos nos quais houve maior ou menor evolução. Ela é utilizada para orientar o estudante sobre quais conteúdos demandam maior atenção, contribuindo para uma aprendizagem mais direcionada e alinhada às suas necessidades individuais.

Além disso, a partir deste diagnóstico, o sistema apresenta sugestões práticas de estudo denominada "*Plano rápido da semana"*, que sugere situações práticas como a realização de exercícios adicionais ou a revisão de determinados conteúdos, configurando um mecanismo de apoio à autorregulação da aprendizagem.

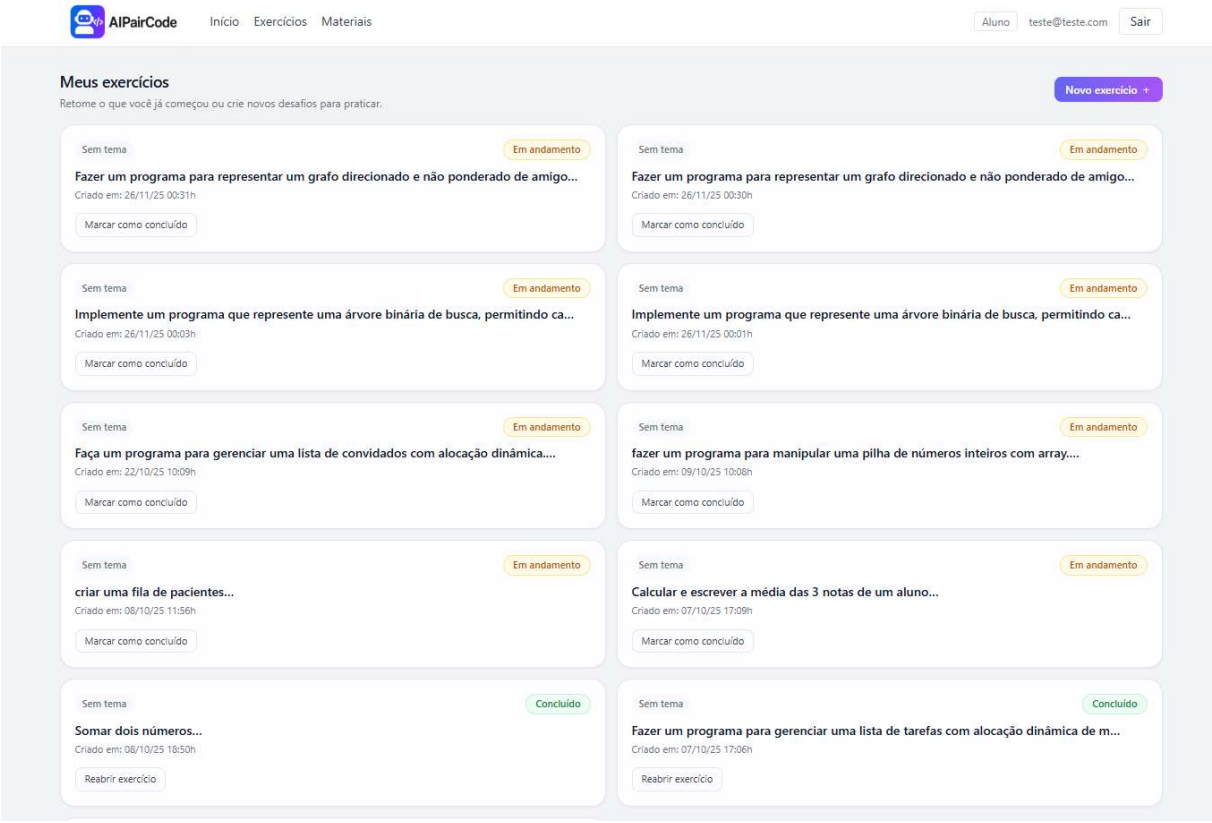
Figura 24 - Diagnóstico de aprendizagem e plano rápido da semana



Fonte: Elaborado pela autora (2025)

A aba de exercícios permite ao estudante criar novas atividades, retomar exercícios anteriores e acompanhar o *status* de cada exercício, que pode estar em andamento ou concluído. A Figura 25 apresenta a interface de listagem de exercícios.

Figura 25 - Lista de exercícios desenvolvidos



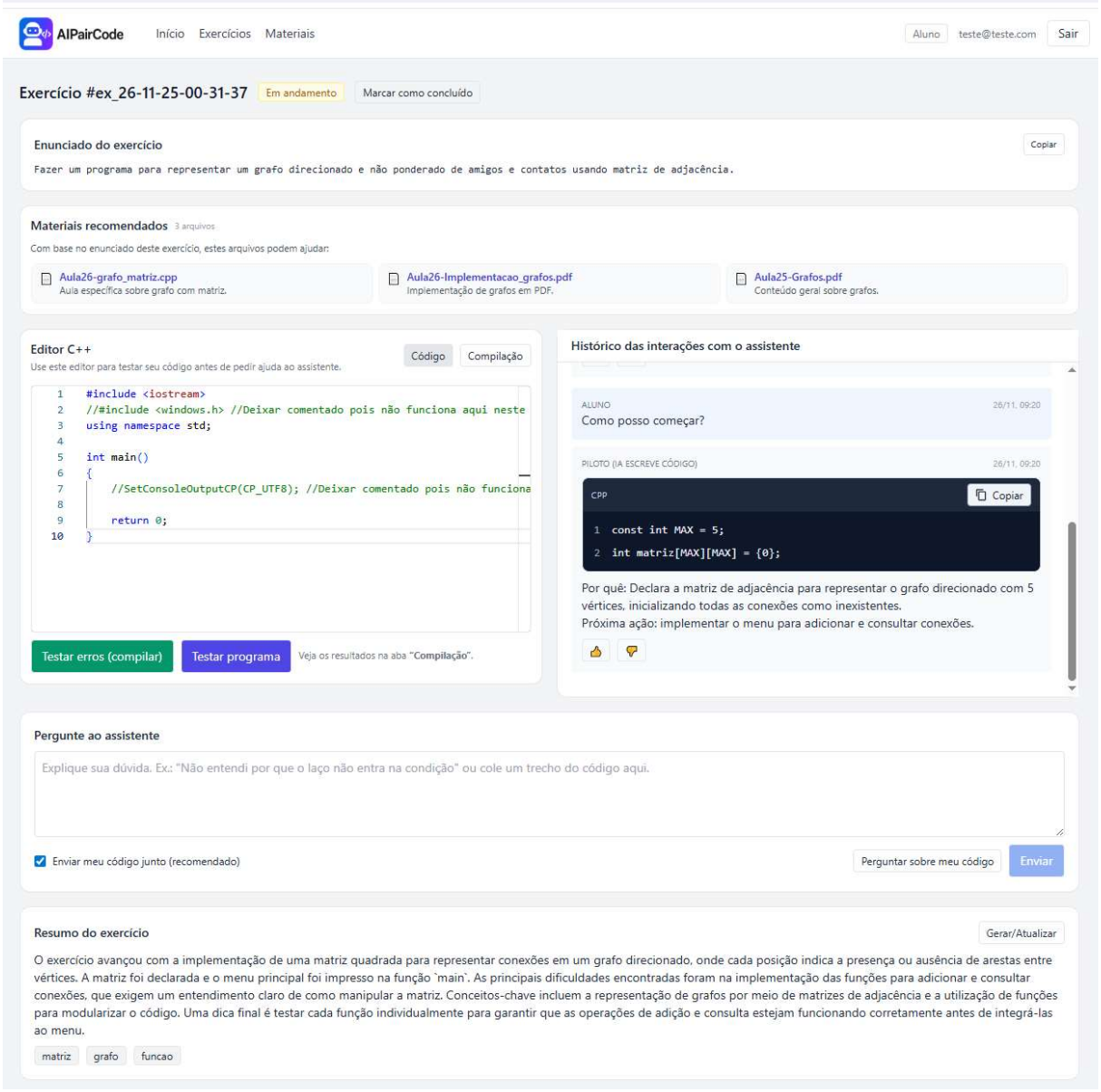
Fonte: Elaborado pela autora (2025)

Ao acessar um exercício específico, o sistema apresenta uma interface integrada composta pelo enunciado do problema, materiais de apoio recomendados (provenientes da base RAG), editor de código e área de *chat* com o histórico de interações com o assistente. Essa organização permite que o estudante concentre todas as atividades relacionadas à resolução do problema em um único ambiente.

Para criar um novo exercício, o aluno insere o enunciado do problema em uma janela *modal*, que é então enviado ao assistente. A partir disso, o sistema identifica, na base RAG, os materiais da disciplina mais relevantes para auxiliar na resolução do exercício e inicializa o ambiente de desenvolvimento, composto pelo editor de código e pela interface de chat do assistente. Os materiais recomendados são apresentados na parte superior da tela, podendo ser acessados pelo estudante sempre que necessário.

A Figura 26 apresenta a interface de resolução de exercícios.

Figura 26 - Interface de resolução de um exercício



Fonte: Elaborado pela autora (2025)

O processo de resolução dos exercícios ocorre de forma integrada com o assistente de programação, permitindo que o estudante formule perguntas, solicite explicações e receba orientações ao longo do desenvolvimento da solução, atuando como um par de programação. Esse processo ocorre de maneira equivalente à extensão do VS Code, uma vez que toda a lógica da interação com a IA (incluindo *prompt*, *guardrails* e acesso aos modelos) é compartilhada entre os módulos por meio de um mesmo *backend*.

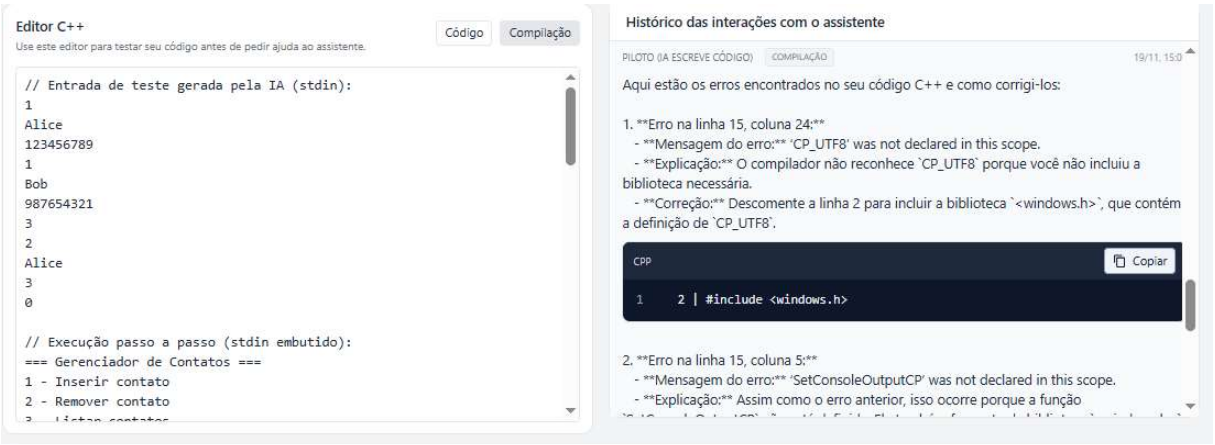
Um aspecto relevante, também presente no módulo web, é a possibilidade de envio do código do editor juntamente com a pergunta, permitindo que o assistente considere o estado atual da solução ao gerar suas respostas.

O módulo web incorpora em sua interface o Monaco Editor[11], um editor de código baseado em navegador, desenvolvido pela Microsoft, que oferece recursos como realce de sintaxe, autocompletar e navegação de código. Este editor permite ao estudante compilar e testar suas soluções por meio dos botões de *"Compilar"* (para verificação de erros de sintaxe, acompanhados de explicações geradas pela IA) e *"Testar"* (para execução de casos de teste automáticos gerados pela IA).

No experimento realizado com a linguagem C++, a execução dos programas foi viabilizada por meio da integração com a API do JDoodle[12], responsável por receber o código submetido, realizar a compilação e execução em ambiente remoto e retornar os resultados ao sistema. Essa mesma arquitetura pode ser estendida para outras linguagens introdutórias utilizadas no assistente, como JavaScript e Python, uma vez que o serviço oferece suporte multilíngue.

Nesse modelo, o código escrito no Monaco Editor é enviado ao *backend*, que encaminha a requisição ao serviço de execução configurado para a linguagem selecionada e devolve a saída do programa, mensagens de erro e demais informações de execução. Esses resultados são então interpretados pelo assistente (IA), que fornece *feedback* ao estudante, auxiliando na identificação de erros e na compreensão do comportamento do programa. A Figura 27 apresenta um exemplo do processo de execução de testes automatizados.

Figura 27 - Exemplo do processo de execução de testes automatizados



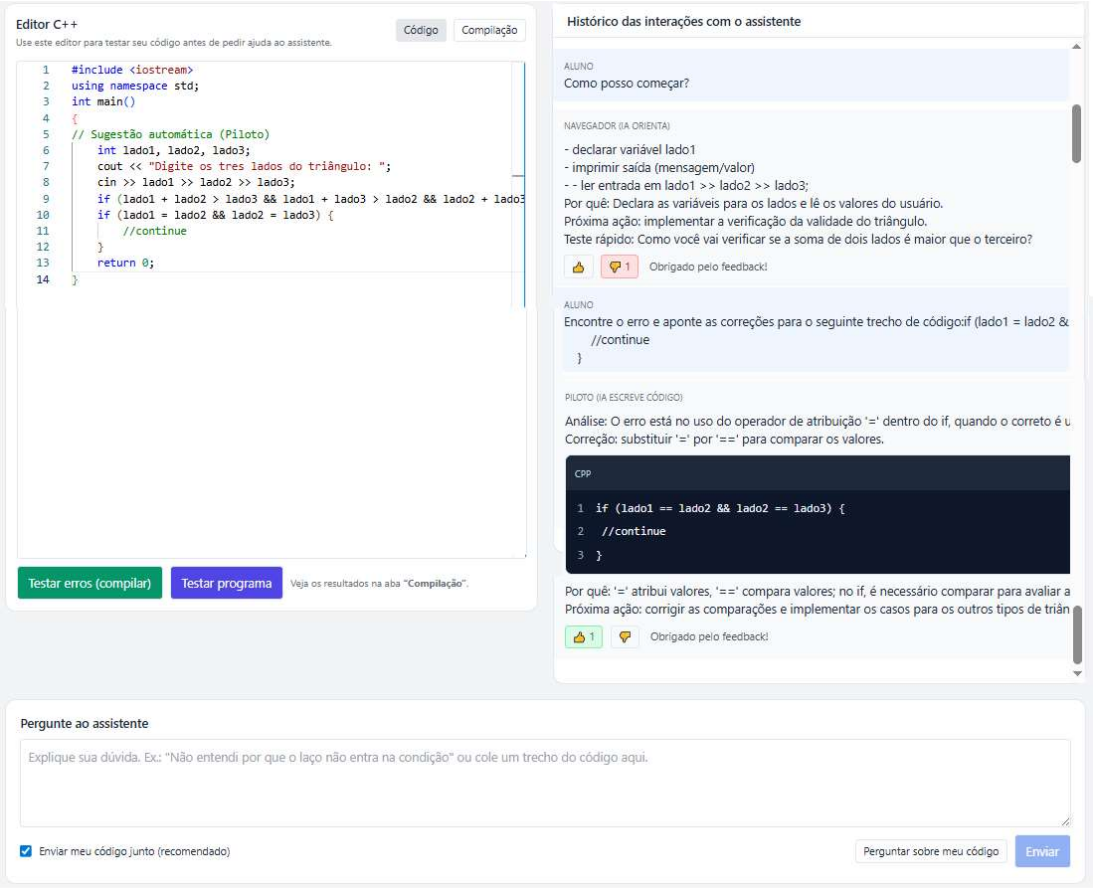
Fonte: Elaborado pela autora (2025)

As interações no chat seguem o mesmo padrão da extensão, incluindo a alternância entre os papéis de navegador e piloto, diferenciação visual das mensagens e estrutura

[11] https://microsoft.github.io/monaco-editor/
[12] https://www.jdoodle.com/

padronizada das respostas (Figura 28). O estudante também pode avaliar as respostas da IA por meio de ícones de *feedback*, positivo ou negativo, e também uma mensagem explicando o porquê de sua avaliação. Essa avaliação é opcional, e o aluno recebe uma mensagem de confirmação caso a envie.

Figura 28 - Interface de *chat* do módulo web do estudante



Fonte: Elaborado pela autora (2025)

Ao finalizar um exercício, o estudante pode gerar um resumo por meio do botão *"Gerar/Atualizar"* que aciona o assistente para sintetizar os conceitos envolvidos, as ações realizadas e possíveis próximos passos. Esse processo também gera *tags* associadas ao conteúdo, utilizadas para alimentar o diagnóstico de aprendizagem.

Além do suporte à resolução de exercícios, o sistema oferece recursos adaptativos voltados à prática e consolidação do conhecimento. Entre eles, destacam-se os mini desafios, que são atividades curtas focadas em habilidades específicas identificadas como pontos de dificuldade do estudante. Neste tipo de atividade o aluno recebe um código base, deve completar a função, pode pedir dicas e ao final o código é enviado para análise da IA que devolverá uma resposta apropriada. A Figura 29 apresenta um exemplo de mini desafio gerado pelo sistema.

Figura 29 - Mini desafios



Fonte: Elaborado pela autora (2025)

De forma complementar, o sistema disponibiliza a opção de gerar exercícios extras baseados no conteúdo trabalhado. Esses exercícios são gerados usando a metodologia de aprendizagem baseada em problemas (ABP), tendo como objetivo promover a aplicação prática dos conceitos em situações mais próximas do mundo real. O aluno tem a autonomia de *"Aceitar e Resolver"* o problema sugerido ou clicar em "*Gerar outro"*, solicitando que a IA elabore uma nova proposta dentro do mesmo escopo. A Figura 30 ilustra exemplos de exercícios extras disponibilizados ao estudante.

Figura 30 - Exercícios extras



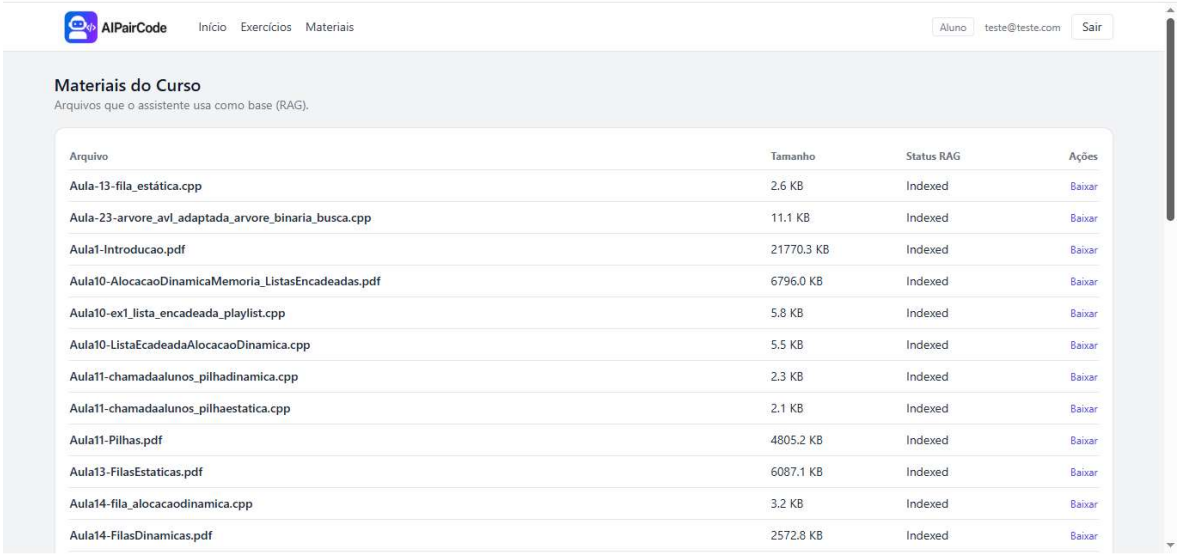
Fonte: Elaborado pela autora (2025)

Outro elemento importante é a integração com materiais da disciplina por meio da abordagem RAG. O sistema disponibiliza ao estudante os conteúdos utilizados na disciplina,

como slides, códigos e materiais de apoio, os quais também são utilizados pelo assistente para fundamentar suas respostas. O aluno pode fazer *download* desses materiais e, sempre que ele cria um novo exercício, recebe sugestões de materiais de apoio que podem ser úteis para sua resolução.

Essa integração contribui para que as orientações fornecidas pela IA estejam alinhadas ao conteúdo trabalhado em aula, reduzindo inconsistências e aumentando a relevância pedagógica das respostas. A Figura 31 apresenta a interface de acesso aos materiais da disciplina.

Figura 31 - Materiais da disciplina



Fonte: Elaborado pela autora (2025)

De modo geral, o módulo web do estudante não se limita a oferecer um ambiente de resolução de exercícios, mas constitui um espaço de apoio à aprendizagem contínua, no qual o estudante pode acompanhar sua evolução, identificar dificuldades e receber recomendações personalizadas. Ao integrar diagnóstico, prática guiada, materiais de apoio e interação com o assistente, o módulo reforça a proposta do AIPairCode como um mediador do processo de construção do conhecimento.

5.2.3 Módulo Web do Professor

O módulo web do professor foi desenvolvido com o objetivo de possibilitar o acompanhamento do processo de aprendizagem dos estudantes, oferecendo recursos de análise, monitoramento e gestão das interações realizadas no sistema. Esse módulo tem como

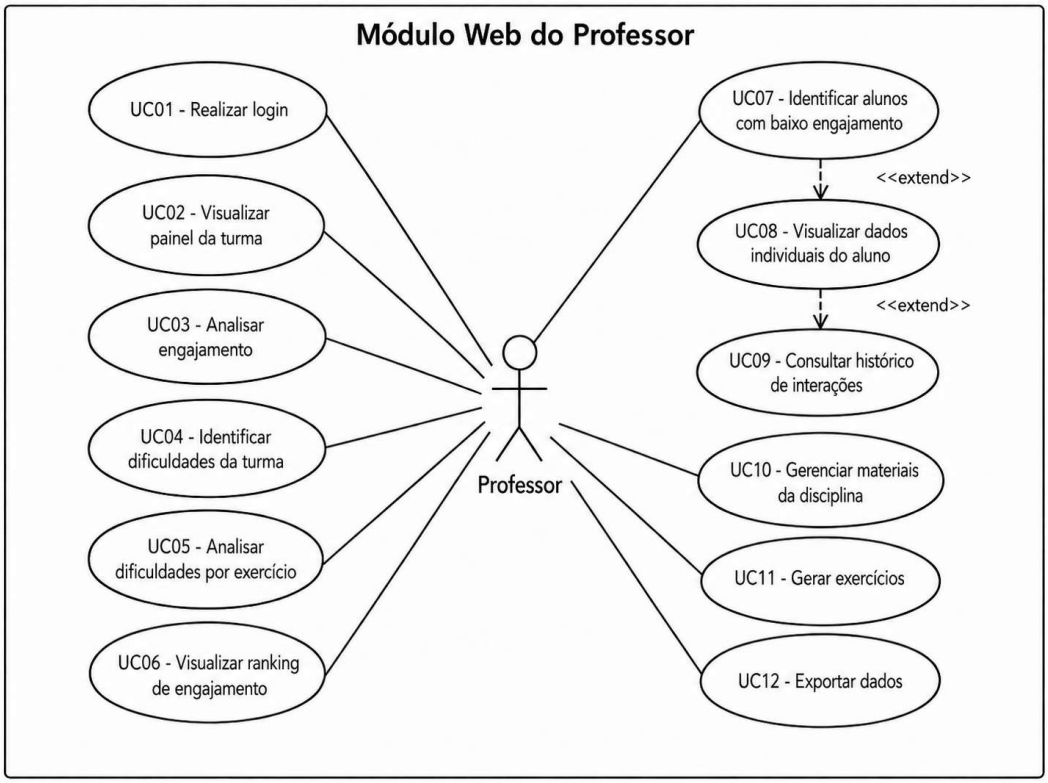
foco a visualização de dados agregados da turma, permitindo identificar padrões de uso, dificuldades recorrentes e níveis de engajamento dos alunos. Os principais casos de uso associados ao módulo web do professor são apresentados a seguir:

- UC01 - Realizar login: o professor autentica-se no sistema para acessar o painel.
- UC02 - Visualizar painel da turma: consultar indicadores gerais, como número de alunos, exercícios realizados, concluídos e interações.
- UC03 - Analisar engajamento: acompanhar a evolução das interações ao longo do tempo.
- UC04 - Identificar dificuldades da turma: visualizar as principais dificuldades organizadas por categorias de conteúdo.
- UC05 - Analisar dificuldades por exercício: investigar, de forma detalhada, os tópicos que geraram maior número de dúvidas em um exercício específico.
- UC06 - Visualizar *ranking* de engajamento: acompanhar a participação dos alunos com base em exercícios, interações e atividades concluídas.
- UC07 - Identificar alunos com baixo engajamento: visualizar alertas automáticos com base em critérios definidos pelo sistema.
- UC08 - Visualizar dados individuais do aluno: acessar informações específicas de cada estudante, incluindo exercícios e progresso.
- UC09 - Consultar histórico de interações: analisar as interações entre aluno e assistente em um exercício específico.
- UC10 - Gerenciar materiais da disciplina: enviar e visualizar materiais didáticos, além de e indexar materiais no RAG disponibilizando conteúdos para uso pelo assistente como base de conhecimento.
- UC11 - Gerar exercícios: gerar novos exercícios baseados no RAG e nas principais dificuldades da turma.
- UC12 - Exportar dados: realizar a exportação de relatórios para análise externa (.csv).

A Figura 32 apresenta o diagrama de casos de uso do módulo web do professor, sintetizando as principais funcionalidades disponíveis para o acompanhamento pedagógico da turma. O diagrama evidencia as ações realizadas pelo professor no sistema, incluindo a

visualização de indicadores gerais, a análise do engajamento, a identificação de dificuldades, o acompanhamento individual dos alunos, o gerenciamento de materiais da disciplina, a geração de exercícios e a exportação de dados. Além disso, representa a relação entre os casos de uso voltados ao acompanhamento individual, indicando que a visualização dos dados individuais do aluno ocorre a partir da identificação de estudantes com baixo engajamento, e que a consulta ao histórico de interações é realizada a partir da análise dos dados individuais do estudante.

Figura 32 - Diagrama de casos de uso do módulo web do professor



Fonte: Elaborado pela autora (2025)

Após realizar o login no sistema, o professor é direcionado ao painel principal, no qual são exibidos indicadores gerais da turma. Esses indicadores incluem o número de alunos vinculados, a quantidade total de exercícios realizados, o número de exercícios concluídos e o volume de interações com o assistente em um determinado período.

O painel permite a seleção do período de análise, podendo o professor optar por intervalos predefinidos (7, 30 ou 90 dias) ou definir um período personalizado, o que possibilita analisar diferentes momentos da disciplina. Além disso, os dados podem ser exportados em formato CSV para análises externas.

A Figura 33 ilustra a interface do painel do professor, destacando os principais indicadores e visualizações disponíveis.

Figura 33 - Página inicial do professor com indicadores gerais da turma



Fonte: Elaborado pela autora (2025)

A análise das interações é apresentada por meio de um gráfico de tendência, que exibe a evolução temporal do número de interações dos estudantes com o assistente ao longo do período selecionado. Esse recurso permite identificar momentos de maior engajamento, geralmente associados à realização de atividades práticas, bem como, períodos de baixa utilização do sistema. Complementarmente, um resumo apresenta indicadores como pico de interações, número de dias com atividade e média de interações por dia ativo. A Figura 34 apresenta o gráfico e o resumo das interações.

Figura 34 - Gráfico de resumo das interações

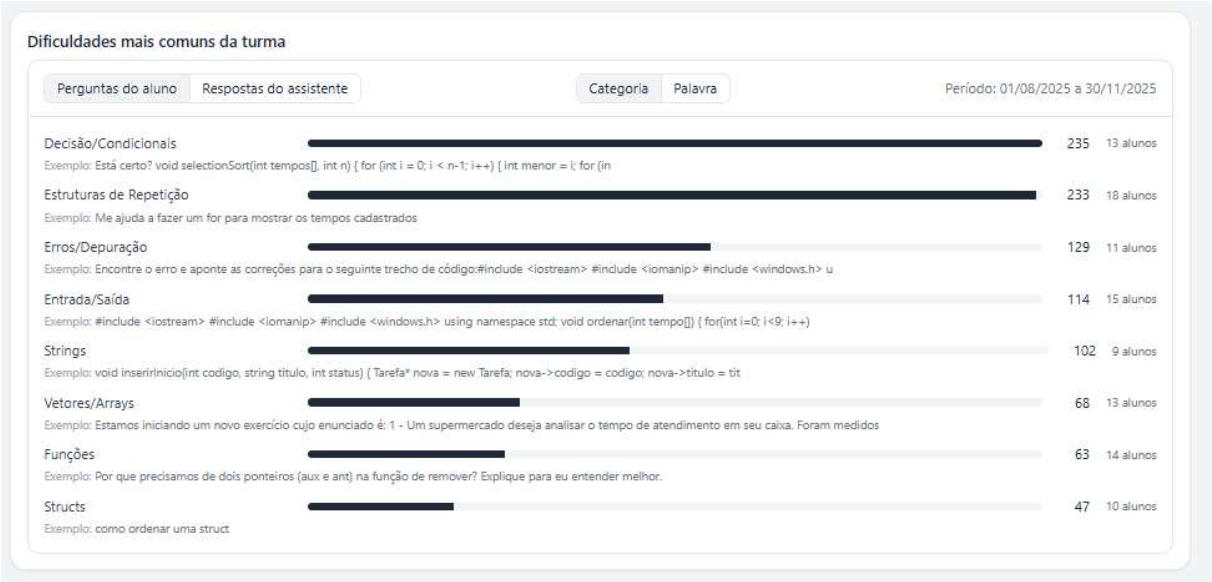


Fonte: Elaborado pela autora (2025)

O sistema apresenta uma seção dedicada à identificação das dificuldades mais comuns da turma, organizada por categorias de conteúdo, como estruturas de repetição, decisão, vetores, entrada/saída, depuração de erros, entre outras. Essa visualização pode ser alternada entre diferentes perspectivas, como perguntas dos alunos ou respostas do assistente, e permite ainda a análise por categoria ou por palavras-chave.

Cada categoria apresenta exemplos reais de dúvidas dos estudantes, permitindo ao professor compreender o tipo de dificuldade enfrentada. Essa classificação é construída automaticamente a partir da análise das interações, auxiliando na identificação de tópicos que demandam maior atenção pedagógica. A Figura 35 apresenta a visualização das dificuldades mais frequentes da turma.

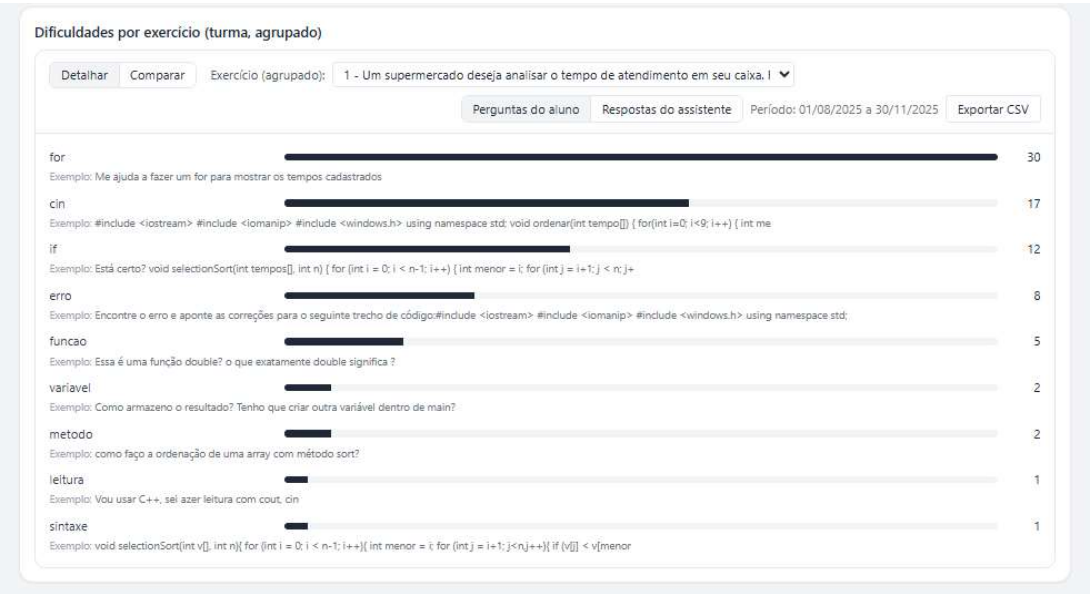
Figura 35 - Visualização das dificuldades mais frequentes da turma



Fonte: Elaborado pela autora (2025)

Outro recurso relevante do módulo é a análise de dificuldades por exercício (Figura 36), que permite ao professor selecionar um exercício específico e visualizar os principais tópicos que geraram dúvidas. Essa funcionalidade possibilita a comparação entre exercícios e a identificação de padrões de dificuldade, contribuindo para o planejamento de intervenções pedagógicas mais direcionadas.

Figura 36 - Visualização das dificuldades por exercício



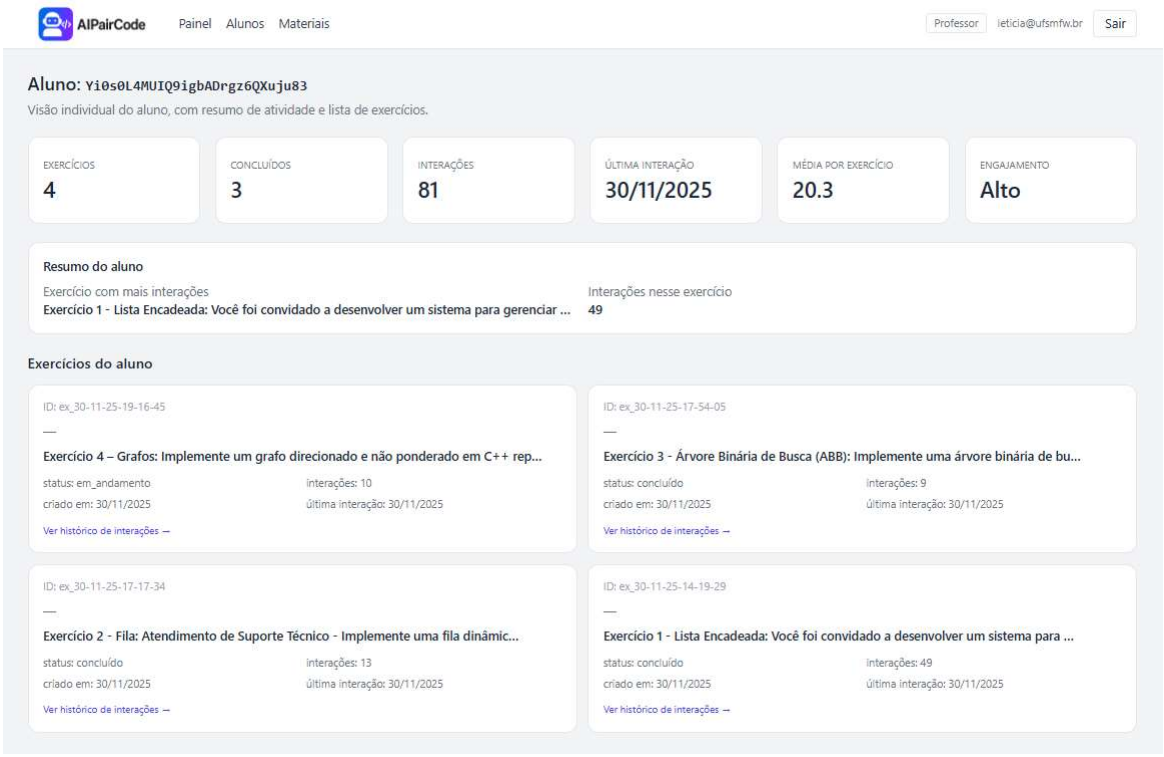
Fonte: Elaborado pela autora (2025)

O sistema também disponibiliza um *ranking* de engajamento dos estudantes, no qual

são apresentados indicadores como número de exercícios realizados, quantidade de interações com o assistente e atividades concluídas. Esse recurso permite ao professor identificar alunos com maior participação, bem como aqueles que apresentam baixo engajamento. Complementarmente, o sistema gera automaticamente alertas de baixo engajamento, com base em critérios como número reduzido de interações em um determinado período ou ausência de exercícios concluídos. Esses alertas auxiliam o professor na identificação precoce de alunos que podem necessitar de acompanhamento mais próximo.

Além da análise agregada da turma, o módulo permite ao professor acessar informações individuais dos estudantes. A partir da lista de alunos, é possível selecionar um estudante específico e visualizar seus exercícios, status de conclusão e nível de atividade. Para apoiar o acompanhamento individual dos estudantes, o módulo professor apresenta um indicador de engajamento (alto, médio ou baixo), calculado com base na quantidade de exercícios iniciados, número de interações com o assistente e existência de atividades concluídas. A Figura 37 apresenta a interface de visualização de um aluno específico, na qual é possível consultar os exercícios realizados, seus respectivos *status* e acessar o histórico de interações associadas a cada atividade.

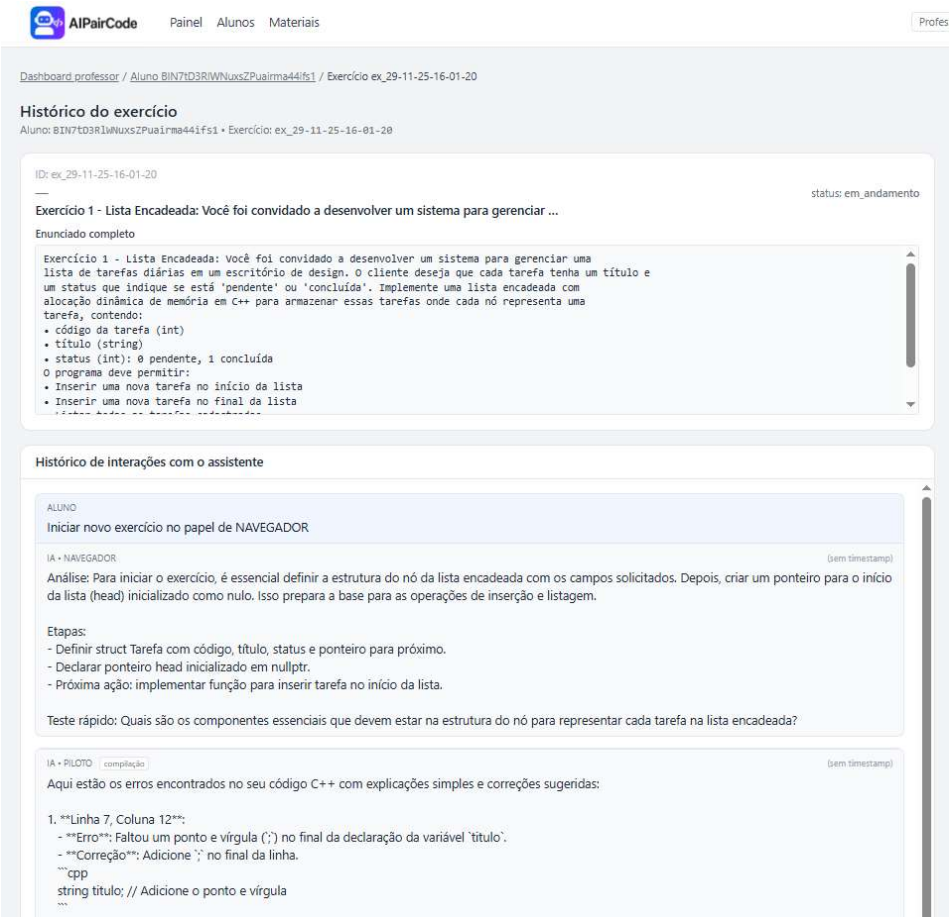
Figura 37 - Página de acompanhamento individual de um aluno



Fonte: Elaborado pela autora (2025)

Ao acessar um exercício específico, o professor pode visualizar o histórico completo das interações entre o estudante e o assistente, incluindo perguntas, respostas e evolução da solução. Essa funcionalidade permite uma análise qualitativa do processo de aprendizagem, possibilitando compreender não apenas o resultado final, mas também as estratégias utilizadas pelo aluno durante a resolução do problema. A Figura 38 ilustra a visualização do histórico de interações de um exercício.

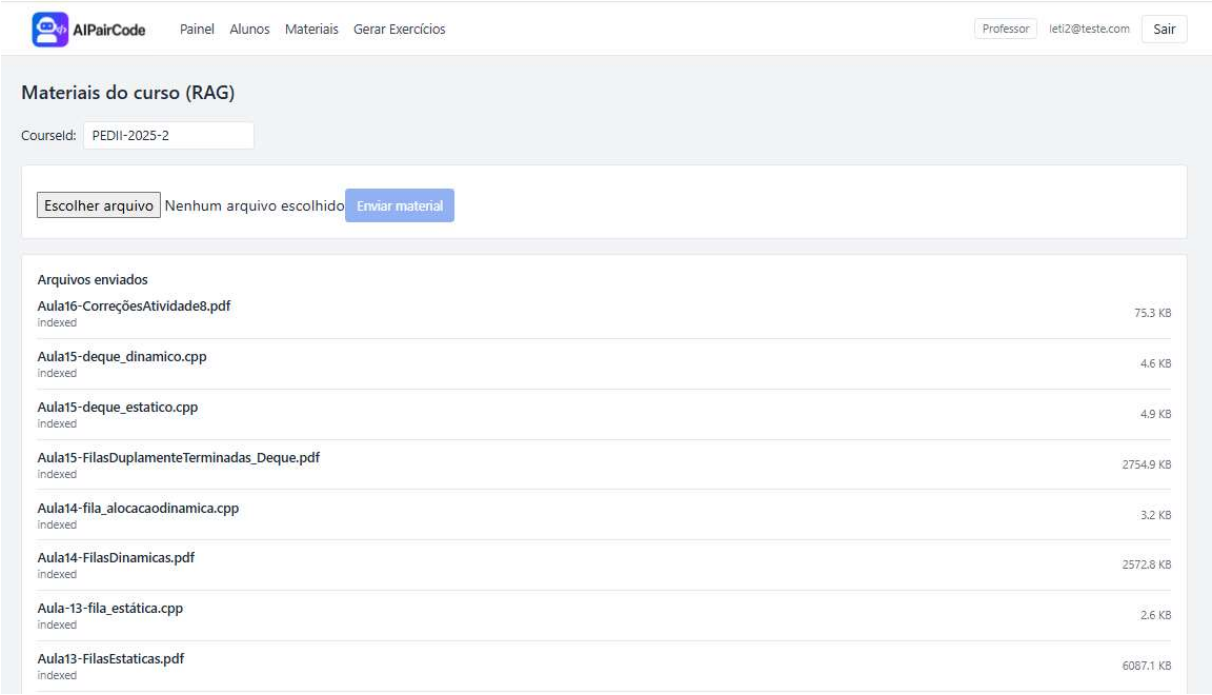
Figura 38 - Histórico de interações entre aluno e assistente



Fonte: Elaborado pela autora (2025)

Outro componente importante do módulo é a gestão de materiais didáticos, integrada ao mecanismo de Recuperação Aumentada por Geração (RAG). Por meio dessa funcionalidade, o professor pode enviar arquivos da disciplina, como slides, códigos e documentos de apoio, que são armazenados no sistema e indexados para utilização pelo assistente. A Figura 39 apresenta a interface de envio e gerenciamento de materiais da disciplina.

Figura 39 - Interface de gerenciamento de materiais da disciplina (RAG)



Fonte: Elaborado pela autora (2025)

Os materiais enviados passam por um processo de indexação na base de conteúdos da *Assistants* API, permitindo que sejam recuperados e utilizados como contexto nas respostas do assistente. Dessa forma, o professor tem controle direto sobre a base de conhecimento utilizada pelo sistema, garantindo que as orientações fornecidas aos estudantes estejam alinhadas ao conteúdo trabalhado na disciplina.

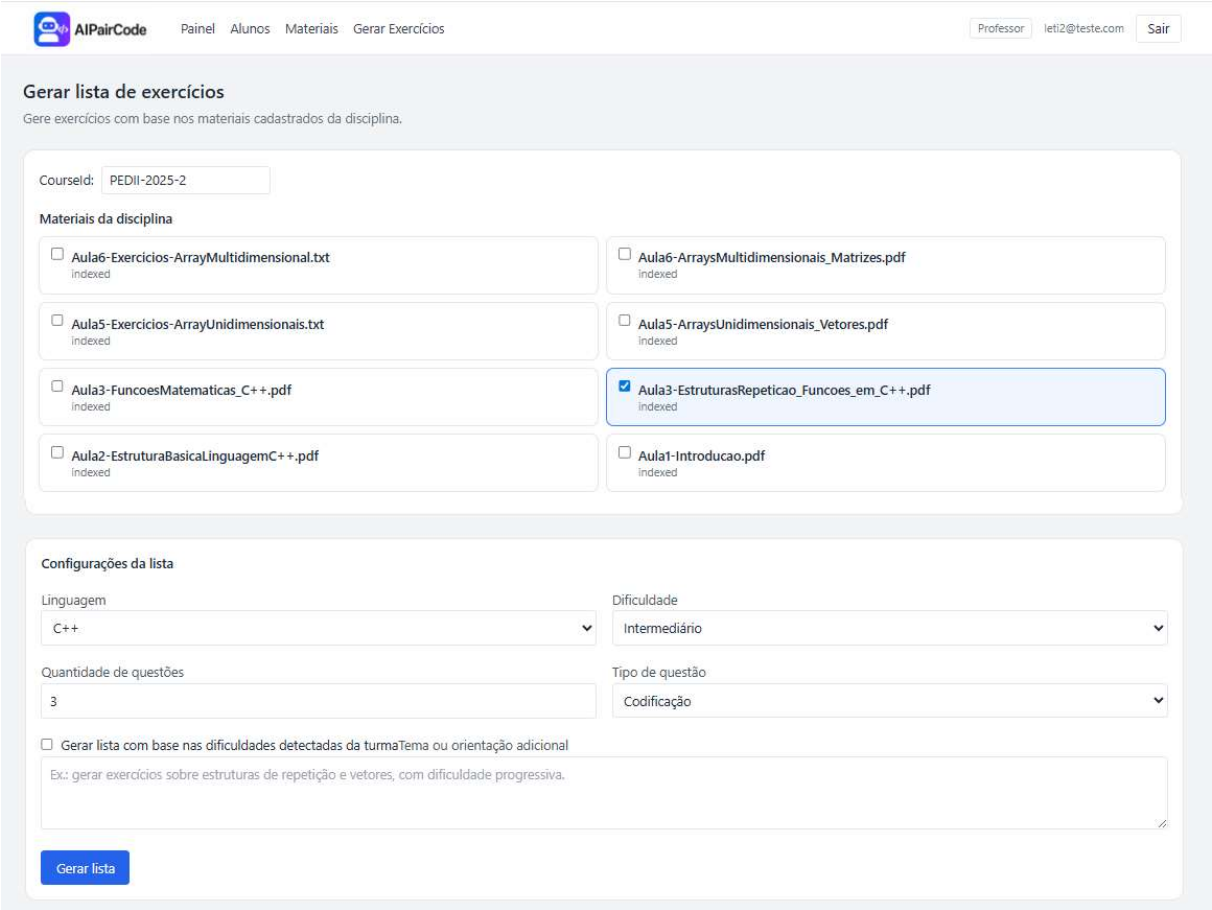
O módulo professor também incorpora uma funcionalidade de geração automática de listas de exercícios apoiada por IA. Esse recurso foi desenvolvido com o objetivo de auxiliar o docente na elaboração de atividades alinhadas aos conteúdos da disciplina, utilizando como base os materiais previamente cadastrados no sistema e indexados no mecanismo de RAG.

O sistema também prevê a geração adaptativa de exercícios com base nas dificuldades identificadas nas interações da turma com o assistente. Nesse cenário, os tópicos mais recorrentes observados no painel de dificuldades são utilizados como parâmetros adicionais para a elaboração das atividades, permitindo ao professor gerar listas direcionadas às necessidades específicas dos estudantes.

Para utilizar essa funcionalidade, o professor seleciona os materiais da disciplina que servirão como referência para a geração da lista, podendo incluir slides, apostilas, exercícios anteriores ou outros documentos didáticos previamente enviados ao sistema. Além disso, o docente pode configurar diferentes parâmetros pedagógicos, como linguagem de programação

(por exemplo, C++, JavaScript ou Python), quantidade de questões, nível de dificuldade (iniciante, intermediário e avançado) e tipo de exercício desejado, incluindo questões de codificação, interpretação de código, correção de erros, exercícios conceituais ou listas mistas. A Figura 40 apresenta a interface de configuração utilizada para a geração das listas de exercícios.

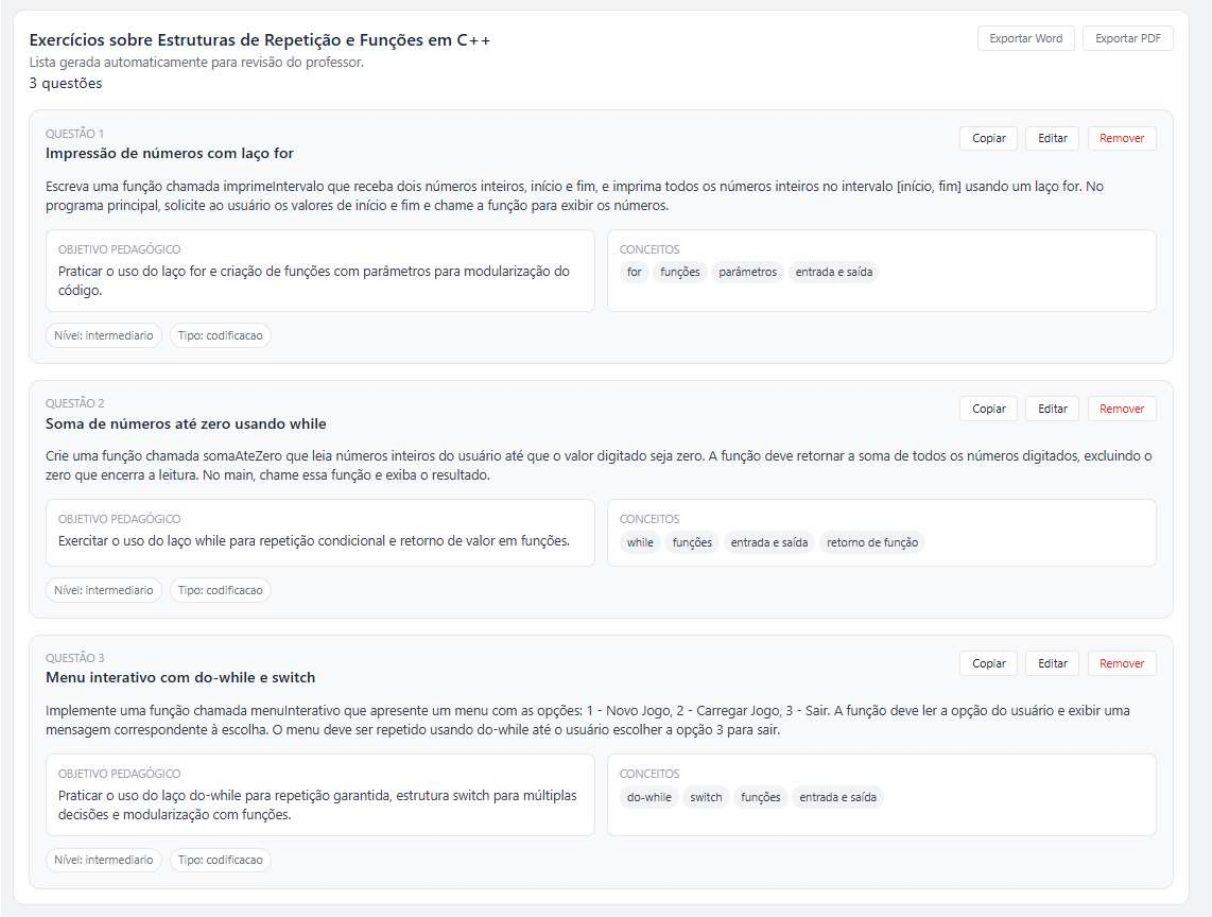
Figura 40 - Interface geração da lista de exercícios



Fonte: Elaborado pela autora (2025)

Após a definição dos parâmetros, o sistema utiliza os materiais selecionados como contexto para a IA, gerando automaticamente uma lista de exercícios coerente com os conteúdos abordados na disciplina. Durante esse processo, a IA busca respeitar os conceitos presentes nos materiais fornecidos, evitando introduzir tópicos avançados ou não trabalhados pelo professor. As listas geradas são apresentadas ao professor em uma interface de revisão, permitindo editar, remover ou copiar questões individualmente antes de sua utilização. O sistema também oferece opções de exportação da lista em formatos como PDF e Word. A Figura 41 apresenta um exemplo da interface de visualização e revisão da lista de exercícios gerada automaticamente.

Figura 41 - Lista de exercícios gerada



Fonte: Elaborado pela autora (2025)

De modo geral, o módulo web do professor amplia a capacidade de acompanhamento do processo de aprendizagem, oferecendo indicadores quantitativos de uso e interação dos alunos e também dados qualitativos para a análise dessas interações. Assim, o módulo contribui para a prática pedagógica, permitindo ao professor tomar decisões e adaptar suas estratégias de ensino às necessidades da turma e dos indivíduos, permitindo um ensino mais personalizado.

Embora o módulo professor tenha sido desenvolvido como componente integrante do AIPairCode, nos estudos finais 1 e 2, apresentados nesta tese, sua utilização ocorreu exclusivamente como ferramenta de monitoramento e acompanhamento das interações realizadas pelos estudantes. O painel foi empregado pela própria pesquisadora/docente para consulta de métricas, histórico de atividades e identificação de dificuldades recorrentes da turma.

Como não era um dos objetivos iniciais da pesquisa, não foram conduzidos testes formais de usabilidade, estudos de aceitação tecnológica ou avaliações sistemáticas com

professores e usuários externos. Dessa forma, essa tese não permite inferir, neste momento, a efetividade ou a usabilidade do módulo professor sob a perspectiva docente, aspecto que permanece como oportunidade para investigações futuras.

5.3 Avaliação Inicial do AIPairCode

Visando verificar a usabilidade e o funcionamento do assistente de programação antes de sua aplicação para todos os alunos dos estudos finais, foi conduzido um estudo piloto no início do mês de março de 2025. Participaram dessa etapa três alunos voluntários, que interagiram individualmente com o protótipo do AIPairCode em uma sessão presencial em laboratório, acompanhada pela pesquisadora. Durante a atividade, os estudantes utilizaram a ferramenta para simular situações de resolução de problemas de programação em JavaScript.

Essa etapa teve como finalidade identificar possíveis ajustes tanto nas funcionalidades técnicas do sistema quanto nas estratégias de mediação utilizadas pelo assistente antes de sua aplicação com toda a turma participante do estudo principal.

Ao término da atividade, os estudantes responderam ao questionário SUS, utilizado para avaliação da usabilidade de sistemas interativos (BROOKE, 1996). A pontuação média obtida a partir das respostas dos alunos ao questionário foi de 69,25, em uma escala que varia de 0 a 100, indicando um nível de usabilidade considerado aceitável para um protótipo em fase inicial.

Para complementar o questionário, foi incluída uma questão aberta solicitando sugestões de melhoria para o comportamento do AIPairCode. De modo geral, os participantes apontaram a necessidade de maior equilíbrio entre perguntas reflexivas e respostas mais diretas em determinados momentos da interação. Alguns estudantes relataram que o assistente frequentemente devolvia perguntas em sequência, o que, em alguns casos, dificultava o avanço na resolução do problema. Outros sugeriram que as respostas fossem apresentadas de forma mais gradual, evitando a exposição de muitas informações simultaneamente.

Também foram mencionadas sugestões relacionadas a funcionalidades desejáveis para a ferramenta, como a possibilidade do assistente avaliar o código em tempo real e sugerir melhorias durante a escrita, ampliando o apoio ao processo de desenvolvimento. Apesar dessas observações, os participantes ressaltaram a utilidade e a aplicabilidade do recurso como suporte à aprendizagem, especialmente pelo incentivo à construção do raciocínio sem a entrega imediata de soluções prontas. As respostas são apresentadas no Quadro 2.

Quadro 2 - Sugestões de melhoria indicadas pelos participantes do estudo piloto

Você tem alguma sugestão de melhoria para o comportamento do assistente?
"Talvez ele dar algumas respostas mais objetivas ao invés de sempre perguntar a forma como o usuário está pensando. E que fosse avaliando meu código em tempo real e dando ideias e melhorias para o mesmo(enquanto vou escrevendo)."
"Minha única sugestão talvez seria melhor ritmar as respostas, ou seja não fornecer tantos dados de uma vez para o usuário, sabe instigar a pensar um pouco mais, mas no geral em minha percepção achei extremamente útil e aplicável como assistente de aprendizagem."
"As dicas costumam ser bem abertas, frequentemente me respondeu com outra pergunta, às vezes parecida com a mesma que fiz, pode ter me confundido um pouco durante o uso... As respostas poderiam ser mais diretas e práticas, ex: código demonstrando soluções ou pensamento que ela teve. Mas continuando na conversa me ajudou bastante pois construir o raciocínio é melhor e não entregou a resposta de bandeja."

Fonte: Elaborado pela autora (2025)

Além do questionário aplicado aos estudantes, a observação da pesquisadora durante as sessões de interação forneceu evidências complementares sobre dificuldades recorrentes dos estudantes, erros e falhas da ferramenta, problemas de interface e padrões de engajamento do assistente. Essas observações permitiram identificar oportunidades de melhoria tanto no funcionamento técnico da ferramenta quanto nas estratégias de mediação adotadas pelo sistema.

Com base nesses dados, foram realizadas duas modificações principais no sistema: (i) revisão do *prompt* responsável por orientar o comportamento do assistente, buscando aprimorar a clareza das instruções e o equilíbrio entre questionamento reflexivo e orientação mais direta; e (ii) melhorias na interface gráfica, visando tornar a experiência de uso mais intuitiva e facilitar a interação dos alunos com o assistente durante a resolução dos exercícios.

Assim, o estudo piloto desempenhou um papel importante na validação inicial do protótipo, permitindo identificar e corrigir aspectos técnicos e pedagógicos do AIPairCode antes de sua aplicação em um contexto mais amplo. As modificações implementadas a partir dessa etapa contribuíram para tornar o sistema mais robusto e melhor alinhado aos objetivos educacionais da pesquisa. Com esses ajustes, foi possível avançar para os estudos finais.

6. RESULTADOS

Conforme citado na seção de metodologia, esta pesquisa foi desenvolvida com a aplicação de três estudos empíricos com os alunos. O estudo preliminar, aplicado antes do desenvolvimento do AIPairCode, teve caráter exploratório, envolvendo análise descritiva dos dados quantitativos do questionário de motivação e análise qualitativa das interações com o ChatGPT e das respostas as perguntas abertas. Os Estudos 1 e 2, aplicados a partir do desenvolvimento do assistente, que adotaram um delineamento quase-experimental, com grupo controle e grupo experimental, pré-teste e pós-teste, sendo realizadas análises inferenciais dos dados quantitativas dos questionários IMI e SUS, além de análise qualitativa das interações com o AIPairCode e das respostas as perguntas abertas, visando à triangulação dos resultados.

Nas seções seguintes serão apresentados os dados de cada um dos três estudos realizados.

6. 1 Estudo Preliminar: Uso de IA como Par de Programação

Com o objetivo de explorar padrões de uso de ferramentas de IA generativa no contexto do ensino de programação e obter subsídios para o refinamento da proposta desta tese, foi realizado um estudo preliminar de caráter exploratório com uma turma do segundo semestre do Curso de Sistemas de Informação da Universidade Federal de Santa Maria (UFSM), Campus Frederico Westphalen.

O estudo teve como propósito investigar percepções, padrões de interação e estratégias de uso dessas ferramentas quando utilizadas como par de programação, buscando compreender de que forma os estudantes recorrem à IA durante a resolução de problemas de programação. Os resultados obtidos contribuíram para identificar potencialidades, limitações e aspectos relevantes para o desenho de sistemas de apoio à aprendizagem baseados em IA, além de orientar ajustes no desenvolvimento do assistente de programação proposto nesta pesquisa.

O estudo foi conduzido entre setembro e outubro de 2024, envolvendo 37 estudantes matriculados na disciplina de Programação e Estrutura de Dados II, que estavam iniciando o aprendizado da linguagem *C++*. Os participantes possuíam experiência prévia em programação com a linguagem Python, adquirida no primeiro semestre do curso.

Para a realização do estudo, os alunos foram divididos em dois grupos, tendo como critério a nota obtida na disciplina de Programação e Estrutura de Dados I, cursada no primeiro semestre do curso. As notas foram ordenadas em ordem decrescente, e os estudantes foram distribuídos alternadamente entre os grupos, buscando equilibrar o desempenho prévio e promover maior homogeneidade inicial entre eles. Trata-se, portanto, de uma estratégia de balanceamento, e não de uma distribuição aleatória.

O Grupo 1, composto por 13 estudantes, realizou a atividade utilizando o ChatGPT como par de programação, configurando uma interação do tipo Aluno+IA. Já o Grupo 2, com 24 estudantes (formando 12 pares), realizou a mesma atividade em formato de par programação Aluno+Aluno, seguindo os princípios tradicionais da programação em pares, com a alternância entre os papéis de piloto e navegador durante a resolução das tarefas. A distribuição dos grupos está descrita na Tabela 2.

Tabela 2 - Resumo da participação dos estudantes no Estudo Preliminar

Grupos	**Pares**	**Quantidade de alunos**
Grupo 1: Pares de programação Aluno + IA	13	13
Grupo 2: Pares de programação Aluno + Aluno	12	24
Total	**25**	**37**

Fonte: Elaborado pela autora (2024)

Ambos os grupos participaram da mesma intervenção didática, que consistiu na resolução de problemas de programação em C++, abordando conteúdos como variáveis, estruturas de decisão, estruturas de repetição, funções e *arrays*. Os enunciados das questões foram elaborados com base nos princípios da aprendizagem baseada em problemas (ABP), nos quais os alunos deveriam interpretar um problema de contexto real e propor soluções práticas utilizando conceitos de programação. Em cada problema, os alunos tiveram que identificar requisitos e restrições, selecionar abordagens adequadas e implementar um código funcional.

Por se tratar de uma intervenção pontual, não foram aplicados pré-teste e pós-teste. O foco concentrou-se na observação dos processos de interação, nas percepções dos estudantes e nos padrões de uso das ferramentas de IA.

No Grupo Aluno+IA, o estudante assumiu o papel de piloto, sendo responsável por analisar o problema, tomar decisões de implementação e escrever o código, enquanto o

ChatGPT atuou como navegador, oferecendo sugestões, esclarecimentos conceituais e orientações durante o processo de resolução das tarefas. Essa configuração buscou replicar, na medida do possível, a dinâmica de reflexão e diálogo característica da programação em pares tradicionais.

Durante a execução das atividades, os alunos foram orientados a utilizar o ChatGPT como um par de programação, ou seja, dialogando com a IA, pedindo sugestões, discutindo estratégias e buscando apoio conceitual, sempre mantendo a responsabilidade pela tomada de decisão e escrita do código. A interação com a IA ocorreu de forma livre, sem restrições quanto ao número de perguntas ou ao tipo de solicitação, com orientação de que não fossem solicitadas soluções completas de código.

No Grupo Aluno+Aluno, os estudantes trabalharam em duplas, utilizando um único computador, com alternância entre os papéis de piloto e navegador, conforme os princípios clássicos da programação em pares, promovendo o revezamento na implementação do código e a discussão conjunta das soluções.

Após a realização da atividade, ambos os grupos responderam a um questionário aplicado via Google Formulários, composto por dois blocos: (i) caracterização do perfil dos participantes e (ii) percepções de motivação, avaliadas por meio de um questionário baseado no IMI, adaptado ao contexto da atividade. As respostas foram mensuradas por meio de uma escala tipo *Likert* de 5 pontos, variando de “Discordo totalmente” a “Concordo totalmente”. As respostas ao questionário foram analisadas por meio de estatísticas descritivas, buscando identificar tendências e diferenças na motivação entre os dois grupos.

No grupo Aluno+IA também foram coletados os registros das interações com o ChatGPT, incluindo as perguntas dos alunos e as respostas geradas pela IA. Esses dados foram submetidos à análise de conteúdo segundo Bardin (2016), contemplando as etapas de pré-análise, categorização e interpretação, considerando as características das solicitações feitas pelos alunos e os padrões de interação estabelecidos com a IA.

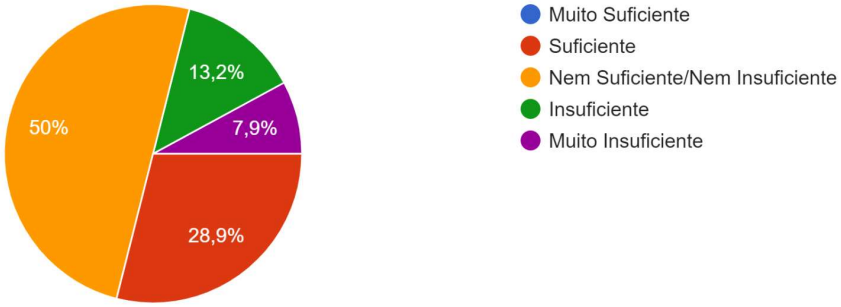
6.1.1 Perfil dos Participantes

No início da intervenção, ambos os grupos responderam a um questionário inicial com objetivo de caracterizar o perfil dos estudantes. Nesse instrumento, foram coletadas informações gerais sobre idade, gênero, semestre e conhecimentos prévios em programação, permitindo uma visão geral da familiaridade dos alunos com o conteúdo abordado.

A média de idade dos participantes foi de 21 anos. Em relação ao gênero, 81% se declararam do gênero masculino e 19% do gênero feminino. A maioria encontrava-se no segundo semestre do curso (n=32; 86,5%), enquanto os demais pertenciam a semestres mais avançados, em decorrência de reprovações anteriores na disciplina (2 alunos do oitavo semestre e 1 aluno de cada um dos seguintes semestres: quinto, sexto e sétimo).

Quanto à autoavaliação do nível de conhecimento em programação, 50% classificaram seu conhecimento como “nem suficiente, nem insuficiente”, conforme apresentado na Figura 42, indicando nível intermediário de confiança no momento da intervenção.

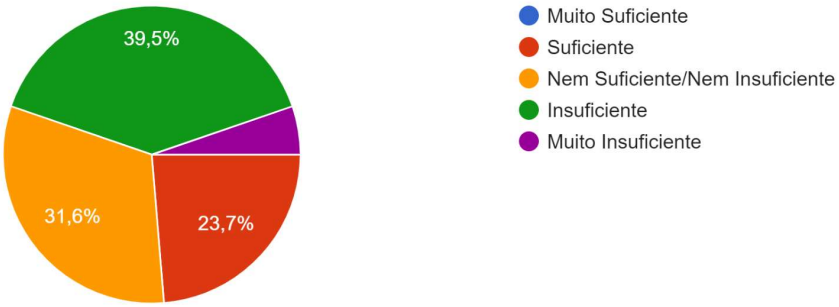
Figura 42 - Estudo preliminar: nível de conhecimento em programação



Fonte: Elaborado pela autora (2024)

Em relação às linguagens de programação conhecidas e já utilizadas pelos participantes, destacam-se Python (92%) e C++ (95%), seguidas por Java (13%), JavaScript (13%), PHP (10%) e Delphi (5%). Embora a maioria tenha indicado já ter tido contato prévio com a linguagem C++, 39,5% declararam possuir conhecimento insuficiente nessa linguagem, conforme ilustrado na Figura 43, o que evidencia o caráter introdutório do conteúdo trabalhado na disciplina.

Figura 43 - Estudo preliminar: nível de conhecimento em C++



Fonte: Elaborado pela autora (2024)

De forma geral, o perfil dos participantes indica um grupo de estudantes em fase inicial de formação em programação, com experiência prévia limitada e níveis variados de confiança em relação aos conteúdos trabalhados. Embora a maioria dos alunos tenha tido contato prévio com a linguagem C++, uma parcela significativa declarou possuir conhecimento insuficiente nessa linguagem, reforçando o caráter introdutório da disciplina e a necessidade de estratégias de apoio ao processo de aprendizagem. Esse contexto torna o cenário do estudo adequado para a investigação do uso de ferramentas de IA generativa como par de programação, uma vez que os estudantes ainda estão desenvolvendo suas habilidades de resolução de problemas e programação.

6.1.2 Padrões de uso da IA como Par de Programação (Grupo Aluno+IA)

Considerando o caráter exploratório do estudo preliminar, a análise das interações buscou compreender como os alunos utilizaram o ChatGPT na resolução de problemas de programação em C++. O foco esteve na identificação dos tipos de solicitações e padrões de interação entre estudantes e a IA, caracterizando diferentes modos de uso do ChatGPT como par de programação. Para isso, foram analisadas as sessões de interação de 13 alunos que utilizaram a ferramenta durante a atividade.

Os participantes não possuíam experiência prévia formal com o uso do ChatGPT em atividades de programação em sala de aula, tendo recebido apenas uma breve orientação inicial sobre seu uso como par de programação. Essa condição é relevante, pois as interações observadas refletem comportamentos iniciais de apropriação da ferramenta, variando entre o uso exploratório e as solicitações diretas de solução.

O número de interações com o ChatGPT variou consideravelmente entre os participantes, com um mínimo de 2 e um máximo de 31 interações por exercício, com uma média geral de 13,1 interações por aluno por exercício. A maioria utilizou a IA em todos os exercícios propostos, embora com níveis distintos de engajamento e intencionalidade. Enquanto alguns recorreram ao ChatGPT predominantemente para obter respostas prontas, outros o utilizaram como apoio ao processo de resolução, solicitando explicações, revisões e orientações graduais.

Com base na análise de conteúdo das conversas, realizada segundo os procedimentos propostos por Bardin (2016), as mensagens foram classificadas em seis categorias principais, apresentadas na Tabela 3. Essas categorias refletem diferentes estratégias adotadas pelos

alunos ao utilizar a IA durante a atividade.

Tabela 3 - Estudo preliminar: Categorias de interação com o ChatGPT

Categoria	**Descrição**	**Frequência (%)**
Solicitação de solução pronta	O aluno apresenta o enunciado do problema e solicita a implementação direta do código.	37,65
Diálogo contextualizado e metacomunicação	Mensagens de natureza organizacional, afetiva ou reflexiva, indicando planejamento, transição entre tarefas ou agradecimento.	22,35
Solicitação de explicação de código ou conceitos	O aluno busca compreender o funcionamento de um trecho de código, função, estrutura ou conceito da linguagem.	13,53
Envio de código próprio para análise ou correção	O aluno compartilha um código de sua autoria, solicitando depuração, análise ou validação.	9,41
Solicitação de dicas e colaboração passo a passo	O aluno solicita orientações graduais, discutindo etapas ou decisões do código em conjunto com a IA.	8,82
Solicita ajustes no código gerado pelo *ChatGPT*	O aluno pede alterações na saída anterior, como remoção de comentários, ajustes simples ou refatoração.	8,24

Fonte: Elaborado pela autora (2024)

A análise das categorias permitiu identificar dois padrões principais de comportamento no uso da IA: (i) uso reprodutivo e dependente e (ii) uso ativo e colaborativo.

O uso reprodutivo e dependente caracteriza-se pela predominância de solicitações de solução pronta (37,65%) e pedidos ajustes no código gerado pela IA (8,24%). Nesse padrão, o aluno tende a reproduzir o enunciado do exercício e solicitar que o ChatGPT gere a solução completa, como nos exemplos: *"Faça um programa em C++ que registre as presenças e faltas dos alunos e calcule o percentual de presença…"* (Aluno 04, Exercício 1); *"Preciso fazer o seguinte programa em c++: <Escreveu o enunciado do problema>"* (Aluno 10, Exercício 2); "*Tire os comentários do código <código gerado pelo ChatGPT>*" (Aluno 04, Exercício 4). Nesses casos, a IA atuou como fornecedora de respostas, enquanto o aluno limitou-se a inserir o enunciado e aguardar o código pronto. Esse tipo de uso indica dependência instrumental e baixo envolvimento cognitivo, em que a IA é utilizada como substituta do esforço de raciocínio lógico.

O uso ativo e colaborativo, foi observado nas interações em que os alunos buscaram compreender, revisar ou construir o código em conjunto com a IA. Esse padrão inclui solicitações de explicações conceituais, depuração de código próprio e pedidos de orientação

passo a passo. Exemplos representativos são: *"Quero entender a lógica do código, explique para um leigo que não entende nada sobre isso, e também ensine a própria linguagem C++."* (Aluno 07, Exercício 1); *"Como funciona a função rand no c++?"* (Aluno 12, Exercício 2). Nesses casos, o ChatGPT assumiu o papel de mediador e orientador, favorecendo a reflexão sobre o processo lógico e a estrutura da linguagem. O aluno engajou-se de forma ativa, explorando alternativas, avaliando decisões e consolidando o aprendizado por meio do diálogo com a IA.

A categoria Diálogo contextualizado e metacomunicação (22,35%) indica ainda que parte dos alunos manteve interações mais próximas de uma comunicação humana, utilizando a IA para organizar o trabalho, comentar o andamento da atividade ou expressar agradecimento. Como exemplo pode-se citar: *"Olha que legal esse, estou pensando em fazer em vector. O que acha e como faria esse programa (sem código), apenas raciocínio?"* (Aluno 05, Exercício 4); *"Olá chat boa noite, teremos muito trabalho de programação em C++ hoje, gostaria de ter você como meu copiloto"* (Aluno 13, Exercício 1). Esse comportamento sugere uma tentativa de personalizar a interação, aproximando-a da dinâmica típica de um par de programação humano.

De modo geral, os resultados evidenciam que o uso do ChatGPT como par de programação pode variar de um uso mais passivo e reprodutivo, centrado na execução imediata das tarefas, e um uso ativo e formativo, voltado à compreensão, reflexão e desenvolvimento da autonomia. As interações que envolveram explicações, revisões e orientações graduais apresentaram maior potencial pedagógico, enquanto as solicitações de código pronto tendem a limitar o caráter formativo da experiência.

Esses achados corroboram estudos anteriores (FAN *et al.*, 2025; RAHMAN; WATANOBE, 2023; PRATHER *et al.*, 2024), que apontam a coexistência de usos dependentes e colaborativos em contextos de programação com IA. No âmbito deste estudo preliminar, os resultados reforçam a necessidade de estratégias pedagógicas e de design de sistemas que orientem o uso da IA para fins educacionais, estimulando a reflexão e a autonomia. Essas evidências fundamentaram decisões de projeto do assistente de programação desenvolvido nesta tese, especialmente no que se refere à definição de papéis inspirados na programação em pares, à adoção de *guardrails* e o uso de abordagens baseadas na mediação pedagógica e no método socrático.

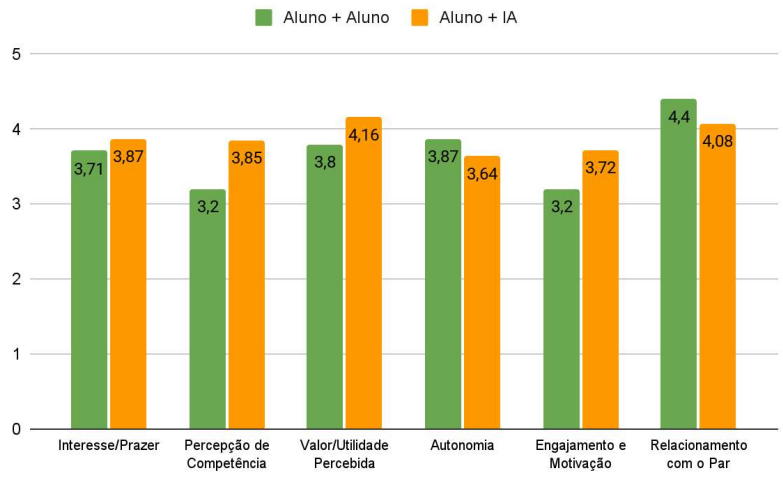
6.1.3 Questionário de Motivação

Após a realização da intervenção didática, os estudantes de ambos os grupos responderam a um questionário de motivação inspirado no IMI, adaptado ao contexto da programação em pares (Apêndice V). Diferentemente da versão original do instrumento, o questionário utilizado neste estudo preliminar foi reorganizado e ampliado para contemplar dimensões consideradas pertinentes ao contexto investigado, incluindo Interesse/Prazer, Competência Percebida, Valor/Utilidade, Autonomia, Motivação/Continuidade e Relacionamento com o Par. As respostas, em escala *Likert* de 1 a 5, foram analisadas por estatísticas descritivas e contrastadas entre os grupos (Aluno+Aluno e Aluno+IA), com o objetivo de identificar tendências motivacionais associadas a cada forma de interação.

Considerando que, no estudo preliminar, os resultados originais estavam disponíveis apenas na forma de distribuições percentuais de frequência (dados agregados), e não em registros individuais por participante, optou-se pelo uso da média ponderada como medida de tendência central. Esse procedimento permitiu sintetizar a percepção geral de cada grupo em um único valor numérico representativo para cada item e para cada dimensão da escala *Likert*.

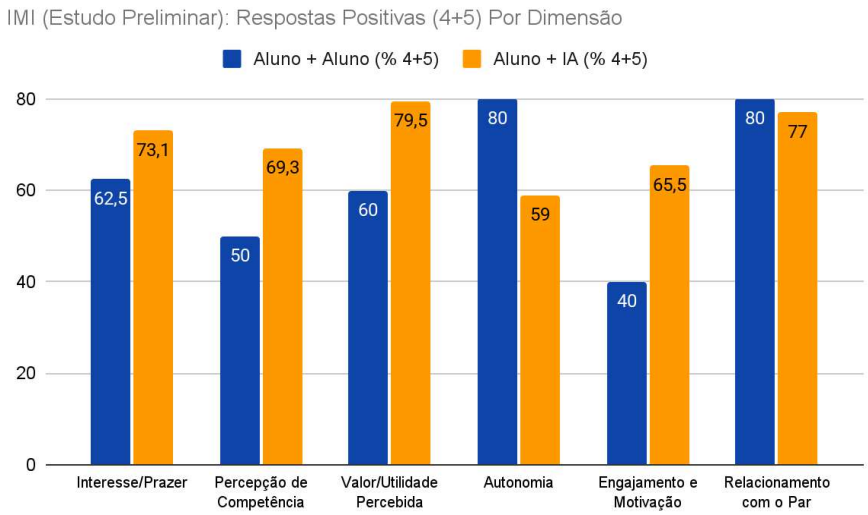
Além da média ponderada, os resultados também são apresentados em termos de percentual de respostas positivas, considerando exclusivamente as categorias de concordância (4 e 5), sendo a categoria 3 interpretada como neutra. Essa medida complementa a análise ao destacar o grau de aceitação ou concordância dos participantes em relação a cada dimensão. A Figura 44 apresenta a média geral por dimensão em cada grupo, enquanto a Figura 45 sintetiza o percentual médio de respostas positivas (4+5) por dimensão.

Figura 44 - Estudo preliminar: Média geral motivação por dimensão



Fonte: Elaborado pela autora (2024).

Figura 45 - Estudo preliminar: Percentual médio de respostas positivas por dimensão



Fonte: Elaborado pela autora (2024).

Nas subseções seguintes, os resultados são detalhados por dimensão com a discussão dos principais padrões observados. Ressalta-se que, por se tratar de dados agregados e de uma análise descritiva e exploratória, esses resultados devem ser interpretados como indicativos iniciais, servindo de base para os estudos subsequentes.

6.1.3.1 Dimensão: Interesse/Prazer

No grupo Aluno+Aluno, a dimensão Interesse/Prazer apresentou tendência favorável, embora com intensidade moderada (média geral = 3,71; percentual de respostas positivas = 62,5%). De modo geral, os itens dessa dimensão indicam que a experiência de programação em pares com outro estudante foi percebida como interessante e envolvente, ainda que com certa variabilidade nas respostas, especialmente em relação à satisfação com a dinâmica colaborativa.

No grupo Aluno+IA, a avaliação da dimensão foi ligeiramente superior (média geral = 3,87; percentual médio de respostas positivas = 73,1%). Os resultados sugerem que a experiência com a IA foi, em geral, percebida como interessante e satisfatória, com destaque para os itens relacionados ao prazer em trabalhar com a ferramenta e à percepção da atividade como interessante. Por outro lado, o item referente ao envolvimento durante a atividade apresentou maior dispersão, indicando que a experiência com a IA não foi percebida de maneira homogênea por todos os estudantes.

De forma geral, observa-se tendência favorável ao grupo Aluno+IA na dimensão

Interesse/Prazer, tanto em termos de média quanto no percentual de respostas positivas. Comparativamente, o grupo Aluno+IA apresentou maior concentração de respostas nos níveis superiores da escala, superando o grupo Aluno+Aluno nos aspectos relacionados ao prazer, à percepção de interesse e à satisfação com a atividade. Em contrapartida, o item referente ao envolvimento durante a colaboração apresentou resultado mais favorável no grupo Aluno+Aluno, sugerindo que a interação com um colega pode favorecer mais diretamente a percepção de engajamento contínuo durante a atividade.

6.1.3.2 Dimensão: Competência Percebida

No grupo Aluno+Aluno, a dimensão Competência Percebida apresentou avaliação moderada, com maior dispersão nas respostas (média geral = 3,20; percentual médio de respostas positivas = 50,0%). De modo geral, os resultados indicam que a colaboração com outro estudante contribuiu de forma parcial para a percepção de aprendizagem, com menor consistência nos itens relacionados à compreensão conceitual e à melhoria na resolução de problemas. Por outro lado, os itens associados à confiança apresentaram avaliações mais favoráveis, sugerindo que, embora os ganhos cognitivos não tenham sido percebidos de forma homogênea, a interação com o colega contribuiu para aumentar a segurança dos estudantes ao programar.

No grupo Aluno+IA, a percepção de competência foi mais elevada e concentrada nos níveis superiores da escala (média geral da dimensão = 3,85; % médio de respostas positivas = 69,3%). Os resultados indicam que a maioria dos estudantes percebeu a contribuição da IA tanto para a compreensão dos conceitos quanto para o desenvolvimento da capacidade de resolução de problemas, ainda que com alguma variabilidade nas respostas. De modo geral, os itens relacionados ao apoio conceitual e à confiança apresentaram avaliações predominantemente positivas, evidenciando uma percepção mais consistente de progresso individual.

Em síntese, observa-se tendência favorável ao grupo Aluno+IA na dimensão Competência Percebida, tanto em termos de média geral (3,85 vs. 3,20) quanto no percentual de respostas positivas (69,3% vs. 50,0%). Esses resultados sugerem que o uso da IA como par de programação esteve associado a uma maior percepção de apoio ao aprendizado, à compreensão conceitual e à confiança na resolução de problemas, possivelmente em função do *feedback* imediato e estruturado fornecido pela ferramenta.

6.1.3.3 Dimensão: Valor/Utilidade Percebida

No grupo Aluno+Aluno, a dimensão Valor/Utilidade Percebida apresentou tendência favorável, ainda que com certa heterogeneidade nas respostas (média geral = 3,80; percentual de respostas positivas = 60,0%). De modo geral, os estudantes reconheceram a utilidade da programação em pares para o aprendizado imediato, embora com menor consenso quanto ao seu impacto no desempenho futuro. Enquanto os itens relacionados à utilidade da atividade foram avaliados de forma mais positiva, aqueles associados à aplicabilidade futura apresentaram maior variabilidade, indicando que nem todos os estudantes percebem a estratégia como um recurso consistente para o desenvolvimento a longo prazo.

No grupo Aluno+IA, a percepção de valor foi mais acentuada (média geral = 4,16; % médio de respostas positivas = 79,5%). Os resultados indicam que a maioria dos estudantes reconheceu a utilidade da IA tanto para o aprendizado imediato quanto para o desempenho futuro em programação, com avaliações predominantemente positivas também em relação à sua relevância como ferramenta de apoio aos estudos. De modo geral, observa-se maior consistência nas respostas, sugerindo uma percepção mais homogênea de valor atribuído à experiência com a IA.

De forma geral, os resultados indicam tendência mais favorável ao grupo Aluno+IA na dimensão Valor/Utilidade Percebida, tanto na média global (4,16 vs. 3,80) quanto na proporção de respostas positivas (79,5% vs. 60,0%). Essa diferença foi mais evidente nos aspectos relacionados à expectativa de impacto futuro e à relevância da estratégia para os estudos, sugerindo que a IA foi percebida como um recurso pedagógico útil e potencialmente aplicável ao aprendizado de programação.

6.1.3.4 Dimensão: Autonomia

No grupo Aluno+Aluno, a dimensão Autonomia apresentou avaliação elevada (média geral = 3,87; percentual de respostas positivas = 80,0%), indicando que os estudantes perceberam alto grau de controle e liberdade durante a realização das atividades. De modo geral, os resultados sugerem que a programação em pares favoreceu a tomada de decisões e a participação ativa no processo de resolução dos problemas, com destaque para a percepção de liberdade para contribuir com ideias, que foi amplamente reconhecida entre os participantes.

No grupo Aluno+IA, as respostas foram mais distribuídas e revelaram maior

heterogeneidade (média geral = 3,64; percentual de respostas positivas = 59,0%). Embora os estudantes tenham indicado percepção relativamente positiva quanto à possibilidade de tomar decisões e utilizar a ferramenta de forma flexível, o item relacionado ao controle do processo de aprendizagem apresentou avaliações mais baixas e dispersas. Esse resultado sugere que parte dos estudantes não se sentiu plenamente no controle ao utilizar a IA como par de programação.

Em síntese, os resultados indicam que a dimensão Autonomia foi percebida de modo mais favorável no grupo Aluno+Aluno (3,87 vs. 3,64; 80,0% vs. 59,0%), especialmente no que se refere ao controle global da atividade. Esse achado sugere que, para parte dos estudantes, o uso da IA como par pode estar associado a uma menor sensação de controle do processo de aprendizagem, reforçando a importância de estratégias de uso orientado e de mecanismos de mediação (como *guardrails* e condução por etapas) para que a IA atue como apoio sem comprometer a autonomia do estudante.

6.1.3.5 Dimensão: Engajamento/Motivação para Continuidade

No grupo Aluno+Aluno, a dimensão Engajamento/Motivação para Continuidade apresentou avaliação moderada e heterogênea (média geral = 3,20; percentual de respostas positivas = 40,0%), indicando variação nas percepções quanto à continuidade da experiência. De modo geral, os estudantes demonstraram motivação para continuar aprendendo durante a atividade, porém com menor consistência quando se considera a intenção de manter a prática de programação em pares ao longo do tempo, especialmente fora do contexto das aulas.

No grupo Aluno+IA, os resultados indicaram maior engajamento e intenção de continuidade (média geral = 3,72; percentual de respostas positivas = 65,5%). Os estudantes relataram níveis mais elevados de motivação para continuar aprendendo com o uso da IA, além de maior predisposição para manter essa forma de interação como parte do processo de aprendizagem. Observou-se também que parte significativa dos participantes demonstrou interesse em utilizar a IA como apoio mesmo fora do ambiente formal de ensino, embora com alguma variabilidade nas respostas.

De forma geral, os dados indicam diferença entre os grupos na dimensão Engajamento/Motivação para Continuidade, com valores superiores no grupo Aluno+IA (3,72 vs. 3,20; 65,5% vs. 40,0%). Essa diferença foi mais evidente nos aspectos relacionados à intenção de continuidade da prática, sugerindo que o uso da IA como par esteve associado a

maior predisposição para dar seguimento às atividades de aprendizagem, possivelmente em função da disponibilidade imediata de suporte e da percepção de progresso individual durante a atividade.

6.1.3.6 Dimensão: Relacionamento com o Par

Na dimensão Relacionamento com o par (humano ou IA), observam-se avaliações positivas em ambos os grupos, indicando que os estudantes, de modo geral, perceberam a parceria como adequada e confiável durante a resolução dos problemas (Aluno+Aluno: média geral = 4,40; percentual de respostas positivas = 80,0%; Aluno+IA: média geral = 4,12; percentual de respostas positivas = 77,0%).

No grupo Aluno+Aluno, a percepção do colega como parceiro foi bastante elevada, com avaliações consistentemente positivas tanto em relação à qualidade da parceria quanto à confiança nas orientações recebidas. De modo geral, os resultados indicam que a interação entre os estudantes foi percebida como colaborativa e confiável, favorecendo a dinâmica de resolução conjunta dos problemas.

No grupo Aluno+IA, as avaliações também foram positivas, porém com maior variabilidade na percepção da IA enquanto parceira. Ainda que nem todos os estudantes a tenham percebido como um par ideal, a confiança nas orientações fornecidas foi elevada, indicando que a IA foi amplamente reconhecida como uma fonte de apoio confiável durante a atividade.

De modo geral, os resultados indicam que o relacionamento com o par foi avaliado de forma favorável nos dois grupos. Enquanto o grupo Aluno+Aluno apresentou avaliações ligeiramente superiores na percepção de parceria, o grupo Aluno+IA apresentou alta confiança nas orientações fornecidas, evidenciando que a IA foi percebida como uma fonte de apoio confiável no processo de resolução dos exercícios.

6.1.3.7 Síntese geral dos resultados do Questionário de Motivação

De modo geral, os resultados descritivos do questionário de motivação indicam percepções positivas em ambos os grupos quanto à experiência de programação em pares, seja com colega ou com IA. Ao comparar as médias gerais por dimensão, observa-se tendência favorável ao grupo Aluno+IA em aspectos relacionados à motivação e ao valor percebido da atividade, especialmente em Interesse/Prazer, Percepção de Competência, Valor/Utilidade

Percebida e Engajamento/Motivação para Continuidade. Esses resultados sugerem que, no contexto desta intervenção pontual, os estudantes que utilizaram o *ChatGPT* perceberam maior apoio à aprendizagem, maior relevância da atividade e maior intenção de continuidade.

Entretanto, ao combinar esses achados com os padrões de interação descritos na Seção 6.1.2, observa-se que tais percepções não foram homogêneas e dependem do modo de uso da IA. As interações classificadas como uso reprodutivo e dependente, nas quais o aluno solicitava solução pronta e realizava apenas ajustes superficiais, podem explicar parte da variabilidade observada em itens como envolvimento e controle do processo, uma vez que esse tipo de uso tende a reduzir o protagonismo do estudante na resolução do problema. Por outro lado, as interações associadas ao uso ativo e colaborativo, nas quais os alunos solicitaram explicações, revisões, depuração de código próprio e orientações passo a passo, são coerentes com as dimensões em que o grupo Aluno+IA apresentou avaliações mais altas, especialmente em competência percebida e utilidade, pois indicam maior envolvimento cognitivo.

Essa relação se torna mais evidente na dimensão Autonomia, que apresentou avaliação mais favorável no grupo Aluno+Aluno em comparação ao grupo Aluno+IA, especialmente no que se refere ao controle do processo de aprendizagem. Esse resultado sugere que, para parte dos estudantes, o uso da IA pode estar associado a menor sensação de controle, sobretudo quando utilizada como fonte de respostas prontas. Assim, embora a IA seja percebida como um apoio relevante, seu potencial depende de estratégias de uso que preservem a tomada de decisão e incentivem o raciocínio do estudante.

Por fim, a dimensão Relacionamento com o par apresentou avaliações elevadas em ambos os grupos, indicando que tanto o parceiro humano quanto a IA foram percebidos como suporte confiável durante a resolução dos exercícios, com destaque para a confiança nas orientações recebidas.

Considerando o caráter exploratório do estudo preliminar e a natureza descritiva da análise, os resultados não permitem generalizações, mas oferecem indícios relevantes sobre potencialidades e limites do uso de IA como par de programação. Em especial, sugerem que os benefícios motivacionais e pedagógicos da IA estão associados a formas de uso que favoreçam interações colaborativas e mediadas, em vez de abordagem centradas na simples obtenção de soluções prontas.

6.1.4 Percepções Qualitativas dos Estudantes

Com o objetivo de identificar desafios e diretrizes para a implementação da IA como par de programação, foi incluída uma questão aberta no questionário final, permitindo que os estudantes registrassem percepções, limitações e sugestões relacionadas ao uso educacional dessas ferramentas. O Quadro 3 apresenta as respostas dos estudantes à questão *"Quais melhorias você gostaria de ver nas ferramentas de IA generativa para uso educacional?"*, evidenciando diferentes expectativas sobre o papel da IA no apoio à aprendizagem.

Quadro 3 - Estudo preliminar: Opiniões dos alunos sobre o uso de IA

Quais melhorias você gostaria de ver nas ferramentas de IA generativa para uso educacional?
"Acredito que se torna muito fácil em conseguir uma resposta, acabamos ficando preguiçosos com isso"
"Talvez uma ferramenta específica para programadores, uma IA que entenda o modus operandis e consiga uma precisão e um entendimento maior."
"algo mais voltados a programadores que ajudasse no desenvolvimento de códigos que possam facilitar o dia a dia do programador"
"Ter uma matéria só para ensinar a utilizar corretamente as IA's como ferramentas para a vida"
"Ia deve ser usada somente em último caso no ensino, talvez uma ia que tenha acesso ao conteúdo disponibilizado ao aluno e faça somente uso deste conteúdo ao fornecer suas respostas."
"Não entregarem um código pronto logo de cara, mas sim ir instigando o conhecimento e dando pistas de como fazer/por onde ir para alcançar o resultado esperado."
"Proibido dar a resposta completa, a IA deve clarear o caminho para que o raciocínio do aluno não fique estagnado."
"Que fosse avaliando meu código em tempo real e dando ideias e melhorias para o mesmo(enquanto vou escrevendo)"
"Talvez que seja mais indicativa, para o uso educacional ela deve mais indicar como realizar do que fazer em si"
"Não mostrar a resposta e sim ensinar como montar a resposta"

Fonte: Elaborado pela autora (2024)

A análise das respostas indica a recorrência de três pontos centrais. O primeiro refere-se ao risco de dependência quando a IA fornece soluções completas de forma imediata, o que, segundo os estudantes, pode reduzir o esforço cognitivo e favorecer o uso passivo da

ferramenta (*"acabamos ficando preguiçosos"*, *"proibido dar a resposta completa"*). O segundo ponto diz respeito à expectativa de uma IA mais orientadora e formativa, que *"instigue o raciocínio"*, ofereça *"pistas"* e ajude o aluno a *"montar a resposta"*, em vez de simplesmente apresentar o código pronto. O terceiro envolve demandas por funcionalidades e personalização, como uma ferramenta mais *"voltada a programadores"*, com maior precisão no contexto de programação, além da possibilidade de avaliação do código em tempo real e sugestões durante o processo de escrita.

Também aparecem sugestões ligadas ao uso pedagógico orientado da IA, como a necessidade de ensinar os estudantes a utilizar essas ferramentas de forma adequada no contexto educacional. Neste sentido, alguns participantes sugeriram que a IA pudesse responder com base apenas nos conteúdos disponibilizados na disciplina, favorecendo maior alinhamento com os materiais didáticos.

Esses achados qualitativos reforçam os resultados observados nas interações analisadas na Seção 6.1.2, nas quais se identificou a coexistência de dois padrões de uso da IA: um mais dependente ou reprodutivo, centrado na obtenção de respostas prontas, e outro mais ativo e colaborativo, voltado à compreensão, revisão e construção gradual de soluções. De modo geral, os estudantes reconhecem benefícios no uso dessas ferramentas, mas também destacam a importância de mecanismos que incentivem o raciocínio e evitem o uso passivo da IA.

Essas percepções mostram-se alinhadas às diretrizes pedagógicas adotadas no desenvolvimento do assistente proposto nesta tese. Em particular, reforçam a relevância de estratégias como a implementação de *guardrails* para reduzir a geração direta de soluções completas, o uso de engenharia de *prompt* para promover orientação gradual, a adoção de abordagens inspiradas na ZDP e no método socrático, incentivando a resolução de problemas por etapas e o pensamento crítico do estudante e a utilização dos materiais da disciplina como base de conhecimento do assistente (RAG).

Considerando o caráter exploratório do estudo preliminar, essas percepções dos estudantes também forneceram subsídios importantes para o refinamento do AIPairCode, indicando que o potencial educacional da IA tende a ser ampliado quando seu uso é acompanhado de orientações pedagógicas e quando a interação é desenhada para apoiar a construção do conhecimento, e não para substituir o processo de resolução de problemas.

6.2 Estudo Final 1 - Aplicação do AIPairCode em uma Disciplina de JavaScript

O Estudo Final 1 foi conduzido no primeiro semestre letivo de 2025 na disciplina Tópicos Especiais em Programação: Introdução à Programação Web com JavaScript, do curso de Sistemas de Informação da Universidade Federal de Santa Maria (UFSM), campus de Frederico Westphalen. Nessa etapa, o objetivo foi avaliar de forma sistemática as percepções dos estudantes, os padrões de interação durante as atividades e o impacto pedagógico do uso de um assistente de programação personalizado, concebido para atuar como par de programação, em um contexto real de ensino e aprendizagem.

O estudo envolveu duas turmas distintas, organizadas como grupo de controle e grupo experimental, possibilitando a comparação entre o modelo tradicional de ensino e uma abordagem apoiada por uma ferramenta de IA.

No grupo experimental, os estudantes utilizaram o *AIPairCode*, no módulo de extensão do *Visual Studio Code*, durante as aulas em laboratório. O assistente foi utilizado como par de programação ao longo da resolução dos exercícios e atividades propostas, oferecendo suporte contínuo ao processo de desenvolvimento, depuração e correção de código. No grupo de controle, as atividades foram realizadas em laboratório segundo a metodologia tradicional da disciplina, sem o uso do assistente, contando exclusivamente com o apoio da professora e dos materiais didáticos.

Em ambas as turmas, foram trabalhados os mesmos conteúdos programáticos, os mesmos conjuntos de exercícios e a mesma carga horária de atividades práticas, de modo a assegurar condições equivalentes de ensino. Assim, a principal variável de diferenciação entre os grupos foi a presença do assistente de programação no grupo experimental. Durante o estudo, foram coletados dados quantitativos e qualitativos, incluindo resultados em testes diagnósticos, registros de interação com a ferramenta e questionários de percepção, motivação e usabilidade. Essa configuração metodológica permitiu investigar o potencial do assistente como apoio ao desenvolvimento do raciocínio crítico e à resolução de problemas em programação no contexto do ensino de JavaScript.

O estudo foi conduzido entre os meses de março e junho de 2025, totalizando 15 semanas letivas e 60 horas/aula, correspondentes à duração integral da disciplina. A primeira turma contou com 24 estudantes matriculados, dos quais 11 desistiram da disciplina por reprovação por falta, resultando em 13 participantes efetivos. A segunda turma teve 18

estudantes matriculados, com 4 desistências, totalizando 14 participantes efetivos.

Dessa forma, a primeira turma foi designada como grupo de controle, enquanto a segunda foi alocada como grupo experimental. A definição dos grupos ocorreu por conveniência, em função da organização prévia das turmas na disciplina e não por meio de randomização, caracterizando o delineamento da pesquisa como quase-experimental. A distribuição dos grupos é apresentada na Tabela 4.

Tabela 4 - Resumo da participação dos estudantes no Estudo Final 1

Grupos	Quantidade de alunos
Grupo Controle	13
Grupo Experimental	14
Total	**27**

Fonte: Elaborado pela autora (2025)

A adoção de grupos por conveniência impõe limitações metodológicas, pois não elimina completamente a possibilidade de vieses decorrentes de diferenças preexistentes entre as turmas. Embora o pré-teste aplicado tenha permitido mapear os conhecimentos iniciais dos estudantes e apoiar a comparação entre os grupos, a ausência de randomização restringe inferências causais mais robustas, exigindo cautela na atribuição de eventuais diferenças de desempenho exclusivamente à intervenção com a ferramenta de IA.

6.2.1 Caracterização das Turmas e Perfil dos Participantes

No início do estudo, os estudantes das duas turmas realizaram um pré-teste com o objetivo de mapear os conhecimentos iniciais relacionados à programação web com JavaScript. Além disso, foram considerados dados demográficos e acadêmicos básicos, como gênero e semestre cursado, permitindo a caracterização do perfil dos participantes e a contextualização das análises subsequentes.

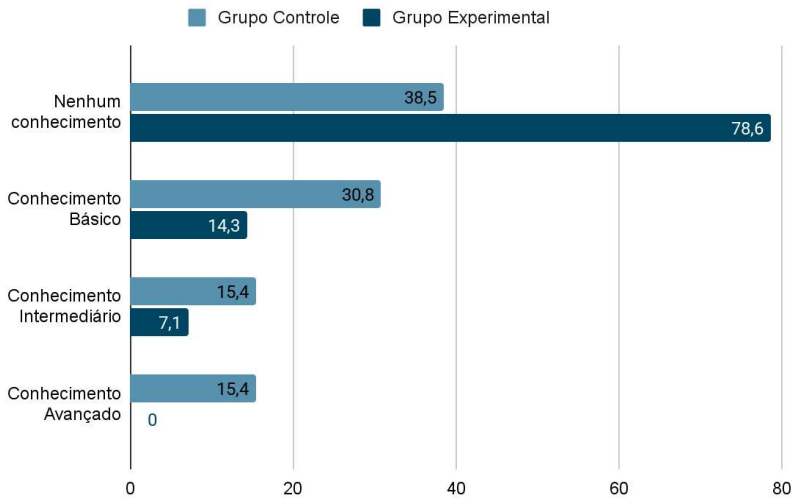
O grupo de controle foi composto por 13 estudantes, enquanto o grupo experimental contou com 14 participantes. Em relação ao gênero, observou-se uma distribuição semelhante entre os grupos, com predominância do sexo masculino. No grupo de controle, 11 estudantes se declararam do sexo masculino e 2 do sexo feminino, enquanto no grupo experimental 12 estudantes se declararam do sexo masculino e 2 do sexo feminino.

Quanto ao semestre cursado, identificou-se uma diferença no estágio de progressão acadêmica entre as turmas. O grupo de controle concentrou-se majoritariamente nos semestres mais avançados do curso, especialmente no 7º e 9º semestres. Já o grupo experimental apresentou uma distribuição mais heterogênea, com maior concentração de estudantes nos semestres iniciais e intermediários (1º, 3º e 5º semestres), além de um estudante do 9º semestre. Essa diferença no perfil acadêmico reforça a importância da utilização de instrumentos diagnósticos para a análise dos conhecimentos prévios dos participantes.

No que se refere ao desempenho inicial, avaliado por meio do pré-teste (escala de 0 a 10), observou-se variabilidade em ambos os grupos. No grupo de controle, as notas variaram de 0 a 7,5, indicando a presença tanto de estudantes com conhecimentos prévios mais limitados quanto de alunos com maior domínio inicial dos conteúdos. No grupo experimental, as notas do pré-teste variaram de 1 a 6, com maior concentração nos níveis mais baixos e intermediários da escala. Esses resultados evidenciam diferenças iniciais entre as turmas, reforçando a necessidade de análises comparativas que considerem o desempenho prévio dos estudantes, conforme apresentado nas seções seguintes.

Além do pré-teste, os estudantes responderam, na primeira aula da disciplina, a uma questão de autoavaliação sobre o grau de conhecimento prévio em JavaScript. Essa informação permitiu complementar a caracterização do perfil inicial das turmas a partir da percepção dos próprios participantes, conforme ilustrado na Figura 46.

Figura 46 - Estudo Final 1: autoavaliação do conhecimento prévio em JavaScript



Fonte: Elaborado pela autora (2025)

No grupo experimental, observou-se que a maioria dos estudantes declarou não possuir

conhecimento prévio em JavaScript, correspondendo a 78,6% dos participantes. Apenas 14,3% indicaram possuir conhecimento básico, 7,1% relataram conhecimento intermediário e nenhum estudante se autodeclarou com conhecimento avançado na tecnologia. Esses dados indicam que o grupo experimental era predominantemente composto por alunos iniciantes no conteúdo abordado.

No grupo de controle, a distribuição das respostas revelou um perfil mais heterogêneo. Embora 38,5% dos estudantes tenham declarado não possuir conhecimento prévio em JavaScript, uma parcela relevante indicou possuir conhecimento básico (30,8%), intermediário (15,4%) ou avançado (15,4%). Esse resultado sugere maior familiaridade prévia com a tecnologia entre os estudantes do grupo de controle, quando comparado ao grupo experimental.

A autoavaliação do conhecimento em JavaScript, acompanhada dos resultados do pré-teste, evidencia diferenças iniciais entre as turmas quanto à familiaridade com o conteúdo, reforçando a necessidade de considerar o conhecimento prévio dos participantes nas análises dos dados, bem como de adotar estratégias que levem em conta essas diferenças iniciais ao avaliar o impacto do uso do assistente de programação. Sendo assim, as análises de aprendizagem priorizaram a evolução individual dos estudantes ao longo do período da intervenção. Além da comparação direta entre os desempenhos no pré e no pós-teste, considerou-se a análise do ganho de aprendizagem como medida central para a comparação entre os grupos, conforme apresentado nas seções seguintes.

6.2.2 Resultados de Aprendizagem: Análise do Pré-teste e Pós-teste

Para avaliar o desenvolvimento da aprendizagem dos estudantes ao longo da intervenção didática, foram aplicados dois instrumentos complementares: um pré-teste, realizado no início da disciplina, e um pós-teste, aplicado ao final do período de intervenção.

6.2.2.1 Desempenho Inicial no Pré-teste

O desempenho inicial dos estudantes foi avaliado por meio de um pré-teste aplicado no início da disciplina, com pontuação variando de 0 a 10. A Tabela 5 apresenta as estatísticas descritivas do pré-teste para os grupos controle e experimental.

Tabela 5 - Estudo Final 1: Estatística Descritiva do Pré-teste

Grupo	N	Média	Mediana	Desvio-padrão	Mínimo	Máximo
Controle	13	4.00	4.50	2.56	0.00	7.50
Experimental	14	2.91	2.70	1.35	1.00	6.00

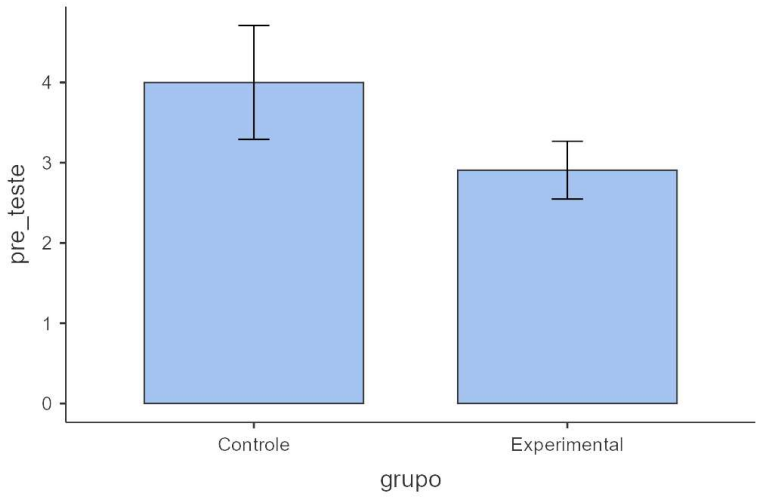
Fonte: Elaborado pela autora (2025)

No grupo de controle (n = 13), a média obtida no pré-teste foi de 4,00 (DP = 2,56), com valores variando entre 0,00 e 7,50. Observa-se, nesse grupo, maior variabilidade no desempenho inicial, indicando a presença de estudantes com diferentes níveis de conhecimento prévio, incluindo tanto alunos com conhecimentos mais limitados quanto aqueles com maior familiaridade com os conteúdos de programação web em JavaScript.

No grupo experimental (n = 14), a média do pré-teste foi de 2,91 (DP = 1,35), com pontuações variando entre 1,00 e 6,00. Comparativamente, esse grupo apresentou menor variabilidade e concentração predominante de estudantes nos níveis iniciais e intermediários da escala, sugerindo menor domínio prévio dos conteúdos abordados.

A Figura 47 ilustra as médias obtidas no pré-teste por grupo, evidenciando tendência de desempenho inicial superior no grupo de controle, ainda que com sobreposição dos intervalos de erro, de acordo com os resultados da análise estatística. As barras representam as médias e os intervalos de erro correspondem ao erro-padrão da média.

Figura 47 - Estudo Final 1: Gráfico de desempenho médio no pré-teste por grupo



Fonte: Elaborado pela autora (2025)

Antes da comparação entre os grupos, foram avaliados os pressupostos estatísticos de

normalidade e homogeneidade de variâncias. A normalidade da distribuição das notas do pré-teste foi verificada por meio do teste de *Shapiro–Wilk*, sendo considerada adequada em ambos os grupos ($p > 0,05$). Entretanto, o pressuposto de homogeneidade de variâncias, avaliado pelo teste de *Levene*, não foi atendido.

Para verificar possíveis diferenças iniciais entre os grupos, foi realizado um teste *t* de *Student* para amostras independentes com correção de *Welch*, adotado em função da violação do pressuposto de homogeneidade de variâncias. Os resultados indicaram que o grupo de controle apresentou desempenho médio superior ao do grupo experimental; entretanto, essa diferença não foi estatisticamente significativa ($t(17,9) = 1,37$; $p = 0,186$). O tamanho de efeito observado foi de magnitude moderada ($d = 0,54$) (COHEN, 1988), com intervalo de confiança amplo (IC 95% = [-0,23; 1,31]), o que reflete, em parte, o tamanho reduzido da amostra. Ainda assim, o valor do tamanho de efeito sugere uma diferença inicial de magnitude moderada no desempenho inicial entre os grupos.

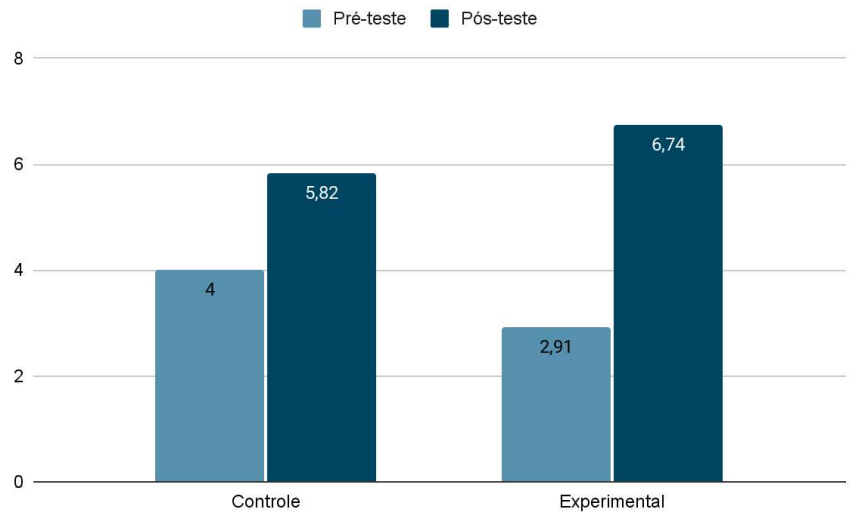
Os resultados do pré-teste, em conjunto com a autoavaliação do conhecimento prévio em JavaScript apresentada na seção anterior, evidenciam diferenças iniciais no desempenho dos estudantes, ainda que sem diferença estatisticamente significativa entre os grupos. Esses dados reforçam a necessidade de considerar o desempenho prévio nas análises subsequentes, justificando a adoção de estratégias analíticas centradas na evolução individual dos estudantes ao longo da intervenção.

6.2.2.2 Evolução do Desempenho ao Longo da Intervenção

A evolução do desempenho dos estudantes ao longo do período da intervenção foi analisada por meio da comparação entre os resultados do pré-teste e do pós-teste em cada grupo, utilizando testes *t* para amostras emparelhadas. Essa análise teve como objetivo verificar se houve progresso significativo no desempenho dos estudantes ao longo da disciplina, independentemente da comparação direta entre os grupos.

A Figura 48 ilustra a evolução do desempenho médio dos estudantes do pré-teste para o pós-teste em cada grupo.

Figura 48 - Estudo Final 1: Gráfico das médias do pré-teste e do pós-teste nos grupos controle e experimental



Fonte: Elaborado pela autora (2025)

No grupo de controle (n = 13), observou-se aumento da média do pré-teste (M = 4,00; DP = 2,56) para o pós-teste (M = 5,82; DP = 2,95). O teste *t* pareado indicou diferença estatisticamente significativa entre os dois momentos (t(12) = 2,45; p = 0,031), com tamanho de efeito moderado (d = 0,68). A verificação do pressuposto de normalidade não indicou violação (*Shapiro–Wilk*, p = 0,960), validando a aplicação do teste paramétrico. Esses resultados sugerem que os estudantes do grupo controle apresentaram evolução no desempenho ao longo da disciplina, compatível com o progresso esperado em um contexto de ensino tradicional.

No grupo experimental (n = 14), que utilizou o assistente de programação AIPairCode como par de programação durante as atividades em laboratório, observou-se uma melhora ainda mais expressiva no desempenho. A média do pré-teste (M = 2,91; DP = 1,35) aumentou consideravelmente no pós-teste (M = 6,74; DP = 2,21). O teste *t* para amostras emparelhadas indicou diferença altamente significativa entre os dois momentos (t(13) = 6,84; p < 0,001), acompanhada de um tamanho de efeito muito grande (d = 1,83). O teste de normalidade (*Shapiro–Wilk*, p = 0,321) confirmou a adequação do modelo paramétrico.

De forma geral, ambos os grupos apresentaram avanço no desempenho ao longo da disciplina. No entanto, os resultados indicam que a evolução observada no grupo experimental foi significativamente superior à do grupo de controle, especialmente considerando que os estudantes desse grupo iniciaram o estudo com níveis mais baixos de

conhecimento prévio. Esses achados reforçam a relevância de análises comparativas baseadas no ganho de aprendizagem, apresentadas na seção seguinte, como estratégia para avaliar de forma mais justa o impacto do uso do assistente de programação.

6.2.2.3 Comparação do Ganho de Aprendizagem entre os Grupos

Com o objetivo de comparar a evolução do desempenho dos estudantes ao longo da intervenção didática, analisou-se o ganho de aprendizagem, calculado como a diferença entre as notas obtidas no pós-teste e no pré-teste para cada participante. Essa abordagem permite avaliar o progresso individual ao longo do período da disciplina, minimizando o efeito de diferenças iniciais entre os grupos. As notas de cada aluno no pré-teste e no pós-teste, bem como os ganhos de aprendizagem individuais são apresentados no Apêndice VI.

A Tabela 6 apresenta as estatísticas descritivas do ganho de aprendizagem para os grupos controle e experimental.

Tabela 6 - Estudo Final 1: Estatísticas descritivas do ganho de aprendizagem

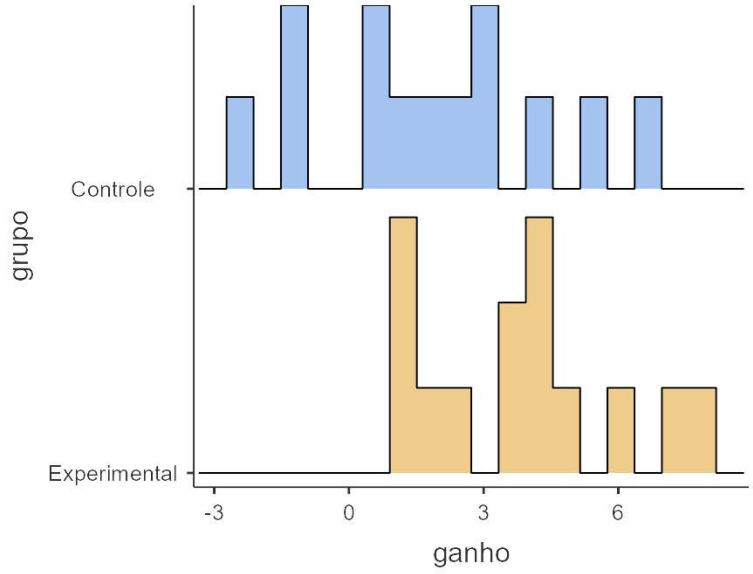
								Shapiro-Wilk	
	grupo	N	Média	Mediana	Desvio-padrão	Mínimo	Máximo	W	p
ganho	Controle	13	1.82	2.00	2.67	-2.40	6.90	0.977	.960
	Experimental	14	3.83	3.95	2.09	1.20	7.90	0.932	.321

Fonte: Elaborado pela autora (2025)

O grupo controle (n = 13) apresentou ganho médio de 1,82 pontos (DP = 2,67), com maior variabilidade e presença de valores negativos, indicando que nem todos os estudantes apresentaram evolução positiva ao longo do período. Em contraste, o grupo experimental (n = 14) apresentou ganho médio de 3,83 pontos (DP = 2,09), com todos os valores positivos, evidenciando progresso mais consistente entre os estudantes.

A Figura 49 apresenta a distribuição do ganho de aprendizagem nos grupos controle e experimental. Observa-se maior dispersão no grupo controle, incluindo valores negativos, enquanto o grupo experimental apresenta ganhos predominantemente positivos e mais concentrados, corroborando os resultados da análise estatística.

Figura 49 - Estudo Final 1: Distribuição do ganho de aprendizagem nos grupos controle e experimental



Fonte: Elaborado pela autora (2025)

Antes da comparação entre os grupos, foram verificados os pressupostos estatísticos. O teste de *Shapiro–Wilk* indicou distribuição aproximadamente normal do ganho de aprendizagem em ambos os grupos ($p > 0,05$), e o teste de *Levene* não apontou violação do pressuposto de homogeneidade de variâncias ($p = 0,406$), indicando adequação para a aplicação de testes paramétricos.

Para comparar o ganho médio entre os grupos, foi realizado um teste *t* de *Student* para amostras independentes com correção de *Welch* que foi mantido para manter a robustez. Os resultados indicaram diferença estatisticamente significativa em favor do grupo experimental ($t(22,8) = –2,17$; $p = 0,041$), com tamanho de efeito elevado ($d = 0,84$), sugerindo um impacto relevante da intervenção apoiada pelo assistente de programação sobre o ganho de aprendizagem.

Os resultados indicam que, embora ambos os grupos tenham apresentado avanço ao longo da disciplina, o grupo que utilizou o assistente de programação como par de programação obteve um ganho médio significativamente maior. Esse resultado sugere que o uso da ferramenta pode ter contribuído para uma evolução mais consistente no desempenho dos estudantes, especialmente considerando as diferenças iniciais de conhecimento prévio entre os grupos.

Observa-se que o maior ganho de aprendizagem obtido pelo grupo experimental pode

ser interpretado não apenas em termos quantitativos, mas também em função do perfil acadêmico dos estudantes e do tipo de apoio oferecido ao longo da intervenção. Considerando que os participantes desse grupo se encontravam majoritariamente nos semestres iniciais do curso, os conteúdos propostos representaram desafios compatíveis com sua ZDP, favorecendo avanços mais expressivos quando acompanhados por mediação pedagógica adequada. Essa interpretação será aprofundada na discussão integrada dos resultados, articulando os dados de desempenho, motivação e interação dos estudantes com o assistente de programação.

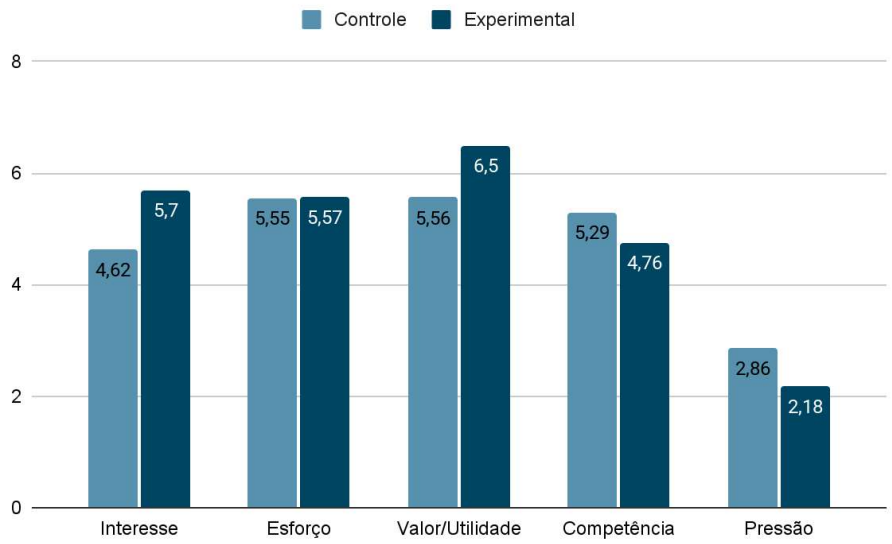
6.2.3 Motivação e Percepções dos Estudantes (IMI)

Com o objetivo de analisar a motivação e a percepção dos estudantes associadas à participação nas atividades desenvolvidas ao longo da intervenção, foi aplicado, ao final do período letivo, uma versão do questionário IMI (Apêndice III), com as questões adaptadas para a atividade. O instrumento contemplou cinco dimensões: Interesse/Prazer, Esforço/Importância, Valor/Utilidade percebida, Competência percebida e Pressão/Tensão, conforme apresentado no Apêndice III. O questionário foi respondido por 25 estudantes, sendo 11 do grupo controle e 14 do grupo experimental, permitindo avaliar diferentes aspectos motivacionais relacionados às atividades realizadas.

As respostas, obtidas em escala *Likert* de 1 a 7, foram analisadas por meio de estatísticas descritivas e comparadas entre os grupos controle e experimental, buscando identificar tendências motivacionais distintas. Considerando a natureza dos dados, o tamanho reduzido da amostra e a presença de casos omissos no grupo de controle (n = 11), optou-se pela utilização do teste não paramétrico *U* de *Mann–Whitney* para a comparação entre os grupos. Essa escolha também foi fundamentada na identificação de violações de pressupostos paramétricos em algumas dimensões do instrumento, incluindo desvios de normalidade e heterogeneidade de variâncias. Para cada dimensão analisada, são apresentadas as estatísticas descritivas por grupo, os valores do teste U, o nível de significância estatística e o tamanho de efeito, estimado por meio da correlação bisserial de ordens (r).

A Figura 50 ilustra graficamente as médias obtidas em cada dimensão do IMI, possibilitando uma visualização comparativa entre os grupos.

Figura 50 - Estudo Final 1: Média do IMI por Dimensão



Fonte: Elaborado pela autora (2025)

A análise geral dos resultados evidencia diferenças no padrão motivacional percebido pelos estudantes dos dois grupos, ainda que nem todas as dimensões tenham apresentado diferenças estatisticamente significativas.

Na dimensão Interesse/Prazer, o grupo experimental apresentou média superior em comparação ao grupo de controle. Conforme ilustrado na figura 50, os estudantes que utilizaram o assistente de programação relataram maior interesse e prazer durante a realização das atividades. As estatísticas descritivas indicaram média 5,70 (DP = 0,88) no grupo experimental e média de 4,62 (DP = 1,77) no grupo de controle. A comparação entre os grupos por meio do teste *U de Mann–Whitney* não indicou diferença estatisticamente significativa (U = 47,5; p = 0,112), porém o tamanho de efeito observado foi de magnitude moderada (r = 0,38), indicando uma tendência favorável ao grupo experimental. Esses resultados sugerem maior interesse e prazer percebidos pelos estudantes que utilizaram o AIPairCode, indicando que, possivelmente, a presença do assistente como par de programação pode ter contribuído para tornar as atividades mais atrativas e envolventes, especialmente para estudantes com menor familiaridade inicial com o conteúdo.

Em relação à dimensão Esforço/Importância, ambos os grupos apresentaram médias semelhantes. O grupo de controle apresentou média de 5,55 (DP = 0,97), enquanto o grupo experimental obteve média de 5,57 (DP = 0,83). O teste *U de Mann–Whitney* não indicou diferença significativa entre os grupos (U = 67,5; p = 0,620), com tamanho de efeito pequeno (r = -0,12). Esses dados indicam que os estudantes, independentemente do uso do assistente

de programação, atribuíram níveis semelhantes de esforço e importância às atividades propostas, sugerindo que a intervenção não alterou significativamente a percepção de comprometimento com as tarefas.

Na dimensão Valor/Utilidade, observou-se uma diferença mais expressiva entre os grupos. O grupo experimental apresentou média de 6,50 (DP = 0,61), enquanto o grupo de controle obteve média de 5,56 (DP = 1,69). A comparação por meio do teste *U* de *Mann–Whitney* indicou tendência favorável ao grupo experimental, embora sem significância estatística ao nível de 5% (U = 43,5; p = 0,065), com um tamanho de efeito moderado (r = 0,44). Esse resultado é coerente com a proposta do assistente de programação como ferramenta de apoio contínuo, capaz de fornecer *feedback* imediato e orientar o processo de resolução de problemas, favorecendo uma maior percepção de relevância das atividades desenvolvidas.

Na dimensão Competência Percebida, o grupo de controle apresentou média ligeiramente superior (M = 5,29; DP = 0,99) em comparação ao grupo experimental (M = 4,76; DP = 1,04). No entanto, essa diferença não foi estatisticamente significativa segundo o teste U de *Mann–Whitney* (U = 54,5; p = 0,226), e o tamanho de efeito observado foi pequeno a moderado (r = -0,29). Esse resultado pode estar associado ao perfil acadêmico dos participantes, uma vez que o grupo de controle foi composto majoritariamente por estudantes de semestres mais avançados, possivelmente mais confiantes em relação às próprias habilidades em programação, independentemente da intervenção realizada.

Quanto à dimensão Pressão/Tensão, ambos os grupos relataram níveis relativamente baixos. O grupo de controle apresentou média de 2,86 (DP = 1,18), enquanto o grupo experimental obteve média de 2,18 (DP = 0,93). A comparação entre os grupos não indicou diferença estatisticamente significativa (U = 49,5; p = 0,137), embora o tamanho de efeito observado tenha sido de magnitude moderada (r = -0,36). Esse padrão sugere menor pressão percebida pelos estudantes do grupo experimental, o que pode estar relacionado à disponibilidade contínua de suporte oferecida pelo assistente de programação durante a resolução das atividades.

A análise conjunta das dimensões do IMI indica que o grupo experimental, composto predominantemente por estudantes de semestres iniciais, apresentou percepções mais favoráveis em aspectos relacionados ao interesse, valor/utilidade e menor percepção de pressão, quando comparado ao grupo de controle. Em contrapartida, o grupo de controle, formado majoritariamente por estudantes de semestres mais avançados, apresentou níveis

semelhantes de esforço, maior percepção de competência e, de modo geral, indicadores menos favoráveis nas dimensões motivacionais analisadas.

Esses resultados podem estar associados a fatores contextuais relacionados ao estágio do curso, como redução do engajamento associada à repetição de atividades acadêmicas, menor efeito de novidade pedagógica e experiências prévias acumuladas em disciplinas de programação. Sob a perspectiva da ZDP, o assistente de programação tende a atuar como mediador pedagógico mais relevante para estudantes em estágios iniciais de aprendizagem, oferecendo suporte compatível com suas necessidades imediatas e favorecendo maior engajamento e percepção de utilidade das atividades.

De forma geral, os resultados do IMI sugerem que o uso do assistente de programação esteve associado a uma experiência motivacional mais favorável, especialmente no que se refere à percepção de utilidade, ao interesse pelas atividades e à redução da pressão durante o processo de aprendizagem. Embora não tenham sido observadas diferenças estatisticamente significativas, os tamanhos de efeito moderados sugerem tendências consistentes, que complementam os resultados de aprendizagem apresentados nas seções anteriores e reforçam o potencial do assistente como apoio pedagógico no ensino de programação.

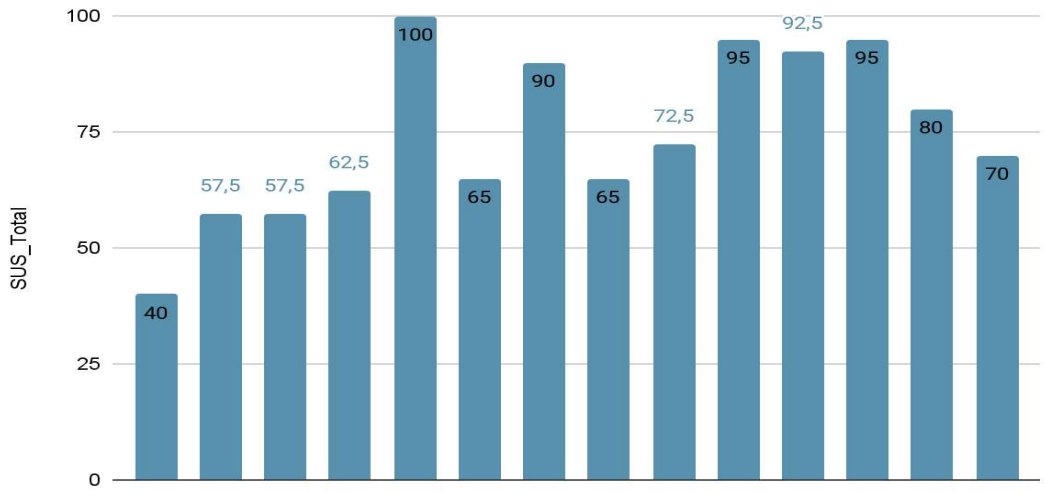
6.2.4 Avaliação de Usabilidade do Assistente (SUS)

A usabilidade percebida do assistente de programação AIPairCode foi avaliada por meio do *System Usability Scale* (SUS) (BROOKE, 1996), aplicado ao final do período de intervenção exclusivamente ao grupo experimental, uma vez que apenas esses estudantes utilizaram a ferramenta durante as atividades em laboratório. Participaram dessa etapa 14 estudantes, correspondendo à totalidade dos participantes ativos do grupo experimental no Estudo Final 1.

O questionário SUS é composto por dez itens em escala *Likert* de cinco pontos, alternando afirmações positivas e negativas sobre a experiência de uso do sistema (Apêndice IV). As respostas foram pontuadas conforme o procedimento padrão de cálculo do SUS, resultando em um escore global variando de 0 a 100, no qual valores mais elevados indicam melhor usabilidade percebida.

A Figura 51 apresenta a distribuição dos escores SUS obtidos pelas avaliações de cada estudante do grupo experimental. O escore médio de usabilidade foi de 74,46 (DP = 17,98), com mediana de 71,25 e variação entre 40,0 e 100,0.

Figura 51 - Estudo Final 1: Distribuição dos escores SUS por estudante

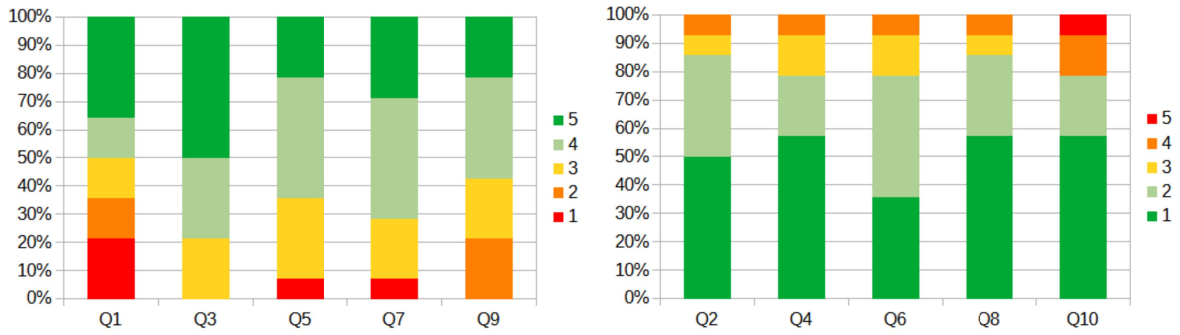


Fonte: Elaborado pela autora (2025)

Considerando o valor de referência utilizado na literatura (LEWIS; SAURO, 2018), no qual escores acima de 68 indicam usabilidade percebida acima da média, os resultados sugerem que, em termos gerais, o AIPairCode foi avaliado como um sistema de boa usabilidade. De acordo com a escala de interpretação de Bangor *et al.* (2008), a pontuação média obtida (74,46) situa a ferramenta na faixa de classificação entre “Bom” e “Excelente”, indicando que, de modo geral, os alunos não encontraram barreiras severas de interação com a extensão AIPairCode durante as atividades de programação. Observa-se, entretanto, variabilidade nas respostas, indicando experiências de uso heterogêneas entre os participantes, incluindo a presença de avaliações significativamente mais baixas.

Com o objetivo de aprofundar a compreensão sobre os aspectos específicos da experiência de uso do assistente, realizou-se uma análise das respostas item a item do questionário SUS. A Figura 52 apresenta o detalhamento das respostas do SUS por questão (Q1 a Q10), com os itens formulados positivamente apresentados à esquerda e os itens negativos à direita, permitindo analisar a distribuição das respostas e identificar os pontos fortes e os aspectos que demandam maior atenção no AIPairCode. No gráfico é possível observar o percentual de respostas bem positivas (verde-escuro), positivas (verde-claro), neutras (amarelo), negativas (laranja) e bem negativas (vermelho).

Figura 52 - Estudo Final 1: Detalhamento das respostas do SUS por questão



Fonte: Elaborado pela autora (2025)

De forma geral, observa-se maior concentração de respostas favoráveis nas questões relacionadas à facilidade de uso, baixa complexidade percebida e clareza do sistema, indicando que os estudantes, em sua maioria, não consideraram o assistente confuso ou excessivamente complexo. Em contrapartida, algumas questões apresentaram maior dispersão nas respostas, especialmente aquelas associadas à intenção de uso frequente, à confiança durante o uso e à integração das funcionalidades.

A análise conjunta das questões permite identificar padrões relevantes na percepção dos estudantes:

- Facilidade de uso e complexidade (Q2, Q3 e Q8): a predominância de respostas favoráveis nessas questões indica que o sistema foi percebido como fácil de usar e pouco confuso, sugerindo uma interface compreensível e acessível para a maioria dos participantes.
- Aprendizado inicial (Q4 e Q10): os resultados indicam que, em geral, os estudantes consideraram o sistema de fácil aprendizado, embora tenham sido identificadas respostas discrepantes que sugerem maior dificuldade inicial para alguns usuários.
- Integração e consistência (Q5, Q6 e Q7): apesar de avaliações majoritariamente positivas, essas questões apresentaram maior variabilidade nas respostas, apontando possíveis oportunidades de melhoria na integração das funcionalidades e na fluidez da experiência de uso.
- Confiança e intenção de uso (Q1 e Q9): essas dimensões concentraram as respostas menos favoráveis, sugerindo que, embora o sistema tenha sido considerado utilizável, nem todos os estudantes demonstraram intenção clara de utilizá-lo com frequência ou plena confiança durante o uso.

Em síntese, os resultados do SUS indicam que o AIPairCode apresentou boa usabilidade percebida, situando-se acima do valor de referência adotado na literatura. No entanto, a análise detalhada revela heterogeneidade na experiência dos usuários, com a presença de avaliações discrepantes e indícios de que fatores como adaptação inicial, integração das funcionalidades e aceitação da proposta pedagógica influenciaram a percepção de usabilidade.

Esses dados sugerem que a avaliação do SUS não deve ser interpretada exclusivamente como uma medida das características da interface, mas também como reflexo do nível de adesão dos estudantes à ferramenta e de sua disposição em utilizá-la como parte do processo de aprendizagem.

6.2.5 Percepções dos Estudantes Sobre o Uso do Assistente de Programação

Com o objetivo de complementar e contextualizar os resultados quantitativos de usabilidade obtidos por meio do questionário SUS, os estudantes do grupo experimental responderam, ao final da intervenção, a três questões abertas relacionadas à experiência de uso do AIPairCode. As questões abordaram os aspectos mais apreciados da ferramenta, as principais dificuldades enfrentadas durante o uso e sugestões de melhoria para o comportamento do assistente. As respostas foram analisadas de forma integrada, buscando identificar padrões recorrentes e estabelecer relações com os resultados de usabilidade previamente apresentados.

De modo geral, uma parcela significativa dos estudantes destacou como aspecto positivo o papel do assistente como mediador do raciocínio, incentivando a reflexão e a construção ativa do conhecimento, em vez de fornecer respostas prontas. Esse posicionamento é evidenciado em relatos como o do AlunoExperimental_04, que afirmou ter gostado *"da parte que ele incentiva o usuário a se perguntar sobre o que está fazendo e como poderia fazer diferente"*, e do AlunoExperimental_13, que resumiu sua percepção ao afirmar que o assistente *"induz o usuário a pensar"*. De forma semelhante, o AlunoExperimental_11 ressaltou que o assistente *"faz a gente pensar e desenvolver a lógica e o pensamento crítico"*, enquanto o AlunoExperimental_05 valorizou *"as interações e a ajuda sem dar a resposta"*.

Outros estudantes destacaram aspectos relacionados à integração do assistente com o ambiente de programação, especialmente o acesso simultâneo ao código e a inserção de comentários diretamente nos trechos em desenvolvimento. O AlunoExperimental_07 relatou que *"uma coisa que me chamou muito atenção foi ele comentar no código e também a*

questão de pedidos rápidos pra IA já estarem ali como opções", enquanto o AlunoExperimental_08 destacou positivamente *"o fato dele ter acesso ao código quase simultaneamente"*. Para o AlunoExperimental_12, a principal vantagem esteve na agilidade proporcionada pelo assistente, ao permitir *"ir codando e tirando as dúvidas rapidamente sem perder muito tempo"*.

Apesar dessas avaliações favoráveis, surgiram também relatos de frustração e resistência, especialmente relacionados ao comportamento do assistente em adotar uma postura predominantemente questionadora. Alguns estudantes manifestaram insatisfação com a ausência de respostas mais diretas, como expresso pelo AlunoExperimental_03, que afirmou que o assistente *"só fica fazendo pergunta e não é nada objetivo"*, e pelo AlunoExperimental_09, que relatou: *"quando pedi ajuda com algo que não sabia ele me respondeu com outras perguntas, não gostei disso, se é pra ter um parceiro de programação que seja um bom que realmente me ajude"*. O AlunoExperimental_02 também apontou como dificuldade o fato de o assistente *"não gerar respostas direto ao ponto"*.

Em alguns casos, os estudantes relataram que o comportamento do assistente se tornava redundante ou pouco eficiente, como no caso do AlunoExperimental_05, que mencionou que os textos eram *"meio redundantes"* e *"enrolavam uma linha transformando em cinco"*, ou do AlunoExperimental_04, que observou que o assistente *"às vezes se perdia no assunto"*. O AlunoExperimental_06 destacou que as dicas eram *"bem abertas"* e que o assistente frequentemente *"respondia com outra pergunta"*, o que inicialmente gerou confusão, embora o próprio estudante reconheça que, ao continuar a interação, isso acabou ajudando na construção do raciocínio.

As sugestões de melhoria apresentadas pelos estudantes reforçam essa tensão entre diferentes expectativas em relação ao papel da IA. Alguns participantes sugeriram que o assistente poderia oferecer respostas mais objetivas em determinados momentos, como o AlunoExperimental_13, que sugeriu *"dar algumas respostas mais objetivas ao invés de sempre perguntar a forma como o usuário está pensando"*. Outros propuseram ajustes no ritmo das respostas, como o AlunoExperimental_10, que sugeriu *"não fornecer tantos dados de uma vez para o usuário, mas instigar a pensar um pouco mais"*, evidenciando a busca por um equilíbrio entre orientação direta e estímulo à reflexão.

Houve também sugestões de cunho mais prático e técnico, como a possibilidade de utilizar o assistente no computador pessoal (AlunoExperimental_12) ou melhorias pontuais de usabilidade, como o processo de login (AlunoExperimental_07).

A análise dessas respostas permite compreender de forma mais aprofundada a heterogeneidade observada nos escores do SUS, discutida na seção anterior. Estudantes que demonstraram maior aceitação do papel do assistente como mediador do raciocínio relataram experiências mais positivas, enquanto aqueles que esperavam uma atuação mais diretiva da IA manifestaram maior frustração e resistência ao modelo de interação adotado. Essa divergência ajuda a explicar a variabilidade nos escores de usabilidade, especialmente em itens do SUS relacionados à confiança no uso do sistema e à intenção de utilizá-lo com frequência.

Dessa forma, os relatos dos estudantes indicam que a usabilidade percebida do AIPairCode não está associada apenas às características técnicas da interface, mas também à aceitação pedagógica do modelo de interação proposto. Em síntese, as respostas abertas reforçam que o assistente foi amplamente reconhecido como uma ferramenta útil e alinhada a práticas de aprendizagem ativa, ao mesmo tempo em que evidenciam desafios relacionados à adaptação de alguns estudantes a um modelo de apoio que privilegia a reflexão, a autonomia e a construção gradual do conhecimento. Esses achados reforçam a importância de considerar, de maneira integrada, aspectos técnicos, pedagógicos e motivacionais na avaliação de assistentes de programação baseados em IA.

6.2.6 Padrões de Interação com o AIPairCode

Além da análise dos resultados de desempenho e dos indicadores de motivação apresentados nas seções anteriores, torna-se fundamental compreender como os estudantes interagiram com o AIPairCode ao longo das atividades propostas. Nesse sentido, esta seção tem como objetivo analisar os padrões de interação estabelecidos entre os alunos e o assistente de programação, a partir da categorização das perguntas formuladas pelos estudantes e dos tipos de respostas fornecidas pela IA.

Do ponto de vista metodológico, a análise das interações foi fundamentada na Análise de Conteúdo proposta por Bardin (2016), cujo procedimento seguiu, de forma adaptada ao contexto do estudo, as três fases clássicas da Análise de Conteúdo: (i) pré-análise, na qual foi realizada a organização do corpus de dados, composto pelos registros textuais das interações entre alunos e IA; (ii) exploração do material, etapa em que as interações foram codificadas e categorizadas; e (iii) tratamento dos resultados e interpretação, envolvendo a sistematização das categorias, o cálculo das frequências e percentuais e a posterior análise dos padrões observados.

As interações analisadas neste estudo foram registradas automaticamente pelo AIPairCode durante o uso da ferramenta pelos estudantes em atividades práticas de programação. Cada solicitação realizada pelo aluno e a respectiva resposta da IA foram armazenadas em tempo real em um banco de dados no *Firebase* (*Firestore Database*), juntamente com informações contextuais, como identificação do aluno, exercício em execução e papel assumido pela IA no momento da interação (piloto ou navegador). Após o término da intervenção, os registros completos de interação foram exportados do Firebase para arquivos no formato .csv, os quais constituíram o corpus de dados utilizado na Análise de Conteúdo.

Na fase de exploração do material, as interações foram analisadas a partir de duas dimensões complementares: (i) a categoria da solicitação do aluno, que indica o tipo de apoio buscado pelo estudante; e (ii) o tipo de resposta da IA, que descreve a estratégia comunicacional e pedagógica adotada pelo AIPairCode. Cada interação foi codificada de acordo com a categoria predominante em cada dimensão, admitindo-se, quando necessário, a identificação de categorias secundárias em interações híbridas.

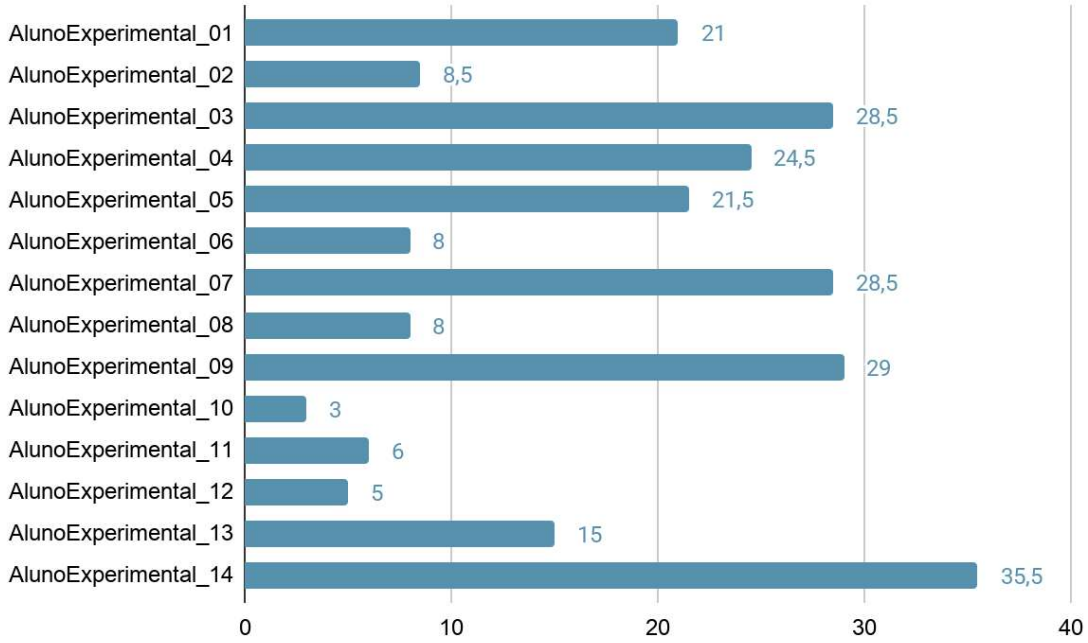
Essa abordagem possibilita não apenas descrever quantitativamente o uso do AIPairCode, mas também compreender qualitativamente como e para que os alunos recorreram à IA, oferecendo subsídios para relacionar os padrões de interação observados com a aprendizagem, a autonomia e os aspectos motivacionais.

6.2.6.1 Categorias de solicitação dos estudantes ao AIPairCode

Nesta seção, analisa-se o tipo de apoio buscado pelos estudantes ao interagir com o AIPairCode, a partir da categorização das solicitações registradas durante as atividades práticas. O número de interações variou consideravelmente entre os participantes, com média geral de 17,3 interações por aluno por exercício.

A Figura 53 apresenta a média de interações por aluno/exercício.

Figura 53 - Estudo Final 1: Média de interações por aluno/exercício



Fonte: Elaborado pela autora (2025)

Enquanto alguns estudantes apresentaram uso intensivo da ferramenta (por exemplo, AlunoExperimental_14, com média de 35,5 interações por exercício; AlunoExperimental_09 com média de 29 interações; AlunoExperimental_07 e AlunoExperimental_03, com média de 28,5 cada; AlunoExperimental_04, com média de 24,5 interações), outros recorreram ao assistente poucas vezes (como o AlunoExperimental_12, com média de 5 interações e o AlunoExperimental_10, com média de 3 interações por exercício). Esse comportamento sugere a existência de perfis distintos de engajamento com a ferramenta, possivelmente relacionados tanto ao nível de autonomia prévia quanto às dificuldades enfrentadas durante a execução das tarefas.

No que se refere à dinâmica de interação, o AIPairCode foi programado, neste estudo, para iniciar as interações no papel de navegador, alternando automaticamente para o papel de piloto a cada duas solicitações do estudante. No papel de navegador, a IA não fornecia trechos de código, limitando-se a orientações conceituais e estratégicas, correções e orientações passo a passo. Já no papel de piloto, eram permitidos exemplos parciais e pequenos trechos ilustrativos de código, conforme definido através do *prompt* das requisições e dos *guardrails* implementados na ferramenta.

Quanto ao papel assumido pela IA ao longo das interações, observou-se distribuição relativamente equilibrada, com leve predominância do papel navegador (52,0%) em relação

ao piloto (48,0%). Essa proximidade é relevante, pois indica que os estudantes não utilizaram o AIPairCode apenas como gerador de soluções prontas, mas também como apoio para organização do pensamento, tomada de decisões e planejamento, aspectos condizentes com a proposta de par de programação baseada na alternância de papéis e com uma abordagem orientada ao desenvolvimento da autonomia.

Embora essa estratégia de alternância tenha sido concebida para estimular a reflexão e evitar a dependência excessiva da geração de código, foram registrados relatos informais de frustração por parte de alguns estudantes, especialmente quando a alternância automática não coincidia com o tipo de apoio esperado naquele momento da atividade. Esses *feedbacks* contribuíram para o refinamento do mecanismo de alternância na versão subsequente da ferramenta, utilizada no Estudo Final 2, no qual foram implementados ajustes para tornar o controle do papel da IA mais flexível e sensível ao contexto da interação.

Com base na análise de conteúdo das interações, conduzida segundo os procedimentos propostos por Bardin (2016), as solicitações dos estudantes ao AIPairCode foram classificadas em oito categorias principais, apresentadas na Tabela 7. Essas categorias refletem diferentes estratégias adotadas pelos alunos ao utilizar a IA durante a atividade.

Tabela 7 - Estudo Final 1: Categorias de solicitação dos estudantes ao AIPairCode

Categoria	**Descrição**	**Frequência (%)**
Planejamento e estratégia	Solicitações relacionadas à organização do raciocínio, decomposição do problema ou definição de passos para a solução.	27,3
Validação de solução	Solicitações para confirmar se uma solução, ideia ou raciocínio está correto.	18,8
Implementação guiada / Como fazer	Solicitação de dicas e colaboração passo a passo.	18,6
Compreensão conceitual	Solicitações voltadas à compreensão de conceitos, estruturas ou princípios da linguagem ou do problema, sem pedido direto de código ou correção de erro.	12,5
Depuração e resolução de erros	Solicitações relacionadas à identificação, compreensão ou correção de erros em código próprio.	10,6
Geração direta de código	Solicitações explícitas por trechos de código ou soluções prontas.	6,6
Revisão e refatoração	Solicitações de avaliação, melhoria ou reorganização do código próprio já existente.	2,8
Metacognição	Solicitações que expressem emoções, dificuldades pessoais ou reflexões sobre o próprio aprendizado.	2,8

Fonte: Elaborado pela autora (2025)

A análise das categorias evidencia a predominância de solicitações associadas à condução da tarefa e à organização do trabalho, seguidas de pedidos de validação e de orientação prática. A categoria mais frequente foi Planejamento e Estratégia (27,3%), indicando que os estudantes recorreram ao AIPairCode principalmente para organizar o caminho de resolução, definir etapas e estruturar a solução antes ou durante a implementação. Exemplos recorrentes desse tipo de solicitação incluíram perguntas como: *"Como posso começar esse exercício?"* (AlunoExperimental_01), *"Seria melhor eu começar pelo JS ou html?"* (AlunoExperimental_07), *"Acredito que fiz toda essa estrutura inicial corretamente, como posso prosseguir agora?"* (AlunoExperimental_04) ou ainda *"Primeiro preciso criar a Página de login, certo? Para isso devo usar o form, correto?"* (AlunoExperimental_12).

Esse padrão é consistente com o papel de mediação esperado de um par de programação, especialmente em contextos nos quais o aluno ainda não domina plenamente o processo de decomposição de problemas, aspecto diretamente relacionado à ZDP, na qual o suporte externo contribui para que o estudante avance em direção a formas mais autônomas de organização e execução de tarefas.

Em seguida, destacam-se Validação de Solução (18,8%) e Implementação Guiada / Como Fazer (18,6%). No caso da validação, alguns exemplos de solicitações são: *"O arquivo de css está na pasta fora do frontend, está correto?"* (AlunoExperimental_03), *"Está correto agora?"* (AlunoExperimental_09) ou *"Acredito que o cadastro.html esta finalizado, pode me confirmar?"* (AlunoExperimental_11). Já as interações classificadas como implementação guiada envolveram perguntas do tipo: *"Fiz a base front-end. Como eu passo essas informações para o servidor?"* (AlunoExperimental_01), *"Como fazer um <select>?"* (AlunoExperimental_05) ou *"Me ajude a produzir o cadastro html baseado no js que te mandei"* (AlunoExperimental_08). Esses resultados sugerem que muitos estudantes utilizaram o AIPairCode como um recurso para confirmar decisões e transformar intenções em ações concretas, reduzindo incertezas durante o processo de resolução de problemas.

Em termos pedagógicos, esse tipo de solicitação pode refletir a busca por *feedback* imediato e redução de incertezas durante a resolução de problemas, funcionando como suporte para manter o fluxo de trabalho. Esse suporte contínuo tende a contribuir para o engajamento, evitando interrupções prolongadas causadas por bloqueios conceituais ou operacionais.

A categoria Compreensão conceitual apresentou 12,5% de ocorrências, sinalizando que uma parcela relevante das interações esteve voltada ao entendimento de conceitos da

linguagem de programação. Exemplos típicos incluíram perguntas como *"Como pego o elemento pai em javascript?"* (AlunoExperimental_02), *"Defina label"* (AlunoExperimental_06) ou *"Me explique como funciona o sessionStorege.setItem."* (AlunoExperimental_14). Ainda que não seja a categoria predominante, sua presença é importante porque indica que o assistente foi utilizado também como mediador de aprendizagem conceitual e não apenas como apoio operacional.

As solicitações relacionadas à Depuração e Resolução de Erros representaram 10,6%, apontando que o AIPairCode também foi utilizado para identificar causas de falhas, compreender mensagens de erro e ajustar trechos problemáticos. Alguns exemplos de questionamentos foram: *"Está tendo problema de salvar esse livro no banco de dados como alterar para funcionar?"* (AlunoExperimental_10), *"como fazer dar certo esse form?"* (AlunoExperimental_09) ou *"Meu query selector não está funcionando"* (AlunoExperimental_07). Em atividades de programação, esse tipo de interação é relevante, pois a depuração demanda raciocínio diagnóstico e frequentemente se encontra acima do nível de desempenho independente de estudantes iniciantes; portanto, o apoio da IA pode atuar como andaime (*scaffolding*) orientando o aluno na identificação de causas, testes de soluções e correções graduais.

As categorias menos frequentes incluem Geração direta de código (6,6%) e Revisão e refatoração (2,8%), além de Metacognição (2,8%). A baixa proporção de pedidos de código pronto é consistente com a proposta do AIPairCode enquanto par de programação com orientação pedagógica e com o foco em promover participação ativa do estudante. Já as interações classificadas como metacognitivas, ainda que pouco numerosas, envolveram reflexões como *"Não tenho certeza, preciso de ajuda"* (AlunoExperimental_04), *"Eu não sei como arrumar"* (AlunoExperimental_03) ou *"Sim, isso eu consigo tranquilamente"* (AlunoExperimental_08), indicando momentos de autorreflexão sobre o próprio processo de aprendizagem, associados ao desenvolvimento gradual da autonomia.

Em síntese, os dados indicam que os estudantes utilizaram o AIPairCode predominantemente como um apoio para planejar, estruturar e validar suas soluções, combinando suporte estratégico e orientação prática. Esse padrão é compatível com a concepção de IA como par de programação e mediador pedagógico do processo de aprendizagem.

6.2.6.2 Tipos de resposta do AIPairCode às solicitações dos estudantes

Nesta seção foram analisadas as estratégias de resposta adotadas pelo AIPairCode ao longo das interações, com o objetivo de identificar de que forma a IA respondeu às solicitações dos alunos, buscando compreender como essas respostas se alinham à proposta pedagógica do sistema enquanto par de programação e mediador do processo de aprendizagem.

A análise das categorias de resposta do AIPairCode permite ainda compreender, de maneira objetiva, o comportamento adotado pelo assistente durante as interações, fornecendo subsídios para interpretar as percepções positivas e negativas relatadas pelos estudantes quanto à postura do sistema.

As respostas do AIPairCode foram categorizadas a partir de uma análise de conteúdo, seguindo os procedimentos descritos na seção 6.2.6, considerando o tipo predominante de resposta retornada pela IA em cada interação. Ressalta-se que uma mesma resposta poderia contemplar mais de uma estratégia, como combinar uma explicação conceitual com perguntas orientadoras. Por essa razão, os resultados são apresentados em termos percentuais, de modo a refletir a incidência relativa de cada categoria no conjunto total de respostas analisadas.

A Figura 54 apresenta graficamente a distribuição percentual das categorias de resposta do AIPairCode no Estudo Final 1.

Figura 54 - Estudo Final 1: Distribuição percentual dos tipos de resposta da IA



Fonte: Elaborado pela autora (2025)

Os dados indicam uma clara predominância da categoria Pergunta socrática, que correspondeu a 52,8% das respostas do AIPairCode. Esse resultado evidencia que a estratégia mais frequentemente adotada pela IA foi a de provocar reflexão por meio de questionamentos, em vez de fornecer respostas diretas. Exemplos recorrentes desse tipo de resposta incluíram intervenções como "*Você conhece a estrutura básica de um elemento <select> com opções em HTML? Se não, que tal pensarmos juntos como montar isso para os destinos?*", "*Será que o uso do operador lógico `&&` no seu `if` está correto?*" ou "*Você sabe como poderia usar o sessionStorage para verificar se o usuário está logado assim que a página carregar?*".

Essa predominância está alinhada ao design pedagógico do AIPairCode, que incorpora elementos do método socrático com o objetivo de estimular o raciocínio ativo do estudante, favorecendo a construção do conhecimento dentro de sua ZDP. Ao devolver perguntas ao aluno, a IA atua como mediadora do pensamento, incentivando a explicitação de hipóteses e a reorganização do raciocínio.

A segunda categoria mais frequente foi Orientação procedural (passos), com 15,0% das respostas. Nesse caso, o AIPairCode assumiu um papel mais diretivo, oferecendo sequências de ações ou sugestões de encaminhamento, sem, contudo, apresentar soluções completas. Exemplos típicos envolveram orientações como "*Você pode criar uma função, por exemplo, chamada 'validarLogin' que vai ser chamada quando o usuário clicar no botão de login. Dentro dessa função, você pode pegar os valores digitados nos campos de usuário e senha, comparar com os valores fixos (ex: 'admin' e '1234'), e, se estiver correto, realizar uma ação (como mostrar mensagem ou redirecionar).*" ou "*comece criando a rota no backend antes de integrar com o frontend*". Esse tipo de resposta funciona como um andaime, fornecendo suporte temporário para que o aluno consiga avançar na tarefa.

As categorias Exemplo Parcial (sem código completo) (10,4%) e *Feedback* Avaliativo (8,1%) também apresentaram incidência relevante. Os exemplos parciais foram utilizados principalmente quando a IA assumia o papel de piloto, oferecendo pequenos trechos ilustrativos ou estruturas incompletas para apoiar a compreensão, enquanto o *feedback* avaliativo apareceu em situações de validação, com respostas como "*Seu código está correto agora, com o <select> envolvendo somente as opções de destino, e os outros campos fora dele, como deveria ser!*" ou "*Ótimo! Você fez uma boa tentativa usando o `filter` para achar usuários com id igual a `params.id`*". Ambas as estratégias contribuem para reduzir a incerteza do estudante, sem comprometer sua participação ativa na construção da solução.

As respostas classificadas como Explicação conceitual representaram 6,0% do total e

estiveram associadas a momentos em que o estudante demonstrou dúvidas sobre conceitos fundamentais, como eventos, estruturas de controle ou mecanismos de armazenamento no navegador. Ainda que menos frequentes, essas respostas são pedagogicamente relevantes, pois reforçam o papel da IA como mediadora conceitual.

Por outro lado, a Geração direta de código foi pouco frequente, correspondendo a apenas 0,8% das respostas. Esse dado é coerente com a proposta do AIPairCode e com as restrições impostas pelo papel de navegador e pelos *guardrails* implementados na ferramenta, reforçando que a IA não foi utilizada predominantemente como um gerador automático de soluções, mas como um parceiro de reflexão e apoio ao desenvolvimento do raciocínio do estudante.

Em conjunto, os resultados sugerem que o AIPairCode atuou principalmente por meio de estratégias que promovem reflexão, organização do pensamento e avanço gradual na resolução de problemas.

Embora a predominância da categoria Pergunta socrática nas respostas do AIPairCode (52,8%) seja coerente com a proposta pedagógica da ferramenta, orientada à promoção da reflexão e da autonomia, os dados qualitativos das interações e os relatos dos estudantes indicam que esse padrão nem sempre foi percebido de forma positiva. Em diversos momentos, especialmente em fases mais operacionais da atividade, alguns alunos demonstraram frustração diante de respostas excessivamente formuladas como perguntas, percebidas como pouco objetivas ou insuficientes para avançar na implementação.

Esse comportamento sugere que, embora a estratégia socrática seja pedagogicamente consistente, sua aplicação contínua e pouco contextualizada pode gerar tensão entre a intenção formativa da ferramenta e as expectativas imediatas dos estudantes, sobretudo em situações nas quais o aluno já havia compreendido o problema conceitual e buscava apoio mais direto para a execução ou validação de uma solução. Em tais casos, a insistência em perguntas reflexivas pode ser interpretada como um obstáculo ao progresso, interrompendo o fluxo da atividade e afetando negativamente a experiência de uso.

Esse resultado evidencia um desafio recorrente em abordagens baseadas em andaimes e mediação, especialmente quando mediadas por sistemas automatizados. A eficácia da intervenção depende não apenas do tipo de estratégia adotada, mas também de sua adequação ao momento da aprendizagem e ao nível de autonomia do estudante. Quando a mediação não se ajusta dinamicamente à ZDP do aluno, o suporte pode deixar de ser facilitador e passar a

ser percebido como excessivo ou inadequado.

Essas evidências empíricas contribuíram diretamente para o refinamento do design pedagógico do AIPairCode no Estudo Final 2, no qual foram implementados ajustes no mecanismo de alternância de papéis e no equilíbrio entre questionamento reflexivo e orientação mais direta. Dessa forma, os resultados do presente estudo não apenas descrevem padrões de interação, mas também informam decisões de *redesign* da ferramenta, reforçando seu caráter investigativo e iterativo.

6.2.7 Discussão Integrada dos Resultados do Estudo Final 1

Os resultados obtidos na comparação do ganho de aprendizagem entre os grupos revelam um aspecto relevante quando analisados à luz da teoria sociocultural de Vygotsky. Embora o grupo controle fosse composto majoritariamente por estudantes dos semestres finais do curso (7º e 9º), com maior experiência acadêmica e contato prévio com conteúdos de programação, o ganho médio observado nesse grupo foi inferior ao do grupo experimental, formado por estudantes dos semestres iniciais (1º, 3º e 5º). Esse achado sugere que a evolução observada não pode ser explicada apenas pelo nível inicial de experiência acadêmica, mas às condições de aprendizagem oferecidas ao longo da intervenção.

Essa diferença pode ser interpretada a partir do conceito de ZDP, definida como a distância entre aquilo que o estudante consegue realizar de forma independente e o que é capaz de realizar com o auxílio de um mediador.

No caso do grupo experimental, os conteúdos propostos podem ser interpretados como desafios compatíveis com o estágio de desenvolvimento dos estudantes, possibilitando avanços significativos quando acompanhados por mediação adequada. Em contraste, parte dos estudantes do grupo controle já apresentava maior familiaridade com os conteúdos abordados, o que pode ter reduzido o potencial de ganho observado.

Nesse contexto, o assistente de programação utilizado pelo grupo experimental atuou como um mediador pedagógico, oferecendo suporte contínuo, *feedback* imediato e orientação individualizada durante a resolução dos exercícios. Essa atuação se aproxima do papel desempenhado por um par de programação mais experiente, conforme descrito na literatura sobre programação em pares, favorecendo a reflexão, a identificação de erros e a construção gradual do conhecimento.

Além dos resultados de aprendizagem, os dados relacionados à motivação dos

estudantes reforçam essa interpretação. Os resultados do IMI indicaram que o grupo experimental apresentou percepções mais favoráveis nas dimensões de interesse/prazer, valor/utilidade e menor pressão/tensão, sugerindo maior engajamento com as atividades propostas. Em contraste, o grupo de controle apresentou níveis semelhantes de esforço, porém com tendência a menores índices de motivação positiva, especialmente no que se refere ao interesse e à percepção de utilidade das atividades.

Essas diferenças motivacionais parecem estar associadas, em parte, ao estágio do curso em que se encontravam os estudantes. Os alunos do grupo controle, majoritariamente matriculados nos semestres finais, apresentaram sinais de desmotivação em relação às atividades propostas, possivelmente relacionados à repetição de conteúdos já trabalhados anteriormente, à proximidade da conclusão do curso e a experiências prévias menos positivas com disciplinas de programação. Esse cenário pode ter limitado o envolvimento ativo desses estudantes com as tarefas, reduzindo o aproveitamento das oportunidades de aprendizagem ao longo do período, o que se refletiu em ganhos mais modestos e, em alguns casos, em regressão no desempenho.

Em contraste, os estudantes do grupo experimental, provenientes dos semestres iniciais, demonstraram maior engajamento e maior abertura para explorar os desafios propostos. A utilização do assistente de programação como par de programação pode estar associado a esse maior envolvimento, ao oferecer suporte contínuo, *feedback* imediato e um ambiente de interação percebido como menos julgador, fatores amplamente associados ao aumento da motivação, da persistência e da disposição para enfrentar desafios em tarefas de aprendizagem.

As percepções de usabilidade e as respostas às questões abertas também contribuem para compreender esses resultados. Embora o assistente tenha sido avaliado como um sistema de boa usabilidade, observou-se heterogeneidade nas experiências relatadas, especialmente no que se refere à postura predominantemente questionadora da IA.

A análise dos padrões de interação evidenciou que os estudantes recorreram ao assistente principalmente para planejar, validar e organizar suas soluções, enquanto a IA respondeu majoritariamente por meio de perguntas orientadoras. Esse padrão contribui para explicar tanto as avaliações positivas, associadas à promoção da reflexão, quanto às críticas, especialmente em momentos mais operacionais, nos quais alguns estudantes esperavam respostas mais diretas.

Em síntese, a triangulação dos dados indica que o impacto positivo observado no grupo experimental resulta da combinação entre mediação pedagógica, engajamento dos estudantes, natureza dos desafios propostos e formas de interação estabelecidas durante as atividades. Esses achados reforçam o potencial do uso de assistentes de programação baseados em IA como apoio ao ensino de programação, especialmente em contextos iniciais de formação.

Apesar dos resultados consistentes observados neste estudo, algumas limitações devem ser consideradas na interpretação dos achados. O delineamento quase-experimental, com grupos formados a partir de turmas distintas e sem randomização, implicou diferenças no perfil acadêmico dos participantes, especialmente no que se refere ao semestre do curso. Isso limita a atribuição causal direta e exclusiva dos efeitos observados exclusivamente ao uso do assistente de programação. Além disso, o tamanho da amostra, ainda que compatível com o contexto real de sala de aula, restringe o poder estatístico das análises e a generalização dos resultados para outros contextos e populações.

Ainda assim, essas limitações não invalidam os achados, mas delimitam seu escopo interpretativo e indicam caminhos para investigações futuras. Com o objetivo de reduzir parte dessas limitações metodológicas, foi conduzido um segundo estudo empírico, apresentado na próxima seção, no qual buscou-se trabalhar com uma única turma e organizar os grupos de forma mais homogênea a partir do desempenho no pré-teste, permitindo uma análise mais controlada do impacto do uso do assistente de programação no processo de aprendizagem.

6.3 Estudo Final 2: Aplicação do AIPairCode em uma Disciplina de C++

O Estudo Final 2 foi conduzido no segundo semestre letivo de 2025, na disciplina de Programação e Estrutura de Dados II (C++), do curso de Sistemas de Informação da Universidade Federal de Santa Maria (UFSM), campus de Frederico Westphalen. Diferentemente do Estudo Final 1, este estudo foi realizado com uma única turma, dividida em grupo de controle e grupo experimental, com o objetivo de permitir uma comparação mais controlada entre o modelo tradicional de ensino e a metodologia apoiada por uma ferramenta de IA.

A divisão dos grupos foi realizada a partir do desempenho dos estudantes no pré-teste aplicado na primeira aula da disciplina. Após a correção das avaliações, os estudantes foram ordenados em um *ranking* de notas, e a distribuição nos grupos ocorreu de forma alternada: o estudante com a maior nota foi alocado no grupo experimental, o segundo no grupo de

controle, o terceiro novamente no grupo experimental, e assim sucessivamente. Esse procedimento teve como objetivo formar grupos mais homogêneos em termos de conhecimento prévio, reduzindo possíveis vieses associados a diferenças iniciais de desempenho e fortalecendo a validade interna do estudo.

As atividades dos dois grupos ocorreram de forma concomitante, no mesmo ambiente físico e sob as mesmas condições didáticas. Ambos os grupos participaram das mesmas aulas teóricas e práticas, no mesmo laboratório de informática, ministradas pela mesma professora, com idêntica carga horária, conteúdos programáticos e conjuntos de exercícios. Dessa forma, a principal variável de diferenciação entre os grupos foi o uso do assistente de programação AIPairCode durante as atividades práticas.

No grupo experimental, os estudantes utilizaram o AIPairCode como par de programação de duas formas complementares. Durante as aulas práticas em laboratório, o assistente foi acessado por meio da extensão integrada ao Visual Studio Code, oferecendo suporte direto no ambiente de desenvolvimento. Além disso, os estudantes puderam utilizar o módulo web do AIPairCode para a realização de atividades extraclasse, permitindo a continuidade do apoio fora do ambiente presencial. O assistente foi empregado ao longo da resolução dos exercícios, oferecendo suporte ao processo de desenvolvimento, depuração e correção de código, conforme os princípios pedagógicos definidos para a ferramenta.

Com base nas percepções dos estudantes e nos padrões de interação identificados no Estudo Final 1, foram realizados ajustes no *prompt* que orienta o comportamento do assistente antes da condução deste segundo estudo. Na etapa anterior, alguns estudantes relataram que a postura predominantemente socrática da IA, caracterizada por respostas baseadas em perguntas reflexivas, por vezes gerava interações excessivamente longas ou pouco objetivas em determinadas situações.

A análise dessas percepções motivou uma revisão do *prompt* do sistema, buscando equilibrar o uso de perguntas orientadoras com explicações mais diretas e instruções passo a passo, especialmente em momentos mais operacionais da resolução de problemas. Esse ajuste teve como objetivo manter os princípios pedagógicos baseados na ZDP e no método socrático, ao mesmo tempo em que tornava as interações mais adequadas às necessidades práticas dos estudantes.

No grupo de controle, as atividades práticas foram realizadas segundo a metodologia tradicional da disciplina, sem o uso do assistente de programação. Nesses casos, os estudantes

contaram exclusivamente com o apoio da professora durante as aulas presenciais e com os materiais didáticos disponibilizados na disciplina para o estudo extraclasse.

Ao longo do estudo, foram coletados dados quantitativos e qualitativos, incluindo o desempenho dos estudantes em testes diagnósticos (pré e pós-teste), registros de interação com o assistente de programação no grupo experimental e questionários de percepção, motivação e usabilidade. Essa configuração metodológica permitiu investigar de forma mais controlada o potencial do assistente como apoio ao desenvolvimento do raciocínio lógico, da resolução de problemas e da aprendizagem de programação no contexto da linguagem C++ e das estruturas de dados.

O Estudo Final 2 foi conduzido entre os meses de agosto e novembro de 2025, totalizando 15 semanas letivas e uma carga horária de 120 horas/aula, correspondendo à duração integral da disciplina. A turma contou inicialmente com 32 estudantes matriculados, dos quais 2 desistiram da disciplina por reprovação por falta, resultando em 30 participantes efetivos. Dessa forma, cada grupo foi composto por 15 estudantes. A Tabela 8 apresenta o resumo da participação dos estudantes no Estudo Final 2.

Tabela 8 - Resumo da participação dos estudantes no Estudo Final 2

Grupos	**Quantidade de alunos**
Grupo Controle	15
Grupo Experimental	15
Total	**30**

Fonte: Elaborado pela autora (2026)

A caracterização detalhada da turma e dos grupos é apresentada na seção a seguir.

6.3.1 Caracterização das Turmas e Perfil dos Participantes

A caracterização da turma e dos grupos do Estudo Final 2 foi realizada a partir de dados demográficos e acadêmicos básicos, bem como do desempenho inicial dos estudantes no pré-teste aplicado no início da disciplina. Essas informações permitiram contextualizar as análises subsequentes e verificar a homogeneidade inicial entre os grupos.

Em relação ao semestre cursado, observou-se predominância de estudantes matriculados no 2º semestre do curso em ambos os grupos. No grupo experimental, todos os 15 participantes estavam no 2º semestre. No grupo de controle, 13 estudantes estavam no 2º semestre, um no 5º semestre e um no 7º semestre. Essa distribuição reflete o perfil típico da disciplina, ofertada prioritariamente para estudantes em fase inicial do curso, ainda que com a presença pontual de alunos de semestres mais avançados.

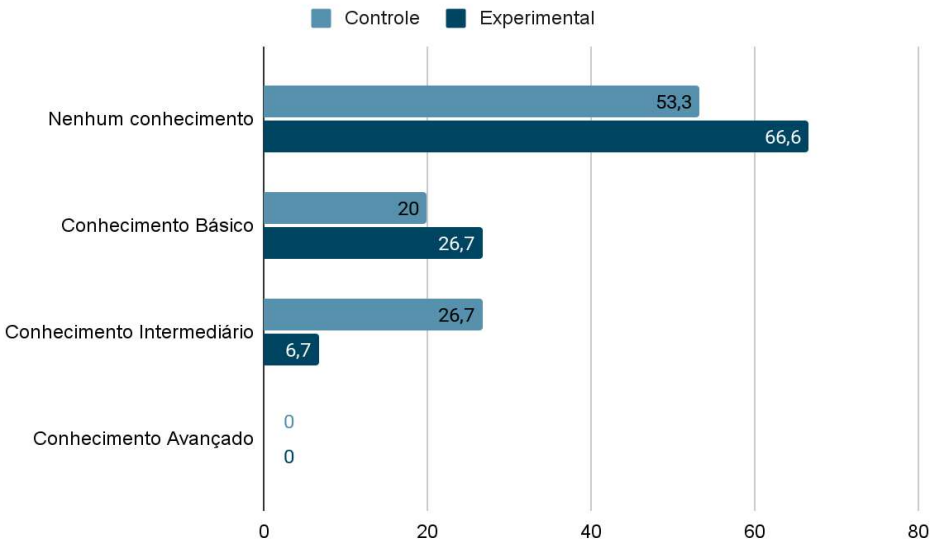
Quanto ao gênero, houve predominância do sexo masculino nos dois grupos. No grupo de controle, 13 estudantes se declararam do sexo masculino e 2 do sexo feminino. No grupo experimental, 11 estudantes se declararam do sexo masculino e 4 do sexo feminino. Embora a distribuição não seja perfeitamente equilibrada, ela é compatível com o perfil historicamente observado no curso de Sistemas de Informação.

No que se refere ao desempenho inicial no pré-teste, verificou-se variabilidade de notas em ambos os grupos, incluindo estudantes com pontuações muito baixas e outros com desempenho mais elevado. No grupo de controle, as notas variaram de 0,0 a 5,0, enquanto no grupo experimental, de 0,0 a 6,0. A estratégia de alocação alternada a partir do *ranking* de notas permitiu distribuir estudantes com diferentes níveis de conhecimento de forma equilibrada entre os grupos, reduzindo diferenças iniciais e favorecendo uma comparação mais equitativa dos resultados de aprendizagem.

De modo geral, a caracterização da turma indica que os grupos apresentaram perfis semelhantes quanto ao estágio de formação e ao conhecimento inicial em programação, reforçando a adequação do delineamento adotado para investigar o impacto do uso do assistente de programação no contexto do ensino de C++ e Estruturas de Dados.

Além do pré-teste, os estudantes responderam, na primeira aula da disciplina, a uma questão de autoavaliação sobre o grau de conhecimento prévio na linguagem de programação C++. Essa informação permitiu complementar a caracterização inicial dos grupos a partir da percepção dos próprios participantes, conforme ilustrado na Figura 55.

Figura 55 - Estudo Final 2: Autoavaliação do conhecimento prévio em C++ por grupo



Fonte: Elaborado pela autora (2026)

No grupo de controle, a maioria dos estudantes declarou não possuir conhecimento prévio em C++ (53,3%), enquanto 20,0% indicaram conhecimento básico e 26,7% relataram conhecimento intermediário. Nenhum estudante desse grupo se autodeclarou com conhecimento avançado na linguagem.

No grupo experimental, observou-se um perfil semelhante, com predominância de estudantes que declararam nenhum conhecimento prévio (66,7%), seguidos por aqueles que indicaram conhecimento básico (26,7%). Apenas um estudante (6,7%) relatou possuir conhecimento intermediário, e, assim como no grupo de controle, não houve declarações de conhecimento avançado.

Esses resultados indicam que ambos os grupos eram predominantemente compostos por estudantes iniciantes em C++, com distribuição semelhante dos níveis de conhecimento percebidos. Esse cenário reforça a adequação da estratégia de divisão dos grupos baseada no desempenho no pré-teste e sustenta a homogeneidade inicial entre as condições experimentais, aspecto fundamental para a interpretação dos resultados de aprendizagem apresentados nas seções seguintes.

6.3.2 Resultados de Aprendizagem: Análise do Pré-teste e Pós-teste

Para avaliar o desenvolvimento da aprendizagem ao longo da intervenção didática, foram aplicados dois instrumentos complementares: um pré-teste, realizado no início da

disciplina, e um pós-teste, aplicado ao final do período letivo. Esses instrumentos permitiram analisar tanto o desempenho inicial dos estudantes, a evolução do aprendizado ao longo da disciplina e comparar os resultados entre os grupos controle e experimental.

6.3.2.1 Desempenho Inicial no Pré-teste

O desempenho inicial dos estudantes foi avaliado por meio de um pré-teste aplicado no início da disciplina, com pontuação variando de 0 a 10. A Tabela 9 apresenta as estatísticas descritivas do pré-teste para os grupos controle e experimental.

Tabela 9 - Estudo Final 2: Estatística Descritiva do Pré-teste

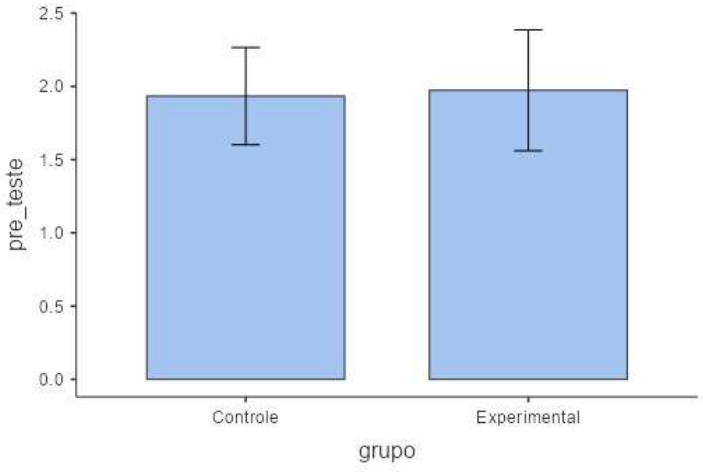
Grupo	N	Média	Mediana	Desvio-padrão	Mínimo	Máximo
Controle	15	1,93	1,80	1,29	0.00	5.00
Experimental	15	1,97	2,00	1,60	0.00	6.00

Fonte: Elaborado pela autora (2026)

No grupo de controle (n = 15), a média obtida no pré-teste foi de 1,93 (DP = 1,29), com valores variando entre 0,00 e 5,00. No grupo experimental (n = 15), a média foi de 1,97 (DP = 1,60), com valores entre 0,00 e 6,00. As estatísticas descritivas indicam desempenhos iniciais muito semelhantes entre os grupos, com médias próximas e variabilidade comparável.

A Figura 56 ilustra as médias do pré-teste por grupo, evidenciando valores muito próximos entre o grupo controle e o grupo experimental, com sobreposição dos intervalos de erro.

Figura 56 - Estudo Final 2: Desempenho médio no pré-teste por grupo



Fonte: Elaborado pela autora (2026)

Antes da comparação entre os grupos, foram avaliados os pressupostos estatísticos. A normalidade da distribuição das notas foi verificada por meio do teste de *Shapiro–Wilk*, que não indicou violação significativa da normalidade quando os grupos foram analisados separadamente (grupo controle: W = 0,918; p = 0,178; grupo experimental: W = 0,921; p = 0,201). O pressuposto de homogeneidade de variâncias, avaliado pelo teste de *Levene*, também foi atendido (F(1,28) = 0,445; p = 0,510).

Para verificar possíveis diferenças iniciais entre os grupos, foi realizado um teste *t* de *Student* para amostras independentes, que não indicou diferença estatisticamente significativa no desempenho inicial entre o grupo controle e o grupo experimental (t(28) = −0,0755; p = 0,940). Como verificador adicional, aplicou-se também o teste não paramétrico *U* de *Mann–Whitney*, que confirmou esse resultado (U = 112; p = 1,000).

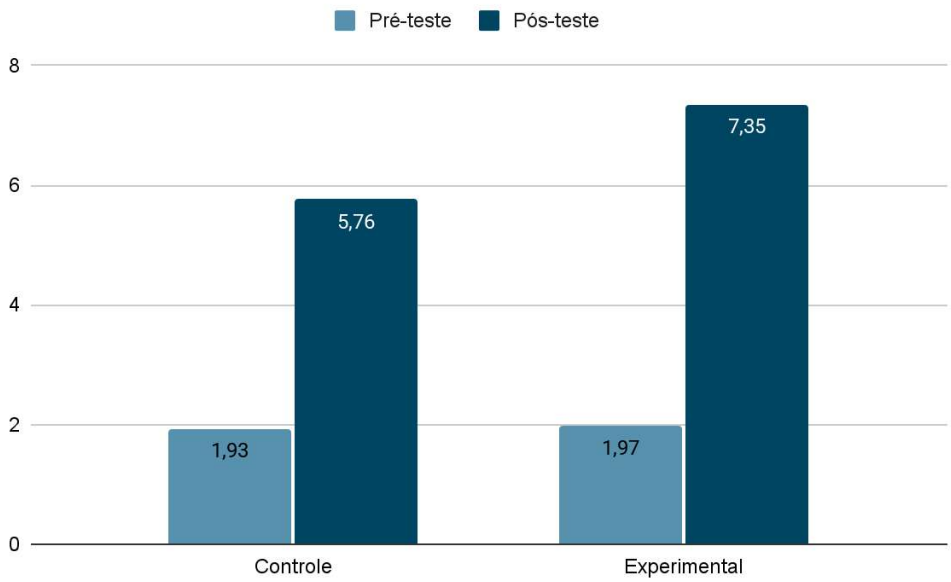
De modo geral, os resultados do pré-teste confirmam a homogeneidade inicial entre os grupos, evidenciando que a estratégia de alocação alternada a partir do *ranking* de notas foi eficaz para equilibrar o nível de conhecimento prévio dos estudantes. Esse resultado fortalece a interpretação das análises subsequentes de evolução do desempenho e de ganho de aprendizagem, uma vez que reduz a influência de diferenças iniciais entre os grupos.

6.3.2.2 Evolução do Desempenho ao Longo da Intervenção

A evolução do desempenho dos estudantes ao longo da intervenção foi analisada por meio da comparação entre os resultados do pré-teste e do pós-teste em cada grupo, utilizando testes *t de Student* para amostras emparelhadas. Essa análise teve como objetivo verificar se houve progresso significativo no desempenho dos estudantes ao longo da disciplina, independentemente da comparação direta entre os grupos. Também foram calculadas, por meio de estatística descritiva, a média e o desvio padrão para cada grupo no pré-teste e no pós-teste.

A Figura 57 ilustra a evolução do desempenho médio dos estudantes do pré-teste para o pós-teste em cada grupo.

Figura 57 - Estudo Final 2: Médias do pré-teste e do pós-teste nos grupos controle e experimental



Fonte: Elaborado pela autora (2026)

No grupo de controle (n = 15), observou-se aumento da média do pré-teste (M = 1,93; DP = 1,29) para o pós-teste (M = 5,76; DP = 2,45). O teste *t* de *Student* para amostras emparelhadas indicou diferença estatisticamente significativa entre os dois momentos de avaliação ($t(14) = 6,83$; $p < 0,001$). A verificação do pressuposto de normalidade não indicou violação (*Shapiro–Wilk*, $p = 0,911$), validando a aplicação do teste paramétrico. O tamanho de efeito, estimado por meio do *d* de *Cohen*, foi elevado (d = 1,76; IC95% [0,93; 2,57]), indicando impacto expressivo no desempenho dos estudantes do grupo de controle. Esses resultados evidenciam que, mesmo sem o uso do assistente de programação, os estudantes desse grupo apresentaram avanço significativo ao longo da disciplina, possivelmente associado à progressão natural do conteúdo e às atividades pedagógicas desenvolvidas durante o semestre.

No grupo experimental (n = 15), que utilizou o assistente de programação AIPairCode como par de programação durante as atividades em laboratório e também em atividades extraclasse, observou-se uma evolução ainda mais expressiva no desempenho. A média do pré-teste (M = 1,97; DP = 1,60) aumentou consideravelmente no pós-teste (M = 7,35; DP = 1,96). O teste *t* de *Student* para amostras emparelhadas indicou diferença estatisticamente significativa entre os dois momentos de avaliação ($t(14) = 10,80$; $p < 0,001$). A verificação do pressuposto de normalidade das diferenças, realizada por meio do teste de *Shapiro–Wilk*, não

indicou violação (W = 0,962; p = 0,721), justificando a aplicação do teste paramétrico. O tamanho de efeito foi muito elevado (d = 2,80; IC95% [1,65; 3,93]), indicando impacto substancial no desempenho dos estudantes do grupo experimental.

De forma geral, ambos os grupos apresentaram avanço significativo ao longo da disciplina, evidenciando a efetividade do processo formativo nos dois os contextos. Como foram constituídos de maneira homogênea, com base nas notas do pré-teste, partindo de níveis iniciais equivalentes de conhecimento, as diferenças observadas nas médias do pós-teste e nos tamanhos de efeito sugerem uma evolução mais expressiva do grupo experimental. Esses resultados reforçam a importância da análise comparativa do ganho de aprendizagem apresentada na seção seguinte, como estratégia para avaliar de forma mais controlada o impacto do uso do assistente de programação no desenvolvimento dos estudantes.

6.3.2.3 Comparação do Ganho de Aprendizagem entre os Grupos

Para avaliar de forma comparativa o impacto da intervenção nos dois grupos, foi calculado o ganho de aprendizagem individual, definido como a diferença entre a nota do pós-teste e a nota do pré-teste (ganho = pós − pré). Essa medida permite analisar a evolução do desempenho ao longo da disciplina, considerando o progresso individual dos estudantes e reduzindo o efeito de possíveis diferenças iniciais entre os participantes. As notas do pré-teste, pós-teste e ganho de aprendizagem de cada aluno estão disponíveis no Apêndice VII.

Inicialmente, foram calculadas as estatísticas descritivas do ganho de aprendizagem para os grupos controle e experimental, conforme apresentado na Tabela 10.

Tabela 10 - Estudo Final 2: Estatísticas descritivas do ganho de aprendizagem por grupo

								Shapiro-Wilk	
	grupo	**N**	**Média**	**Mediana**	**Desvio-padrão**	**Mínimo**	**Máximo**	**W**	**p**
ganho	Controle	15	3,83	3,80	2,17	- 0,50	7.30	0,974	0,911
	Experimental	15	5,37	6,00	1,92	1,20	8,40	0,962	0,721

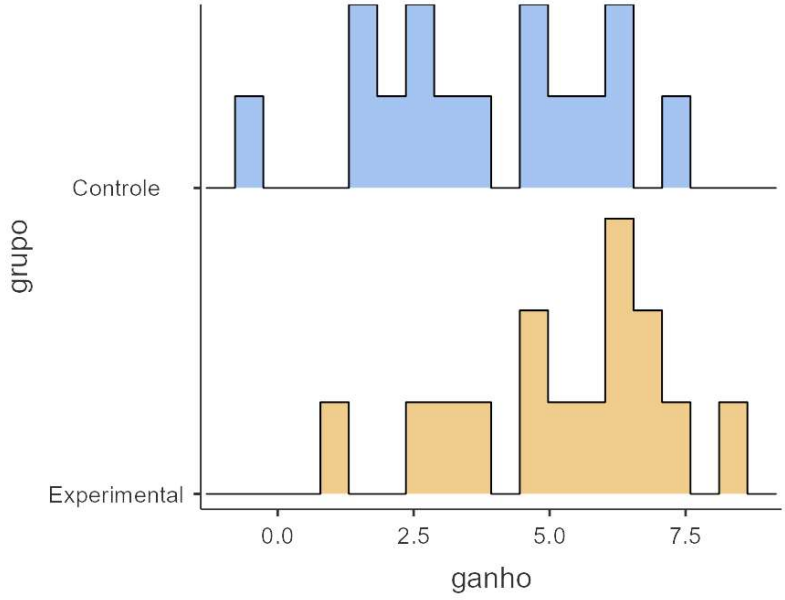
Fonte: Elaborado pela autora (2026)

O grupo controle (n = 15) apresentou ganho médio de 3,83 pontos (DP = 2,17), com maior dispersão dos resultados. O grupo experimental (n = 15) apresentou ganho médio de 5,37 pontos (DP = 1,92), indicando evolução mais acentuada. Além da média superior, o

grupo experimental apresentou mediana mais elevada (6,00), sugerindo que a maioria dos estudantes desse grupo obteve ganhos substanciais ao longo da disciplina.

A Figura 58 apresenta a distribuição do ganho de aprendizagem nos dois grupos. Visualmente, observa-se maior concentração dos ganhos do grupo experimental em faixas mais elevadas, enquanto o grupo controle apresenta maior dispersão e maior incidência de ganhos moderados, incluindo um caso de ganho negativo.

Figura 58 - Estudo Final 2: Distribuição do ganho de aprendizagem nos grupos controle e experimental



Fonte: Elaborado pela autora (2026)

Antes da comparação entre os grupos, foram verificados os pressupostos estatísticos. O teste de *Shapiro–Wilk* indicou distribuição aproximadamente normal do ganho de aprendizagem em ambos os grupos (Controle: W = 0,974; p = 0,911; Experimental: W = 0,962; p = 0,721), não havendo indícios de violação do pressuposto de normalidade. O teste de Levene também não indicou violação do pressuposto de homogeneidade de variâncias (F = 0,520; p = 0,477), indicando adequação para a aplicação de testes paramétricos.

Para comparar o ganho médio entre os grupos, foi realizado um teste *t* de *Student* para amostras independentes. Os resultados indicaram diferença estatisticamente significativa, embora próxima ao limiar convencional de significância, em favor do grupo experimental (t(28) = –2,07; p = 0,048), com tamanho de efeito de magnitude moderada a alta (*d de Cohen* = –0,755). Esse resultado sugere que os estudantes que utilizaram o assistente de programação apresentaram, em média, maior ganho de aprendizagem ao longo da disciplina.

De modo geral, ambos os grupos apresentaram avanço no desempenho, porém o grupo que utilizou o AIPairCode como par de programação obteve ganho médio superior. Considerando que, neste estudo, os grupos foram constituídos de forma homogênea a partir do desempenho no pré-teste, partindo de níveis iniciais equivalentes de conhecimento, essa diferença pode ser interpretada, de forma mais controlada, como compatível com o efeito do uso do assistente de programação.

Entretanto, devido ao tamanho relativamente reduzido da amostra e à proximidade do valor de significância ao limiar adotado, os resultados devem ser interpretados com cautela. Ainda assim, o padrão observado é consistente com a hipótese de que o suporte contínuo e a mediação individualizada oferecidos pelo AIPairCode podem ter contribuído para uma progressão mais consistente do raciocínio lógico e da capacidade de resolução de problemas.

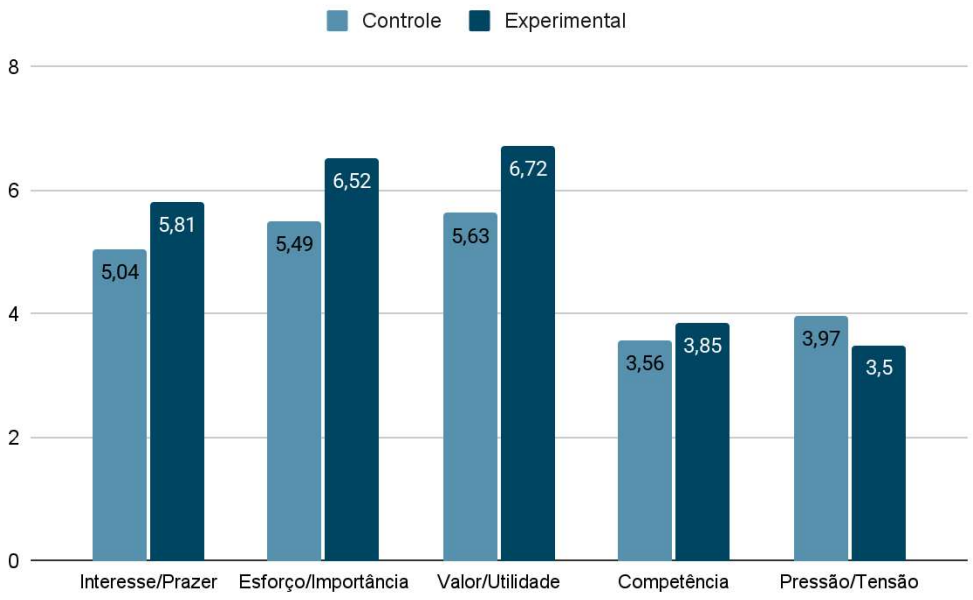
6.3.3 Motivação e Percepções dos Estudantes (IMI)

Com o objetivo de analisar os aspectos motivacionais e as percepções dos estudantes em relação às atividades desenvolvidas ao longo da disciplina, foi aplicado, ao final da intervenção didática, o Inventário de Motivação Intrínseca (IMI). O instrumento permitiu avaliar dimensões relacionadas à experiência subjetiva dos estudantes durante o processo de aprendizagem, incluindo interesse/prazer, esforço/importância, valor/utilidade, competência percebida e pressão/tensão.

O questionário foi respondido por 30 estudantes, sendo 15 do grupo controle e 15 do grupo experimental, possibilitando a análise comparativa dos diferentes aspectos motivacionais associados às atividades realizadas. As respostas, obtidas em escala *Likert* de 1 a 7, foram analisadas por meio de estatísticas descritivas, permitindo identificar tendências nas diferentes dimensões avaliadas.

A Figura 59 apresenta as médias obtidas em cada dimensão do IMI, possibilitando uma visualização comparativa entre os grupos.

Figura 59 - Estudo Final 2: Gráfico da média do IMI por dimensão



Fonte: Elaborado pela autora (2026)

No contexto do Estudo Final 2, a análise do IMI assume particular relevância em função do delineamento adotado. Como os grupos de controle e experimental foram constituídos de forma homogênea a partir do desempenho no pré-teste e participaram das mesmas condições didáticas ao longo do semestre, diferenciando-se apenas pelo uso do assistente de programação AIPairCode no grupo experimental, as diferenças observadas nas percepções e níveis de motivação podem ser interpretadas de maneira mais controlada, reduzindo a influência de fatores externos à intervenção.

Na dimensão Interesse/Prazer, o grupo experimental apresentou valores superiores aos do grupo controle. As estatísticas descritivas indicaram média de 5,04 (DP = 1,37) no grupo controle e 5,81 (DP = 1,00) no grupo experimental. A verificação da normalidade indicou distribuição aproximadamente normal em ambos os grupos (*Shapiro–Wilk*: controle p = 0,453; experimental p = 0,259). Ainda assim, optou-se pela utilização do teste não paramétrico U de *Mann–Whitney,* a fim de manter consistência metodológica com outras dimensões do instrumento que apresentaram violações de pressupostos.

Os resultados indicaram diferença não estatisticamente significativa, embora próxima ao limiar convencional de significância (U = 74,0; p = 0,114), com tamanho de efeito moderado (correlação bisserial de ordens r = 0,34). Esse resultado sugere uma tendência de maior interesse e prazer nas atividades entre os estudantes do grupo experimental, possivelmente associada à interação com o assistente de programação AIPairCode.

Na dimensão Esforço/Importância, o grupo experimental apresentou média superior à do grupo controle. O grupo controle obteve média de 5,49 (DP = 1,07) enquanto o grupo experimental apresentou média de 6,52 (DP = 0,55). A verificação da normalidade por meio do teste de *Shapiro–Wilk* indicou possível violação desse pressuposto no grupo experimental (p = 0,009), justificando a adoção de um teste não paramétrico para a comparação entre os grupos. A análise realizada por meio do teste *U* de *Mann–Whitney* indicou diferença estatisticamente significativa entre os grupos (U = 49,0; p = 0,008), acompanhada de tamanho de efeito moderado a elevado (correlação bisserial de ordens r = 0,56).

Esses resultados sugerem que os estudantes do grupo experimental atribuíram maior esforço e importância às atividades realizadas ao longo da disciplina, possivelmente em função do suporte contínuo e da mediação oferecida pelo assistente de programação AIPairCode durante a resolução dos exercícios.

Na dimensão Valor/Utilidade, observou-se uma diferença mais acentuada entre os grupos. O grupo experimental apresentou média de 6,72 (DP = 0,36), enquanto o grupo controle obteve média de 5,63 (DP = 1,37). A verificação de normalidade por meio do teste de *Shapiro–Wilk* indicou indícios de não normalidade no grupo experimental (p = 0,002), reforçando a adoção de um teste não paramétrico para a comparação entre os grupos. A análise realizada por meio do teste U de *Mann–Whitney* indicou diferença estatisticamente significativa em favor do grupo experimental (U = 54,5; p = 0,014), com tamanho de efeito moderado a elevado (correlação bisserial de ordens r = 0,52).

Esses resultados indicam que os estudantes do grupo experimental atribuíram maior valor e utilidade às atividades realizadas, possivelmente em função do suporte oferecido pelo AIPairCode durante o processo de resolução dos exercícios.

Na dimensão Competência Percebida, os resultados foram semelhantes entre os grupos. O grupo controle apresentou média de 3,56 (DP = 1,56) e o grupo experimental média de 3,85 (DP = 1,89), com variabilidade elevada em ambos os casos. A comparação por meio do teste U de *Mann–Whitney* não indicou diferença estatisticamente significativa (U = 103,5; p = 0,724), com tamanho de efeito pequeno (correlação bisserial de ordens r = 0,08). Esses resultados indicam ausência de evidência de diferença na percepção de competência entre os grupos ao final da intervenção.

Na dimensão Pressão/Tensão, o grupo experimental apresentou média inferior à do grupo controle, indicando tendência de menor pressão percebida durante as atividades. O

grupo controle obteve média de 3,97 (DP = 1,63), enquanto o grupo experimental apresentou média de 3,50 (DP = 1,88). A comparação entre os grupos por meio do teste U de *Mann–Whitney* não indicou diferença estatisticamente significativa (U = 100,0; p = 0,617), com tamanho de efeito pequeno (correlação bisserial de ordens r = –0,11). Embora as estatísticas descritivas apontem níveis ligeiramente menores de pressão no grupo experimental, essa diferença não foi confirmada estatisticamente.

De forma geral, os resultados do IMI no Estudo Final 2 indicam diferenças relevantes entre os grupos controle e experimental, especialmente nas dimensões associadas ao engajamento nas atividades e à percepção de valor do processo de aprendizagem. Considerando que os grupos foram constituídos de maneira homogênea e participaram de condições didáticas equivalentes, as diferenças observadas podem ser interpretadas de forma mais controlada em relação ao uso do assistente de programação AIPairCode.

As dimensões Esforço/Importância e Valor/Utilidade apresentaram diferenças estatisticamente significativas em favor do grupo experimental, com tamanhos de efeito de magnitude moderada a elevada. Esses resultados indicam que os estudantes que utilizaram o AIPairCode atribuíram maior relevância às atividades propostas e demonstraram maior disposição para investir esforço na resolução das tarefas. Esse padrão sugere que o uso do assistente pode ter contribuído para tornar o processo de aprendizagem mais significativo, possivelmente ao oferecer suporte contínuo e reduzir dificuldades operacionais durante a programação.

Na dimensão Interesse/Prazer, observou-se uma tendência favorável ao grupo experimental, com médias mais elevadas e tamanho de efeito moderado. Entretanto, essa diferença não atingiu significância estatística, indicando que, embora o uso do assistente possa ter favorecido maior interesse nas atividades, esse efeito não se manifestou de forma consistente em toda a amostra.

Por outro lado, nas dimensões Competência Percebida e Pressão/Tensão, não foram identificadas diferenças estatisticamente significativas entre os grupos. A ausência de diferença em competência percebida sugere que os estudantes de ambos os grupos avaliaram suas habilidades de forma semelhante ao final da disciplina, possivelmente em função do progresso geral proporcionado pelas atividades didáticas como um todo. Já em relação à pressão/tensão, embora o grupo experimental tenha apresentado valores menores, os resultados indicam que o uso do assistente não foi suficiente para produzir uma redução consistente desse aspecto emocional.

Em conjunto, os resultados do IMI indicam que o impacto do AIPairCode se manifesta principalmente em dimensões relacionadas ao engajamento e à percepção de valor das atividades, mais do que em aspectos afetivos mais amplos ou na percepção direta de competência. Esses achados sugerem que o assistente atuou principalmente como um facilitador do envolvimento e da persistência nas atividades de programação, mais do que como um fator isolado de aumento de autoconfiança dos estudantes.

Esse padrão é coerente com os resultados observados no Estudo Final 1, no qual também foram identificadas tendências favoráveis ao grupo que utilizou o assistente de programação, especialmente nas dimensões de interesse e valor/utilidade, embora sem diferenças estatisticamente significativas. No presente estudo, tais efeitos se manifestaram de forma mais evidente, com diferenças significativas nas dimensões de esforço/importância e valor/utilidade, acompanhadas de tamanhos de efeito moderados a elevados.

Essa convergência entre os estudos, aliada ao fortalecimento dos resultados no Estudo Final 2, sugere que o impacto do AIPairCode tende a se tornar mais consistente em contextos nos quais há maior controle das variáveis, como a homogeneidade dos grupos e a equivalência das condições didáticas. Dessa forma, reforça-se a interpretação de que o assistente contribui principalmente para o aumento do engajamento e da percepção de relevância das atividades, atuando como um mediador pedagógico alinhado à ZDP dos estudantes.

6.3.4 Avaliação de Usabilidade do Assistente (SUS)

A usabilidade percebida do assistente de programação AIPairCode foi avaliada por meio do *System Usability Scale* (SUS) (BROOKE, 1996), aplicado ao final do período de intervenção exclusivamente aos estudantes do grupo experimental, uma vez que apenas esses participantes utilizaram a ferramenta durante as atividades práticas de programação.

O questionário SUS é composto por dez itens em escala *Likert* de cinco pontos, alternando afirmações positivas e negativas sobre a experiência de uso do sistema (Apêndice IV). As respostas foram pontuadas de acordo com o procedimento padrão de cálculo do SUS, resultando em um escore global variando de 0 a 100, no qual valores mais elevados indicam melhor usabilidade percebida.

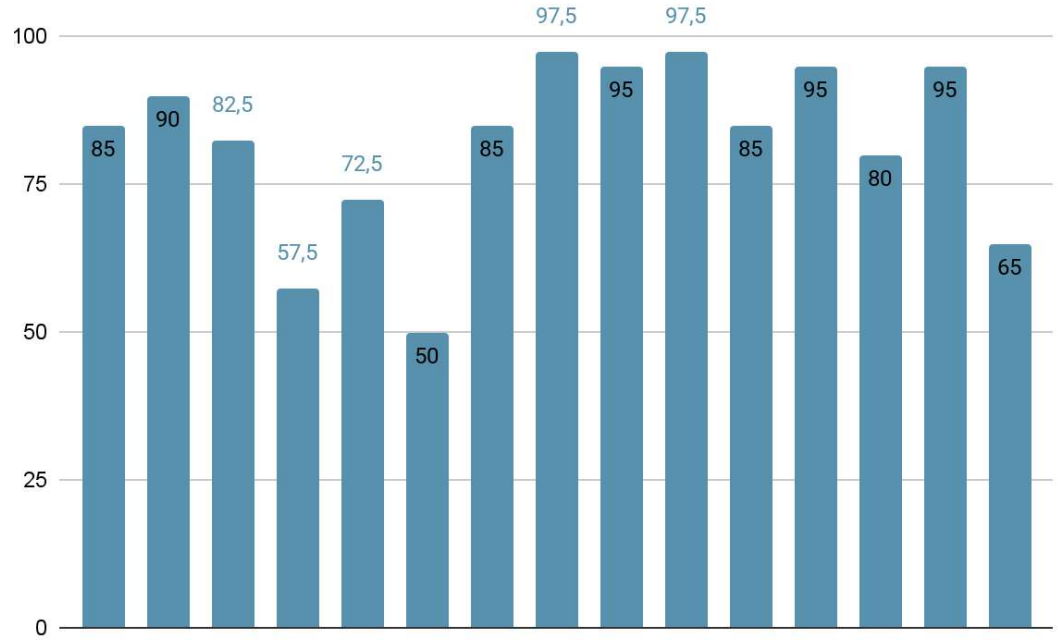
No Estudo Final 2, o AIPairCode foi disponibilizado aos estudantes por meio de dois módulos: a extensão integrada ao *Visual Studio Code*, utilizada durante as aulas práticas em laboratório, e o módulo web, empregado principalmente para a realização de atividades

extraclasse. Considerando que cada módulo apresenta características próprias de interação e contexto de uso, optou-se por aplicar o questionário SUS em dois momentos diferentes, avaliando separadamente a extensão do *Visual Studio Code* e o módulo web. Essa estratégia permitiu uma análise mais precisa e contextualizada da usabilidade percebida em cada ambiente de uso.

6.3.4.1 Usabilidade da Extensão AIPairCode no Visual Studio Code

Participaram dessa etapa 15 estudantes, correspondendo à totalidade dos participantes do grupo experimental no Estudo Final 2. A Figura 60 apresenta a distribuição dos escores totais do SUS obtidos para a extensão do AIPairCode integrada ao *Visual Studio Code*, utilizada pelos estudantes durante as aulas práticas em laboratório. O escore médio de usabilidade foi de 82,2 (DP = 14,8), com mediana de 85,0 e valores variando entre 50,0 e 97,5.

Figura 60 - Estudo Final 2: Distribuição dos escores SUS totais para a extensão do VS Code



Fonte: Elaborado pela autora (2026)

De acordo com as diretrizes de interpretação do SUS propostas por Brooke (1996) e por estudos posteriores, escores acima de 68 são geralmente considerados indicativos de boa usabilidade, enquanto valores superiores a 80 são associados a sistemas com usabilidade excelente. Nesse sentido, a média observada sugere que a extensão do AIPairCode foi percebida pelos estudantes como um sistema altamente utilizável no contexto das atividades

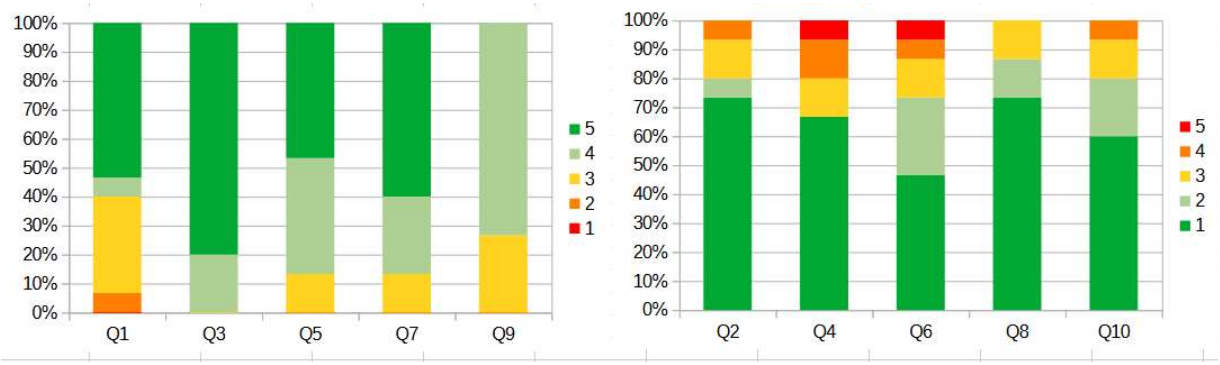
práticas em laboratório.

A distribuição dos escores revela ainda que a maior parte dos estudantes atribuiu avaliações elevadas à extensão, com concentração de valores entre 80 e 97,5, indicando uma percepção predominantemente positiva da interface, da integração com o ambiente de desenvolvimento e da facilidade de uso durante a resolução dos exercícios de programação. Embora alguns estudantes tenham atribuído escores mais baixos, esses casos foram pontuais e não comprometeram a avaliação geral do sistema.

Com o objetivo de aprofundar a compreensão sobre os aspectos específicos da experiência de uso do assistente, foi realizada uma análise das respostas item a item do questionário SUS. A Figura 61 apresenta o detalhamento das respostas por questão (Q1 a Q10), organizadas de modo a distinguir os itens formulados positivamente (Q1, Q3, Q5, Q7 e Q9), apresentados à esquerda, e os itens formulados negativamente (Q2, Q4, Q6, Q8 e Q10), apresentados à direita.

No gráfico as respostas são apresentadas em termos percentuais, agrupadas nas categorias: bem positivas (verde-escuro), positivas (verde-claro), neutras (amarelo), negativas (laranja) e bem negativas (vermelho).

Figura 61: - Estudo Final 2: Detalhamento das respostas do SUS por questão para a Extensão do VS Code



Fonte: Elaborado pela autora (2026)

De modo geral, observa-se predominância de respostas positivas e bem positivas na maior parte dos itens, reforçando a avaliação global favorável da usabilidade da extensão. Destacam-se particularmente as questões relacionadas à facilidade de uso, integração das funcionalidades e confiança no uso do sistema, nas quais a maior parte dos estudantes indicou concordância elevada. Esse padrão sugere que a extensão foi percebida como intuitiva, bem integrada ao ambiente de desenvolvimento e adequada ao contexto das atividades práticas em

laboratório.

Nos itens formulados negativamente, embora também se observa predominância de respostas favoráveis (discordância das afirmações negativas), nota-se maior dispersão em algumas questões, indicando que determinados aspectos da experiência de uso foram percebidos de forma mais heterogênea entre os estudantes. Esses resultados apontam para oportunidades de refinamento futuro, sem comprometer, contudo, a avaliação geral positiva do sistema.

6.3.4.2 Percepções dos estudantes sobre o uso da Extensão do VS Code

Além da avaliação quantitativa da usabilidade por meio do questionário SUS, os estudantes do grupo experimental responderam a três questões abertas relacionadas às suas percepções sobre o uso da extensão do AIPairCode no *Visual Studio Code*. Essas respostas permitiram aprofundar a compreensão sobre os aspectos mais valorizados da ferramenta, as principais limitações percebidas e as sugestões de melhoria apontadas pelos próprios usuários, complementando a análise estatística apresentada anteriormente.

De modo geral, as respostas à questão *"O que você mais gostou no uso do assistente?"* evidenciam uma percepção amplamente positiva em relação ao papel do AIPairCode como apoio ao aprendizado. Muitos estudantes destacaram o fato de o assistente não fornecer respostas prontas, mas orientar o raciocínio e estimular a autonomia. Esse aspecto pode ser observado em relatos como: *"explica as melhorias que podem ser feitas no código ao invés de apenas dar respostas prontas, estimulando o aluno a pensar de forma mais autônoma"* (AlunoExperimental_01) e *"ele é bom para estudos, principalmente por não dar as respostas completas"* (AlunoExperimental_03).

Outro ponto frequentemente mencionado refere-se ao apoio na identificação e compreensão de erros, aspecto recorrente em disciplinas de programação. Alguns estudantes relataram que o assistente os ajudou a perceber falhas que não identificariam sozinhos, como ilustrado pelas falas: *"me ajudou a perceber erros sutis que havia cometido"* (AlunoExperimental_03) e *"a identificação de erros no código, consegui aplicar conceitos que não conseguiria sozinho"* (AlunoExperimental_11).

Também foi valorizada a forma como o assistente auxilia na organização do raciocínio e na estruturação das soluções, especialmente em momentos de bloqueio. Exemplos dessa percepção aparecem em comentários como *"ele ajuda a pensar na estrutura"*

(AlunoExperimental_08) e *"ajuda principalmente a dar uma direção do que fazer quando fico perdido"* (AlunoExperimental_09). A integração da ferramenta ao ambiente de desenvolvimento também foi destacada, sobretudo pela possibilidade de utilizá-la diretamente no *VS Code* e interagir com trechos específicos de código, como relatado pelo estudante que mencionou *"poder selecionar uma linha ou parte do código e pedir para ele explicar em específico"* (AlunoExperimental_04).

Em relação à questão *"O que você menos gostou no uso do assistente?"*, parte dos estudantes afirmou não ter identificado aspectos negativos relevantes, indicando satisfação geral com a ferramenta. Contudo, entre aqueles que relataram limitações, destacou-se a percepção de que o assistente, em alguns momentos, foi excessivamente questionador ou pouco direto, especialmente em etapas mais operacionais das atividades. Esse aspecto aparece em comentários como *"é pouco direto, faz muitas perguntas ao usuário"* (AlunoExperimental_12), *"ele enrola demais"* (AlunoExperimental_13) e *"às vezes um pouco confuso e repetitivo, já que pergunta muitas vezes a mesma coisa"* (AlunoExperimental_10).

Além disso, alguns estudantes relataram dificuldades relacionadas à clareza das explicações e à interpretação das perguntas, mencionando situações em que não compreenderam plenamente as orientações do assistente. Exemplos incluem comentários como *"falta de clareza em algumas situações"* (AlunoExperimental_07) e *"às vezes não entendo bem o que ele quer dizer quando usa linguagem natural"* (AlunoExperimental_08). Também foram apontados problemas pontuais de natureza mais técnica, como formatação e indentação de código ou inconsistências no diálogo, conforme relatado por estudantes que mencionaram *"problemas na formatação dos códigos"* (AlunoExperimental_11) e *"inconsistência no diálogo, sem dar sequência ao que estava sendo discutido"* (AlunoExperimental_03).

As sugestões apresentadas na questão *"Que melhorias você sugeriria para o assistente?"* reforçam essas percepções. Uma demanda recorrente foi por respostas mais diretas ou exemplos mais explícitos, especialmente após várias tentativas sem sucesso. Essa expectativa aparece claramente em sugestões como *"respostas mais diretas às perguntas"* (AlunoExperimental_02), *"ser mais direto"* (AlunoExperimental_13) e *"seria interessante uma prévia de como ficaria aquele trecho do código"* (AlunoExperimental_07). Outros estudantes sugeriram melhorias na interpretação das perguntas e maior clareza nas explicações, como em *"melhorias na interpretação de perguntas"* (AlunoExperimental_04) e

“respostas mais explicadas e mais passo a passo” (AlunoExperimental_10).

Em síntese, as respostas às questões abertas indicam que os estudantes perceberam o AIPairCode como uma ferramenta relevante e alinhada ao processo de aprendizagem de programação, especialmente por estimular o raciocínio, a autonomia e a identificação de erros. Ao mesmo tempo, os relatos evidenciam a necessidade de ajustes no equilíbrio entre questionamento reflexivo e orientação mais direta, sobretudo em situações operacionais.

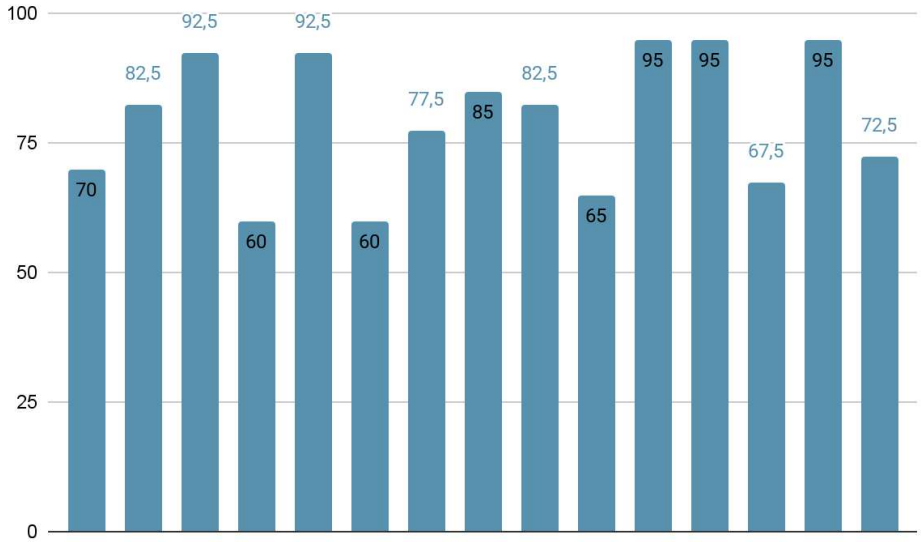
Esses achados qualitativos corroboram os resultados do SUS e dialogam com os padrões de interação observados ao longo do estudo, fornecendo subsídios importantes para o aprimoramento contínuo do design pedagógico e da experiência de uso do assistente.

6.3.4.3 Usabilidade do Módulo Web do AIPairCode

O questionário de usabilidade percebida do módulo web do AIPairCode foi respondido por 15 estudantes do grupo experimental. Diferentemente da extensão do Visual Studio Code, o módulo web foi utilizado predominantemente em atividades extraclasse, oferecendo suporte ao estudo individual e à continuidade das atividades fora do ambiente presencial.

A Figura 62 apresenta a distribuição dos escores totais do SUS atribuídos ao módulo web pelos participantes. Observou-se um escore médio de usabilidade de 79,5 pontos (DP = 13,0), com mediana de 82,5, e valores variando entre 60,0 e 95,0. Esses resultados posicionam a usabilidade do módulo web na faixa considerada boa a excelente, de acordo com as interpretações tradicionais do SUS, indicando uma avaliação global positiva por parte dos estudantes.

Figura 62 - Estudo Final 2: Distribuição dos escores SUS do módulo web do AIPairCode

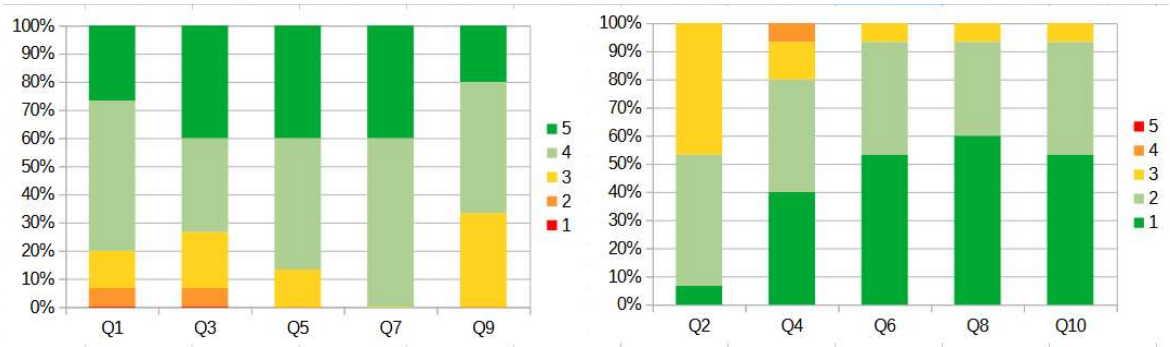


Fonte: Elaborado pela autora (2026)

De forma semelhante ao observado na extensão do VS Code, a maioria dos escores concentrou-se acima do ponto de referência de 68 pontos, geralmente adotado como limiar de aceitabilidade para sistemas interativos. A presença de escores elevados (≥ 90) em uma parcela significativa das respostas sugere que o módulo web foi percebido como intuitivo, funcional e adequado ao contexto de uso proposto.

A Figura 63 apresenta o detalhamento das respostas do SUS por questão (Q1 a Q10), organizadas de modo a distinguir os itens formulados positivamente (Q1, Q3, Q5, Q7 e Q9), apresentados à esquerda, e os itens formulados negativamente (Q2, Q4, Q6, Q8 e Q10), apresentados à direita. No gráfico as respostas são apresentadas em termos percentuais, agrupadas nas categorias: bem positivas (verde-escuro), positivas (verde-claro), neutras (amarelo), negativas (laranja) e bem negativas (vermelho).

Figura 63 - Estudo Final 2: Detalhamento das respostas do SUS por questão para o módulo web do AIPairCode



Fonte: Elaborado pela autora (2026)

Quando comparados aos resultados obtidos para a extensão do VS Code, os escores do módulo web apresentaram valores médios próximos, indicando uma percepção de usabilidade consistente entre os dois artefatos do AIPairCode, apesar das diferenças de contexto e finalidade de uso. Enquanto a extensão esteve mais associada ao suporte direto durante a codificação em laboratório, o módulo web parece ter atendido satisfatoriamente às demandas de estudo autônomo, revisão de conceitos e resolução de dúvidas fora do ambiente presencial.

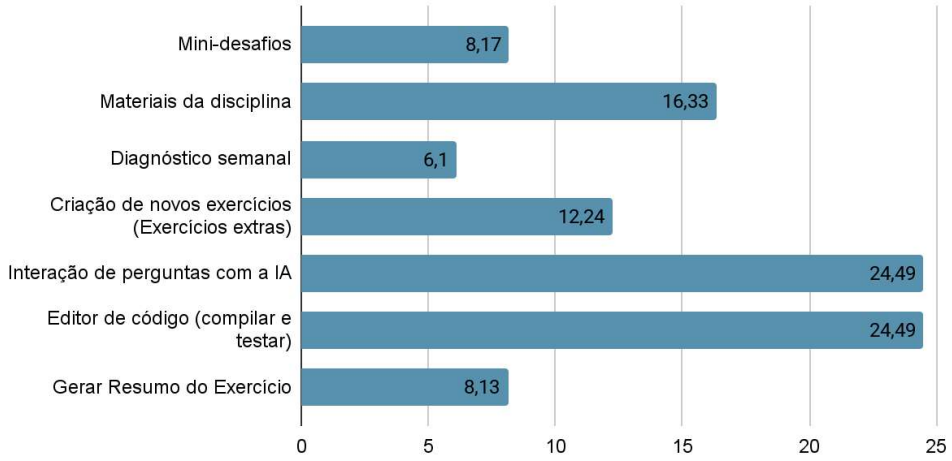
Em síntese, os resultados do SUS para o módulo web indicam que o AIPairCode apresentou níveis elevados de usabilidade percebida também no contexto extraclasse, reforçando a adequação da ferramenta em diferentes cenários de uso. Esses achados complementam a avaliação da extensão do VS Code e sustentam a proposta do AIPairCode como um assistente de programação multimodal, capaz de oferecer uma experiência de uso consistente tanto em ambientes presenciais quanto em atividades realizadas de forma autônoma pelos estudantes.

6.3.4.4 Percepções dos estudantes sobre o uso do Módulo Web

Além da avaliação da usabilidade por meio do questionário SUS, os estudantes do grupo experimental responderam a questões abertas com o objetivo de aprofundar a compreensão sobre suas percepções em relação ao uso do módulo web do AIPairCode. Inicialmente, os participantes foram questionados sobre quais recursos da ferramenta utilizaram ao longo da intervenção.

A Figura 64 apresenta a distribuição das respostas referentes aos recursos mais utilizados no módulo web. Observa-se que as funcionalidades *"Interação de perguntas com a IA"* e *"Editor de código (compilar e testar)"* foram as mais utilizadas. Esse resultado indica que o módulo web foi amplamente empregado como um ambiente ativo de apoio à resolução de problemas, permitindo aos estudantes tanto dialogar com o assistente quanto testar e validar seus códigos de forma integrada, de maneira semelhante ao uso da extensão do Visual Studio Code.

Figura 64 - Estudo Final 2: Recursos mais utilizados no módulo web



Fonte: Elaborado pela autora (2026)

Os materiais da disciplina também foram frequentemente acessados, evidenciando o uso do módulo web como espaço de consulta e revisão de conteúdos, especialmente em atividades realizadas fora do ambiente presencial. Esse achado reforça o papel do módulo web como suporte ao estudo autônomo, complementando as atividades desenvolvidas em laboratório com a extensão do VS Code.

Recursos como criação de novos exercícios e exercícios extras foram utilizados por parte significativa dos estudantes, sugerindo que o módulo web também foi explorado como ferramenta para prática adicional e consolidação dos conceitos abordados na disciplina. Por outro lado, funcionalidades como diagnóstico semanal, mini desafios e geração de resumo do exercício apresentaram menor frequência de uso, possivelmente em função do perfil das atividades propostas ou das preferências individuais dos estudantes.

De forma geral, a distribuição apresentada na Figura 59 evidencia que o módulo web foi utilizado de maneira diversificada, atendendo a diferentes objetivos pedagógicos, como esclarecimento de dúvidas, experimentação de soluções, revisão de conteúdos e prática adicional. Esses resultados corroboram a avaliação positiva de usabilidade apresentada anteriormente e indicam que o módulo web se configurou como um ambiente funcional e relevante no apoio às atividades extraclasse, complementando o uso da extensão do VS Code ao longo da disciplina.

Na questão *"O que você mais gostou no uso do assistente?"*, as respostas indicam percepções positivas principalmente quanto à facilidade de uso, acesso a materiais da disciplina, apoio na identificação de erros e recursos práticos do módulo web. O

AlunoExperimental_01 destacou: *"Sua facilidade, e poder ver conteúdos das aulas para resolver o exercício e também gerar exercícios complementares foi uma escolha muito inteligente"*. De forma semelhante, o AlunoExperimental_02 ressaltou *"A simplicidade da interface junto com as claras explicações sobre os programas da IA.",* enquanto o AlunoExperimental_03 mencionou: *"Gostei da facilidade de adaptação e das ferramentas. E poder usar o assistente na Web."*.

O apoio na identificação e compreensão de erros também foi amplamente valorizado. O AlunoExperimental_04 afirmou: *"Gostei do modo de como ela ajuda e das soluções e ajudas que o assistente dá para localizar os erros em meus programas."*. De maneira semelhante, o AlunoExperimental_06 destacou: *"A ajuda para compreender os erros no codigo, tanto de logica quanto erro de sintaxe"*.

Aspectos relacionados à usabilidade e simplicidade da interface também foram mencionados, como no relato do AlunoExperimental_05, que avaliou: *"É bem simples de utilizar, a tela não fica poluída de informações e é bem intuitivo."*. Já o AlunoExperimental_07 destacou a *"Praticidade e agilidade nas respostas."*.

Alguns estudantes também valorizaram funcionalidades específicas do módulo web, como a geração de testes e a persistência dos exercícios. O AlunoExperimental_08 apontou como diferencial *"A função de geração de entradas para teste automatizado"*, enquanto o AlunoExperimental_09 destacou *"O formato de guardar os exercícios e podemos usar de qualquer lugar"*. De maneira semelhante, o AlunoExperimental_15 reforçou o valor do teste automatizado ao afirmar: *"Saber se o trecho de código está certo sem ter que testar manualmente pois ele fazia o teste automático."*.

Além disso, alguns relatos indicam que o assistente contribuiu para orientar o estudante durante o processo de resolução. O AlunoExperimental_13 afirmou: *"Saber o que fazer depois, uma das minhas dificuldades é travar depois de fazer uma parte e não saber como prosseguir. E aprendizado de conceitos, principalmente ponteiros."*. Já o AlunoExperimental_14 destacou que o assistente tornava os erros de compilação mais compreensíveis, relatando que: *"Nos erros de compilação, deixava mais explicado qual era o erro."*.

Em relação à questão *"O que você menos gostou no uso do assistente?"*, as respostas evidenciam limitações principalmente em relação à clareza das explicações, tempo de resposta, aspectos da interface e funcionamento do módulo de testes e compilação. Alguns

estudantes mencionaram dificuldades pontuais de compreensão, como o AlunoExperimental_01, que relatou: *"Talvez a complexidade de alguns termos, precisei pesquisar sobre algumas respostas pois não entendi."*.

Aspectos da interface também foram citados. O AlunoExperimental_04 mencionou dificuldades relacionadas a elementos visuais reduzidos, enquanto o AlunoExperimental_08 relatou:: *"A janela do editor de código é ínfima."*. Problemas técnicos ligados ao processo de compilação também apareceram, como no comentário do AlunoExperimental_05, que destacou: *"A parte do compilador não aceitar o #include <windows.h> acaba sendo meio incômodo, por ter que ficar deixando comentado no código."*.

O tempo de resposta do assistente foi outro ponto sensível nas respostas. O AlunoExperimental_06 mencionou *"a demora para a resposta da IA"*, percepção reforçada pelo AlunoExperimental_11 (*"A lerdeza do assistente em responder")* e pelo AlunoExperimental_12 (*"tempo de resposta lento")*. Além disso, alguns estudantes apontaram situações em que as respostas não eram claras ou sequer retornavam, como observado pelo AlunoExperimental_07: *"Algumas respostas demoraram ou não ficaram tão claras."*, e pelo AlunoExperimental_14, que relatou: *"algumas vezes ele não respondia, sendo necessário reenviar, talvez seja culpa da internet"*.

Parte dessas percepções pode estar relacionada às características técnicas do ambiente adotado no módulo web durante o estudo. Nesse ambiente, os estudantes escreviam seus códigos em um editor integrado (Monaco Editor) e, ao acionar a funcionalidade de compilação e teste, o código era enviado a um serviço externo de compilação e execução (JDoodle). Os resultados retornados eram então encaminhados a API de IA, que os interpretava para gerar a resposta ao estudante. Esse fluxo, que envolve múltiplas etapas e a comunicação com serviços externos, pode ter contribuído para atrasos perceptíveis em algumas situações, especialmente em cenários de maior latência ou instabilidade de conexão.

Também surgiram críticas relacionadas ao módulo de testes automatizados. O AlunoExperimental_09 afirmou que *"Algumas perguntas não são validadas e consequentemente não respondidas"*, enquanto o AlunoExperimental_13 observou: *"Embora o teste gerasse entradas para testar todo o código, ele não exibe todos os testes"*. De forma semelhante, o AlunoExperimental_15 destacou que o sistema poderia fornecer maior detalhamento sobre os testes executados.

Por outro lado, alguns participantes indicaram não ter identificado aspectos negativos

relevantes, como o AlunoExperimental_03 (*"Não me vem nada em mente.")* e o AlunoExperimental_10 (*"nada em específico"),* sugerindo que, para parte dos estudantes, a experiência de uso foi amplamente satisfatória..

Na questão *"Que melhorias você sugeriria para o assistente?"*, as sugestões focaram principalmente em aperfeiçoamentos na clareza das respostas, maior agilidade no tempo de resposta, ajustes na interface do editor de código e melhorias no módulo de testes e compilação.

Algumas sugestões referem-se à forma de comunicação do assistente, indicando a necessidade de respostas mais diretas e linguagem mais objetiva. O AlunoExperimental_01 sugeriu *"mais opções de personalização, e talvez usar palavras mais sucintas"*, enquanto o AlunoExperimental_12 destacou a necessidade de *"respostas mais claras"*. De maneira semelhante, o AlunoExperimental_02 apontou a importância de um *"melhor acompanhamento passo a passo para a resolução"*, reforçando a demanda por um suporte mais guiado em determinados momentos do processo de aprendizagem.

A velocidade de resposta da IA também foi apontada como aspecto a ser aprimorado, como nas sugestões do AlunoExperimental_06 (*"Agilizar as respostas da IA"*), do AlunoExperimental_11 (*"a velocidade em que ele responde"*) e do AlunoExperimental_14 (*"acelerar a velocidade que ele responde"*). Essas respostas indicam que, embora o assistente seja considerado útil, atrasos no retorno podem comprometer a fluidez da interação durante a resolução de exercícios.

Aspectos relacionados à interface do editor de código também foram mencionados. O AlunoExperimental_03 sugeriu *"aumentar a tela de programação, às vezes quando uma linha é muito extensa fica difícil de enxergar"*, enquanto o AlunoExperimental_08 apresentou uma proposta mais detalhada: *"Aumentar a caixa do editor de código (pelo menos para 50% da tela), 25% pra terminal/console, 25% para o assistente."*. Essas sugestões reforçam a necessidade de ajustes visuais que favoreçam a leitura e a manipulação do código, especialmente em atividades mais longas.

Por fim, alguns estudantes indicaram não ter sugestões adicionais, como o AlunoExperimental_04 (*"pelo o que eu olhei nada"*) e o AlunoExperimental_13 (*"não"*), enquanto o AlunoExperimental_10 apresentou uma avaliação integradora ao afirmar que *"O módulo web é mais rápido para tirar dúvidas e testar ideias. A extensão no VS Code ajuda direto no código. Cada um tem suas vantagens e sim, eles se complementam bem."*. Esse

comentário reforça a percepção de que os dois artefatos do AIPairCode atendem a propósitos distintos e complementares no processo de aprendizagem.

De forma geral, as sugestões apresentadas pelos estudantes fornecem subsídios importantes para o refinamento do AIPairCode, especialmente no que diz respeito à usabilidade, à clareza das interações e ao desempenho técnico, alinhando-se aos resultados quantitativos obtidos por meio do SUS e às percepções observadas ao longo do estudo. Em conjunto, esses resultados reforçam que o módulo web foi percebido como um recurso relevante de apoio às atividades extraclasse, ao mesmo tempo em que evidenciam aspectos técnicos e de interações passíveis de aprimoramento em versões futuras do AIPairCode.

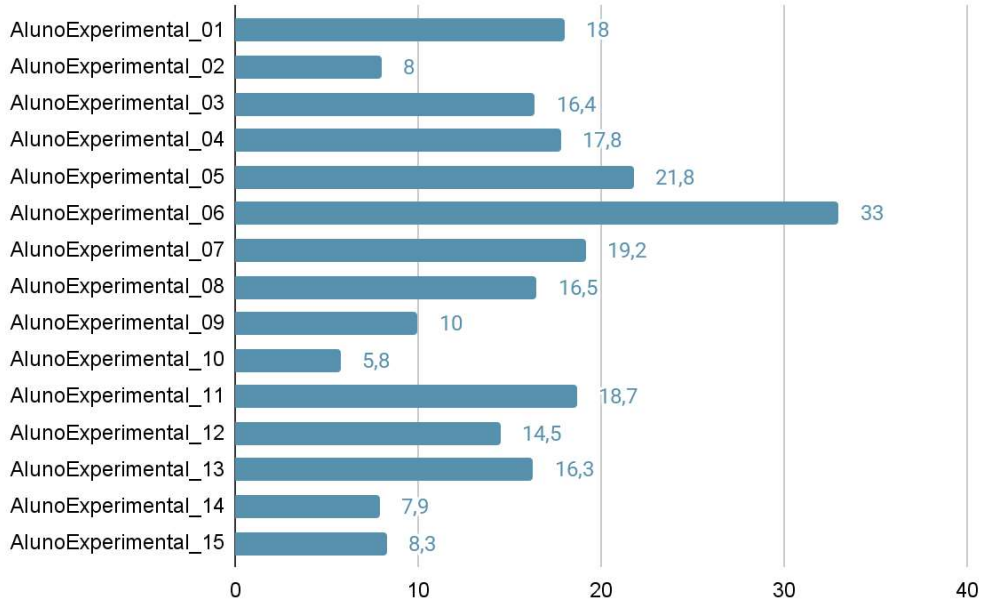
6.3.5 Padrões de interação com o AIPairCode

Com o objetivo de compreender de que forma os estudantes do grupo experimental utilizaram o AIPairCode ao longo do Estudo Final 2, esta seção apresenta a análise dos padrões de interação estabelecidos entre os participantes e o assistente de programação. As interações analisadas correspondem aos registros coletados tanto no módulo web quanto na extensão do *Visual Studio Code*, contemplando diferentes momentos do processo de aprendizagem, como a compreensão do enunciado, o planejamento da solução, a implementação do código e a identificação e correção de erros. A seguir, são apresentadas as categorias de solicitações realizadas pelos estudantes e os tipos de resposta percebidos nas interações com o AIPairCode.

6.3.5.1 Categorias de solicitação dos estudantes ao AIPairCode

Nesta seção, analisa-se o tipo de apoio buscado pelos estudantes ao interagir com o AIPairCode, a partir da categorização das solicitações registradas durante as atividades práticas. O número de interações variou consideravelmente entre os participantes, com média geral de 15,5 interações por aluno por exercício. A Figura 65 apresenta a média de interações por aluno/exercício.

Figura 65 - Estudo Final 2: Média de interações por aluno/exercício



Fonte: Elaborado pela autora (2026)

Enquanto alguns estudantes apresentaram uso intensivo da ferramenta, como o AlunoExperimental_06, com média de 33 interações por exercício, e o AlunoExperimental_05 com média de 21,8 interações, outros recorreram ao assistente com menor frequência, como o AlunoExperimental_10, que apresentou média de 5,8 interações por exercício. Esse comportamento sugere a existência de perfis distintos de engajamento com a ferramenta, possivelmente relacionados tanto ao nível de autonomia prévia quanto às dificuldades enfrentadas durante a execução das tarefas.

No que se refere ao papel assumido pela IA ao longo das interações, verificou-se predominância do papel de piloto (57,1%), seguido pelo papel de navegador (42,9%). Esse resultado sugere que os estudantes recorreram com maior frequência ao assistente durante a fase de implementação e resolução de problemas, sem, contudo, negligenciar o uso da IA para planejamento e organização do raciocínio, características associadas ao papel de navegador.

Com base na análise de conteúdo das interações, conduzida segundo os procedimentos propostos por Bardin (2016), as solicitações dos estudantes ao AIPairCode foram categorizadas nas mesmas categorias utilizadas no Estudo Final 1, acrescidas de uma nova categoria denominada “Uso do Ambiente e ferramentas”, apresentadas na Tabela 11. Essas categorias refletem diferentes estratégias adotadas pelos alunos ao utilizar a IA durante as atividades.

Tabela 11 - Estudo Final 2: Categorias de solicitação dos estudantes ao AIPairCode

Categoria	**Descrição**	**Frequência (%)**
Implementação guiada / Como fazer	Solicitação de dicas e colaboração passo a passo.	25,5
Depuração e resolução de erros	Solicitações relacionadas à identificação, compreensão ou correção de erros em código próprio.	20,7
Validação de solução	Solicitações para confirmar se uma solução, ideia ou raciocínio está correto.	18,8
Planejamento e estratégia	Solicitações relacionadas à organização do raciocínio, decomposição do problema ou definição de passos para a solução.	11,9
Geração direta de código	Solicitações explícitas por trechos de código ou soluções prontas.	10,9
Metacognição	Solicitações que expressem emoções, dificuldades pessoais ou reflexões sobre o próprio aprendizado.	8,2
Compreensão conceitual	Solicitações voltadas à compreensão de conceitos, estruturas ou princípios da linguagem ou do problema, sem pedido direto de código ou correção de erro.	2,3
Revisão e refatoração	Solicitações de avaliação, melhoria ou reorganização do código próprio já existente.	1,4
Uso do ambiente e ferramentas	Solicitações relacionadas ao uso do ambiente de desenvolvimento ou ferramentas.	0,3

Fonte: Elaborado pela autora (2026)

No conjunto das solicitações, a categoria mais frequente foi *"Implementação guiada / Como fazer"* (25,5%), indicando que os alunos utilizaram o AIPairCode principalmente para obter dicas e orientação passo a passo durante a implementação das soluções. Exemplos recorrentes desse tipo de solicitação incluíram perguntas como: *"Como faço para calcular a média de veículos por dia? soma += dia[i]??"* (AlunoExperimental_04), *"Preciso representar 10 fileiras e 12 assentos em c++ para um teatro como faço linha por linha?"* (AlunoExperimental_06) e *"Como faço para imprimir menor e maior tempo dentro do void"* (AlunoExperimental_01).

Esse padrão é coerente com o papel de mediação esperado de um par de programação, especialmente em contextos nos quais o aluno ainda não domina o processo de decomposição de problemas. Nesses casos, o suporte externo pode favorecer avanços em direção a formas mais autônomas de organização e execução das tarefas, em consonância com a ZDP.

Em seguida, destacam-se as categorias *"Depuração e resolução de erros"* (20,7%) e *"Validação de solução"* (18,8%). No caso da validação, exemplos de solicitações incluem: *"Confira se meu código está ficando certo?"* (AlunoExperimental_05), *"Dessa forma que fiz a matriz está correto?"* (AlunoExperimental_07) e *"Do jeito que eu fiz está certo de acordo com o enunciado?"* (AlunoExperimental_09). Esses resultados sugerem que muitos estudantes utilizaram o AIPairCode como um recurso para confirmar decisões, reduzir incertezas e validar raciocínios, especialmente durante etapas intermediárias da resolução de problemas.

As solicitações relacionadas à depuração e resolução de erros indicam que os estudantes recorreram ao AIPairCode para compreender falhas no código, interpretar mensagens de erro do compilador e identificar possíveis correções em trechos específicos da implementação. Em muitos casos, as interações envolviam pedidos diretos de identificação de problemas no código ou de orientação para ajustar instruções que não estavam produzindo o comportamento esperado. Exemplos desse tipo de solicitação incluem perguntas como *"Corrija o erro na linha 107"* (AlunoExperimental_01) e *"Encontre o erro e aponte as correções para o seguinte trecho de código: #include <iostream>..."* (AlunoExperimental_02).

Esse tipo de interação evidencia o uso do assistente como apoio ao raciocínio diagnóstico, uma etapa central na aprendizagem de programação. Para estudantes em níveis iniciais, a identificação da causa de um erro frequentemente exige habilidades que ainda estão em desenvolvimento, como a interpretação de mensagens do compilador ou a análise lógica do fluxo do programa. Nesse contexto, o assistente pode atuar como um recurso de andaime (*scaffolding*), orientando o estudante na investigação do problema e na construção gradual de uma solução adequada.

A categoria *"Planejamento e estratégia"* (11,9%) evidencia que parte dos estudantes utilizou o assistente nas fases iniciais da resolução dos exercícios, buscando apoio para estruturar o raciocínio antes da implementação. Esse tipo de solicitação aparece principalmente quando os alunos ainda estão construindo uma representação mental do problema, procurando compreender quais etapas ou estruturas seriam necessárias para resolvê-lo. Perguntas como *"Como posso começar esse exercício?"* ou *"Qual seria a melhor forma de estruturar esse problema?"* ilustram esse tipo de interação. Nesses casos, o AIPairCode tende a assumir um papel próximo ao de navegador, auxiliando na organização do pensamento e na definição de estratégias antes da escrita do código.

As solicitações classificadas como “Geração direta de código” (10,9%) estiveram presentes nas interações, embora em menor proporção. Nessa categoria, os estudantes solicitavam explicitamente trechos de código ou implementações completas, como nos exemplos *“faz em código esse for”* (AlunoExperimental_06) e *“me apresente o código estruturado”* (AlunoExperimental_10). Apesar dessas solicitações, a análise geral das interações indica que o uso do assistente não se concentrou na obtenção automática de soluções prontas, mas esteve majoritariamente associado a processos de construção orientada da solução. Esse padrão sugere que o AIPairCode foi utilizado predominantemente como suporte ao processo de aprendizagem, e não apenas como um mecanismo de geração de respostas.

A categoria “Metacognição” (8,2%) reúne interações nas quais os estudantes expressam estados emocionais, dificuldades ou reflexões sobre o próprio processo de aprendizagem. Exemplos incluem manifestações de incerteza, como *“Não sei fazer, preciso de ajuda”* (AlunoExperimental_02), ou expressões de satisfação após superar um desafio, como *“Consegui, muito obrigado!”* (AlunoExperimental_05). Embora menos frequentes que as solicitações técnicas, essas interações revelam que o assistente também foi utilizado como um recurso de apoio ao processo de autorregulação da aprendizagem, contribuindo para a continuidade do engajamento em momentos de bloqueio ou insegurança.

As solicitações relacionadas à “Compreensão conceitual” (2,3%) ocorreram com menor frequência e envolveram dúvidas pontuais sobre conceitos ou elementos da linguagem de programação. Exemplos incluem perguntas como *“Onde eu estruturo o endl para quebrar a linha?”* (AlunoExperimental_03) e *“Como declarar uma struct?”* (AlunoExperimental_15). A baixa incidência desse tipo de interação pode estar associada ao caráter predominantemente prático das atividades propostas, nas quais os conceitos fundamentais já haviam sido introduzidos em aula, levando os estudantes a recorrerem ao assistente principalmente para apoio na implementação e resolução de problemas.

Por fim, as categorias “Revisão e refatoração” (1,4%) e “Uso do ambiente e ferramentas” (0,3%) apresentaram baixa frequência. A reduzida ocorrência de solicitações relacionadas ao ambiente de desenvolvimento sugere que os estudantes encontraram poucas dificuldades operacionais com as ferramentas utilizadas. Da mesma forma, a baixa incidência de pedidos de refatoração indica que o foco das interações esteve predominantemente na implementação funcional e correção do código, mais do que na otimização ou reorganização estrutural das soluções.

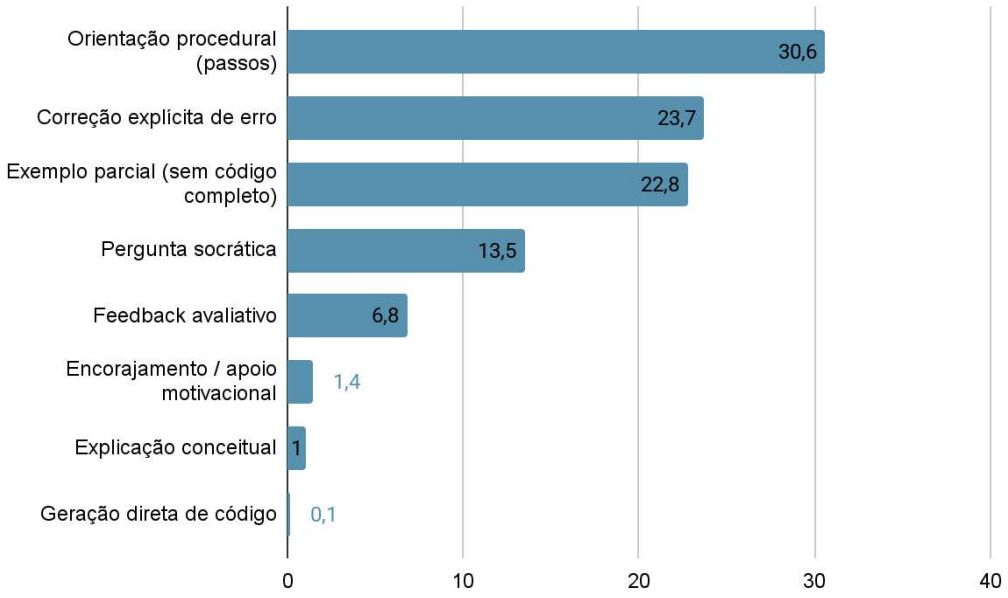
Em síntese, os resultados indicam que os estudantes utilizaram o AIPairCode principalmente como um recurso de apoio ao processo de resolução de problemas, combinando orientações para implementação, validação de soluções e depuração de erros. Esse padrão de interação é compatível com a concepção do sistema como par de programação e mediador pedagógico, oferecendo suporte contínuo ao estudante ao longo das diferentes etapas da atividade de programação.

6.3.5.2 Categorias de resposta da IA às solicitações dos estudantes

Além da análise das solicitações realizadas pelos estudantes, foi examinada também a forma como o AIPairCode respondeu às interações registradas durante o Estudo Final 2. Essa análise teve como objetivo compreender quais estratégias de mediação foram mais frequentemente utilizadas pelo assistente ao apoiar os estudantes na resolução dos exercícios de programação.

Com base na análise de conteúdo das respostas da IA, foram identificadas oito categorias de resposta, apresentadas na Figura 66. Essas categorias representam diferentes estratégias de mediação adotadas pelo assistente durante as interações, variando desde orientações passo a passo até correções diretas de erros ou estímulos à reflexão.

Figura 66 - Estudo Final 2: Distribuição percentual das categorias de resposta da IA



Fonte: Elaborado pela autora (2026)

Os resultados indicam uma predominância de respostas do tipo Orientação procedural (passos), que corresponderam a 30,6% do total de respostas analisadas. Esse dado evidencia

que, no Estudo Final 2, o AIPairCode atuou com frequência oferecendo sequências de ações, encaminhamentos e decomposição do problema, apoiando o estudante a avançar na implementação sem necessariamente entregar uma solução completa. Esse padrão sugere uma atuação fortemente orientada a andaimes, fornecendo suporte estruturado para que o aluno mantenha o fluxo de resolução do exercício.

Em seguida, destacam-se as categorias Correção explícita de erro (23,7%,) e Exemplo parcial (sem código completo) (22,8%). A incidência elevada de correções explícitas indica que o assistente foi frequentemente mobilizado em situações de depuração, nas quais o estudante apresentava erros de sintaxe, lógica, índices ou estrutura do programa. Já os exemplos parciais aparecem como uma estratégia recorrente de apoio durante a implementação, oferecendo estruturas ilustrativas e trechos incompletos que ajudam o estudante a compreender o formato esperado de uma solução, preservando sua participação ativa na finalização do código. Em conjunto, esses resultados reforçam que o suporte do AIPairCode no Estudo Final 2 esteve fortemente associado a demandas práticas de programação, especialmente em momentos de implementação e correção.

A categoria Pergunta socrática representou 13,5% das respostas, valor consideravelmente menor do que o observado no Estudo Final 1. Esse resultado sugere que, embora o questionamento reflexivo permaneça presente como estratégia pedagógica, no Estudo Final 2 a IA apresentou maior equilíbrio, priorizando respostas mais procedurais e corretivas quando necessário. Embora essa estratégia tenha sido amplamente utilizada no Estudo Final 1, observa-se que, no Estudo Final 2, sua frequência foi relativamente menor, possivelmente refletindo ajustes realizados no comportamento do assistente para tornar as respostas mais equilibradas entre questionamento e orientação direta. Ainda assim, a presença de perguntas socráticas indica que o assistente continuou promovendo momentos de reflexão, incentivo à explicitação de hipóteses e checagem de compreensão, particularmente quando o estudante precisava reorganizar o raciocínio antes de avançar.

O *Feedback* avaliativo correspondeu a 6,8% das respostas, aparecendo especialmente em situações de validação de solução, quando o aluno buscava confirmação sobre decisões de implementação ou adequação ao enunciado. Esse tipo de resposta tende a reduzir incertezas e a apoiar a autorregulação do estudante, sinalizando pontos corretos e aspectos que ainda precisam ser ajustados.

Por sua vez, Explicação conceitual apareceu em apenas 1,0% das respostas. Esse resultado sugere que as interações no Estudo Final 2 foram menos centradas em dúvidas

conceituais isoladas e mais voltadas a ações concretas de implementação, depuração e validação, o que é coerente com a natureza dos exercícios e com o perfil de solicitações observado na seção anterior.

As categorias Encorajamento / apoio motivacional (1,4%,) e Geração direta de código (0,1%, n = 1) foram pouco frequentes. A baixa incidência de geração direta de código reforça a aderência do AIPairCode aos *guardrails* do sistema, indicando que a IA não foi utilizada predominantemente como um gerador automático de soluções prontas. Já o apoio motivacional, embora minoritário, evidencia que em alguns momentos o assistente também foi mobilizado para lidar com aspectos emocionais e de persistência diante de dificuldades.

De modo geral, a distribuição das categorias de resposta indica que o AIPairCode atuou predominantemente por meio de orientações estruturadas, correções de erro e exemplos parciais, oferecendo suporte contínuo ao estudante sem substituir sua participação ativa na resolução das tarefas. Esse padrão de mediação é consistente com a proposta do sistema como um par de programação pedagógico, capaz de apoiar o desenvolvimento do raciocínio lógico e da autonomia na resolução de problemas.

6.3.6 Discussão Integrada do Estudo Final 2

Os resultados do Estudo Final 2 permitem analisar de forma integrada o impacto do uso do AIPairCode no processo de aprendizagem de programação, considerando evidências provenientes de diferentes fontes de dados, incluindo desempenho acadêmico, indicadores motivacionais, avaliações de usabilidade e padrões de interação entre os estudantes e o assistente de programação. A triangulação desses resultados contribui para uma compreensão mais ampla do papel desempenhado pela IA no contexto da disciplina de Programação e Estrutura de Dados II.

Em termos de aprendizagem, os resultados indicaram que ambos os grupos apresentaram evolução significativa entre o pré-teste e o pós-teste, evidenciando que o processo formativo desenvolvido ao longo da disciplina contribuiu para o avanço no desempenho dos estudantes.

No entanto, observou-se que o grupo experimental, que utilizou o AIPairCode como par de programação durante as atividades práticas, apresentou ganho médio de aprendizagem superior ao grupo de controle. Considerando que os grupos foram formados a partir de uma estratégia de divisão balanceada baseada no desempenho inicial no pré-teste, essa diferença

pode ser interpretada de forma mais controlada como associada ao suporte oferecido pelo assistente de programação durante a resolução dos exercícios.

Sob a perspectiva da teoria sociocultural de Vygotsky (1978), esse resultado pode ser interpretado à luz do conceito de ZDP, que descreve a distância entre aquilo que o estudante consegue realizar de forma independente e aquilo que pode realizar com o auxílio de um mediador. No contexto do estudo, o AIPairCode atuou como um mediador pedagógico capaz de oferecer suporte contínuo durante a resolução das tarefas, auxiliando os estudantes na organização do raciocínio, na identificação de erros e na construção progressiva das soluções.

Os padrões de interação observados reforçam essa interpretação. A análise das solicitações realizadas pelos estudantes revelou que as categorias mais frequentes envolveram implementação guiada, depuração de erros e validação de soluções, indicando que os alunos utilizaram o assistente principalmente para obter orientação durante o processo de resolução de problemas. Esse padrão sugere que a ferramenta foi utilizada predominantemente como apoio cognitivo para estruturar e validar o raciocínio, e não apenas como um recurso para obtenção automática de respostas.

De forma complementar, a análise das categorias de resposta do AIPairCode mostrou predominância de estratégias de mediação baseadas em orientação procedural, correção explícita de erros e apresentação de exemplos parciais, enquanto a geração direta de código ocorreu em proporção extremamente reduzida. Esse comportamento indica que o assistente manteve uma postura alinhada à proposta pedagógica do sistema, oferecendo apoio estruturado sem substituir a participação ativa do estudante na construção da solução.

Esse padrão de interação aproxima o funcionamento do AIPairCode da dinâmica observada em contextos de programação em pares, nos quais um parceiro mais experiente auxilia na resolução de problemas, oferecendo sugestões, *feedback* e orientação ao longo do processo de desenvolvimento. Ao desempenhar esse papel de forma contínua, o assistente contribuiu para reduzir bloqueios cognitivos e manter o fluxo de trabalho dos estudantes durante as atividades de programação.

Os resultados relacionados à motivação dos estudantes, analisados por meio do IMI, também oferecem evidências relevantes para a interpretação dos efeitos observados. As dimensões Esforço/Importância e Valor/Utilidade apresentaram diferenças estatisticamente significativas em favor do grupo experimental, indicando que os estudantes que utilizaram o assistente perceberam as atividades como mais relevantes e demonstraram maior disposição

para investir esforço na resolução dos exercícios. Esses resultados sugerem que o uso do AIPairCode pode ter contribuído para tornar o processo de aprendizagem mais significativo e envolvente.

Por outro lado, as dimensões Competência percebida e Pressão/Tensão não apresentaram diferenças estatisticamente significativas entre os grupos. Esse resultado pode indicar que, embora o assistente tenha contribuído para aumentar o engajamento e a percepção de utilidade das atividades, a percepção de habilidade individual dos estudantes permaneceu relativamente semelhante entre os dois grupos ao final da disciplina. Essa interpretação é consistente com o fato de que ambos os grupos tiveram acesso ao mesmo conteúdo e às mesmas atividades pedagógicas ao longo do semestre.

As avaliações de usabilidade, obtidas por meio do SUS, indicaram níveis elevados de usabilidade percebida tanto para a extensão integrada ao Visual Studio Code quanto para o módulo web do AIPairCode. Esses resultados sugerem que a interface da ferramenta não representou uma barreira significativa para a interação dos estudantes, permitindo que o foco da experiência estivesse centrado no processo de aprendizagem e na resolução dos exercícios.

As percepções qualitativas dos estudantes reforçam essa interpretação. Os relatos coletados nas questões abertas indicam que os participantes valorizaram especialmente aspectos como a orientação na identificação de erros, a possibilidade de testar soluções rapidamente e o suporte oferecido durante momentos de dificuldade. Ao mesmo tempo, foram apontadas algumas limitações relacionadas ao tempo de resposta do sistema, à clareza de determinadas explicações e a aspectos específicos da interface, fornecendo subsídios importantes para aprimoramentos futuros da ferramenta.

Outro aspecto relevante observado no Estudo Final 2 refere-se ao equilíbrio entre questionamento e orientação nas respostas da IA. No estudo anterior, parte dos estudantes haviam relatado que o assistente realizava um número elevado de perguntas socráticas, o que, em alguns momentos, era percebido como excessivamente indireto. No Estudo Final 2, ajustes realizados nas instruções e na configuração do assistente buscaram equilibrar esse comportamento, combinando questionamentos reflexivos com orientações mais diretas. A distribuição observada nas categorias de resposta da IA, com predominância de orientação procedural e menor frequência de perguntas socráticas, sugere que esse ajuste contribuiu para tornar a interação mais adequada às necessidades dos estudantes durante a resolução dos exercícios.

De forma geral, a triangulação dos resultados indica que o impacto positivo observado no grupo experimental pode ser compreendido como resultado da combinação entre mediação pedagógica, engajamento dos estudantes, usabilidade da ferramenta e padrões de interação estabelecidos ao longo das atividades. O AIPairCode atuou não apenas como um recurso tecnológico, mas como um mediador do processo de aprendizagem, oferecendo suporte contínuo para a organização do raciocínio, a resolução de problemas e a superação de dificuldades.

6.4 Síntese dos Resultados dos Estudos Empíricos

Os resultados apresentados neste capítulo mostram que o uso do assistente de programação AIPairCode pode contribuir para o processo de aprendizagem de programação quando integrado a estratégias pedagógicas adequadas e orientadas por princípios de mediação do aprendizado.

De modo geral, os estudos indicaram que os estudantes utilizaram o assistente predominantemente como apoio para compreender problemas, estruturar soluções, validar raciocínios e identificar erros durante o desenvolvimento de código, e não apenas como ferramenta de geração automática de respostas. Esse padrão de uso reforça a efetividade dos mecanismos de mediação e dos *guardrails* implementados no sistema, que incentivam a participação ativa do estudante na construção da solução.

No que se refere aos resultados de aprendizagem, observou-se, nos estudos finais, que os estudantes que utilizaram o AIPairCode apresentaram maior ganho médio de desempenho em comparação aos grupos de controle. No Estudo Final 1, esse resultado foi interpretado com cautela devido às diferenças no perfil das turmas, já no Estudo Final 2, conduzido com grupos mais homogêneos e em condições mais controladas, esse padrão se manteve, fortalecendo a evidência de que o uso do assistente pode estar associado a uma evolução mais consistente no desempenho dos estudantes ao longo da disciplina.

Em relação aos aspectos motivacionais, os resultados indicaram diferenças favoráveis ao grupo experimental, especialmente nas dimensões relacionadas ao esforço/importância e ao valor/utilidade das atividades em ambos os estudos. Esses dados sugerem que o uso do assistente contribuiu para tornar o processo de aprendizagem mais significativo e engajador, favorecendo a persistência dos estudantes na resolução de problemas, ainda que não tenha produzido diferenças consistentes em todas as dimensões avaliadas, como competência

percebida e pressão/tensão.

As análises de usabilidade e as percepções qualitativas dos estudantes indicaram que o AIPairCode foi avaliado como um sistema de boa usabilidade, tanto na extensão integrada ao ambiente de desenvolvimento quanto no módulo web. Os relatos dos estudantes evidenciam que a ferramenta foi percebida como útil para apoiar a identificação de erros, orientar o raciocínio e reduzir bloqueios durante a resolução dos exercícios, ao mesmo tempo em que apontam a necessidade de ajustes no equilíbrio entre respostas reflexivas e orientações mais diretas, bem como melhorias em aspectos técnicos e de desempenho.

Os padrões de interação observados ao longo dos estudos reforçam o papel do AIPairCode como um mediador do processo de aprendizagem. As solicitações dos estudantes concentraram-se principalmente em atividades de implementação guiada, depuração e validação de soluções, enquanto as respostas do assistente priorizaram estratégias de orientação procedural, correção de erros e apresentação de exemplos parciais, com baixa incidência de geração direta de código. Esse comportamento indica que a ferramenta atuou de forma alinhada à proposta pedagógica, promovendo o desenvolvimento do raciocínio lógico e da autonomia dos estudantes.

Em conjunto, os resultados sugerem que assistentes de programação baseados em IA, quando orientados por princípios pedagógicos claros, como mediação, andaimes e alinhamento à ZDP, podem atuar como suporte relevante ao ensino de programação. Mais do que um recurso tecnológico, o AIPairCode se configura como um elemento mediador do processo de aprendizagem, capaz de oferecer apoio contínuo e adaptado às necessidades dos estudantes ao longo da resolução de problemas.

A partir desses achados, o capítulo seguinte apresenta as considerações finais da pesquisa, discutindo suas principais contribuições, limitações e possibilidades de investigações futuras.

7 CONSIDERAÇÕES FINAIS

A presente pesquisa teve como objetivo investigar de que maneira o uso de ferramentas de Inteligência Artificial generativa, atuando como par de programação, pode impactar o processo de aprendizagem de estudantes em disciplinas introdutórias de programação, considerando seus efeitos no desempenho, na motivação e nas formas de interação durante a resolução de problemas.

A revisão de literatura evidenciou que ferramentas de IA, como assistentes de programação, possuem potencial para atuar como mediadores pedagógicos, apoiando o processo de desenvolvimento cognitivo dos estudantes por meio de *feedback* personalizado e suporte contínuo. Esse papel está alinhado à teoria da zona de desenvolvimento proximal (ZDP) de Vygotsky, bem como a metodologia de programação em pares, na qual a aprendizagem ocorre por meio da interação e da colaboração.

Entretanto, também foi identificado que ferramentas amplamente utilizadas, como o GitHub Copilot e o ChatGPT, apresentam limitações quando aplicadas diretamente no contexto educacional, especialmente com alunos iniciantes. Essas ferramentas foram treinadas em grandes volumes de código profissional, e sua aplicação direta no ensino pode resultar em respostas complexas demais para os estudantes em fase de aprendizagem básica. Diante desse cenário justificou-se o desenvolvimento do AIPairCode, um assistente de programação personalizado, projetado para atuar como par de programação com foco pedagógico, utilizando estratégias de engenharia de *prompt, guardrails* e integração com materiais didáticos por meio de técnicas de *Retrieval-Augmented Generation* (RAG).

Os resultados dos estudos empíricos realizados indicam que o uso do AIPairCode pode contribuir de forma positiva para o processo de aprendizagem de programação. Nos estudos finais, observou-se que os estudantes que utilizaram o assistente apresentaram maior ganho de aprendizagem em comparação aos grupos de controle, além de percepções mais favoráveis em relação ao valor e à importância das atividades desenvolvidas. Os resultados foram mais evidentes no Estudo Final 2, conduzido em condições mais controladas, o que fortalece a evidência de que o uso do assistente está associado a uma evolução mais consistente no desempenho dos estudantes.

No que se refere à motivação, os resultados indicaram que o uso do assistente contribuiu para tornar o processo de aprendizagem mais significativo e engajador, especialmente nas dimensões relacionadas ao valor/utilidade e ao esforço/importância.

Entretanto, os dados também evidenciam que nem todas as dimensões apresentaram diferenças significativas, como a competência percebida, indicando que o apoio fornecido pela IA nem sempre se traduz diretamente em maior percepção de domínio por parte do estudante.

As análises dos padrões de interação demonstraram que os estudantes utilizaram o AIPairCode predominantemente como ferramenta de apoio à compreensão, ao planejamento e à validação de soluções, e não apenas como gerador de respostas prontas. Esse resultado sugere que os mecanismos de mediação implementados, como a limitação de respostas completas e o uso de questionamentos orientadores, foram eficazes em incentivar a participação ativa do estudante e o desenvolvimento do raciocínio durante a resolução dos problemas.

Por outro lado, os resultados também destacam desafios importantes. Em especial, a dimensão Autonomia apresentou resultados menos favoráveis no uso da IA em comparação ao par humano, sugerindo que, em determinadas situações, o uso do assistente pode estar associado a uma menor sensação de controle por parte do estudante. Além disso, as respostas qualitativas indicaram que alguns alunos demonstraram preferência por respostas mais diretas em momentos operacionais, evidenciando a necessidade de equilíbrio entre orientação reflexiva e suporte mais objetivo.

Retomando a hipótese norteadora desta tese, os resultados obtidos sustentam a ideia de que uma ferramenta de IA generativa, quando concebida como par de programação e orientada por princípios pedagógicos, pode contribuir para a aprendizagem de programação ao oferecer suporte individualizado, *feedback* contextualizado e mediação durante a resolução de problemas. Nos estudos finais, essa hipótese foi corroborada principalmente pelos maiores ganhos de aprendizagem observados nos grupos experimentais, bem como pelas percepções positivas dos estudantes em relação à utilidade da ferramenta e pela boa avaliação de usabilidade do AIPairCode. Além disso, a análise das interações evidenciou que o assistente foi utilizado em diferentes momentos do processo de programação, incluindo compreensão do enunciado, planejamento, implementação, correção de erros e validação de soluções, reforçando seu papel como mediador do processo de aprendizagem. Assim, a hipótese foi sustentada pelos dados quantitativos de desempenho e usabilidade, bem como pelos dados qualitativos e interacionais analisados ao longo da pesquisa.

Dessa forma, os resultados desta pesquisa reforçam que o potencial educacional da IA generativa não está apenas na tecnologia em si, mas na forma como ela é integrada ao

processo de ensino e construção do conhecimento. Assistentes de programação baseados em IA, quando orientados por princípios pedagógicos claros, como mediação, andaimes e alinhamento à ZDP, podem atuar como suporte relevante ao aprendizado, promovendo maior engajamento, apoio individualizado e desenvolvimento progressivo das habilidades de programação.

Como contribuições, esta pesquisa destaca: (i) o desenvolvimento de um assistente de programação com foco pedagógico, (ii) a proposição de um modelo de interação baseado em papéis (piloto/navegador) e mediação orientada, (iii) a evidência empírica do impacto do uso da IA no ensino de programação, e (iv) a análise dos padrões de interação entre estudantes e assistentes de IA em contexto educacional.

Apesar dos resultados promissores, a pesquisa apresenta limitações, especialmente relacionadas ao tamanho das amostras e à impossibilidade de controle completo de variáveis em contextos reais de sala de aula. Essas limitações não invalidam os achados, mas indicam a necessidade de estudos adicionais, com amostras maiores e delineamentos experimentais mais controlados.

Como trabalhos futuros, destaca-se a possibilidade de aprimorar o AIPairCode, explorando mecanismos mais adaptativos de personalização, capazes de ajustar o nível de suporte com base no progresso do estudante. Além disso, a migração para novas APIs de IA, como a *Responses* API, pode ampliar as possibilidades de desenvolvimento do sistema, permitindo maior flexibilidade na gestão do contexto e na construção das respostas. Também se destaca a necessidade de investigar o uso do assistente em diferentes contextos educacionais, linguagens de programação e perfis de estudantes.

Por fim, conclui-se que o uso de IA generativa como par de programação representa uma abordagem promissora para o ensino de programação, desde que integrada a estratégias pedagógicas adequadas, que promovam a autonomia, o pensamento crítico e a participação ativa do estudante no processo de aprendizagem.

REFERÊNCIAS

ALI, Azad; WIBOWO, Kustim. ASSESSMENT OF CHATGPT-GENERATED PROGRAMMING CODE BASED ON EXERCISES IN AN INTRODUCTORY PROGRAMMING COURSE. Issues In Information Systems, 2023. Disponível em: https://iacis.org/iis/2023/2_iis_2023_203-212.pdf.

ALVIM, Ícaro Vasconcelos; BITTENCOURT, Roberto A.; DURAN, Rodrigo S.. Evasão nos Cursos de Graduação em Computação no Brasil. In: Simpósio Brasileiro de Educação em Computação (EDUCOMP), 4. , 2024, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, p. 1-11. ISSN 3086-0733. DOI: https://doi.org/10.5753/educomp.2024.237328, 2024.

AMARAL, Érico Marcelo Hoff do. Processo de Ensino e Aprendizagem de Algoritmos Integrando Ambientes Imersivos e o Paradigma de Blocos de Programação Visual. Tese de Doutorado, Universidade Federal do Rio Grande do Sul - UFRGS, 2015.

AMBROSINI, Alejandro et al. Study of the impact of ChatGPT in the teaching of introductory subjects to programming. In: Proceedings of the LACCEI international Multi-conference for Engineering, Education and Technology. [s.l.: s.n.], 2023. Disponível em: https://laccei.org/LACCEI2023-BuenosAires/all-papers/Contribution_1592_a.pdf.

AMBRÓSIO, A. P. L. *et al*. Programação de computadores: compreender as dificuldades de aprendizagem dos alunos. Revista Galego-Portuguesa de Psicoloxía e Educación, v. Vol. 19, 2011. Disponível em: http://hdl.handle.net/2183/10436

ANICIC, Katarina Pazur; STAPIC, Zlatko. Teaching Methods in Software Engineering: A Systematic Review. IEEE Software, [s. l.], v. 39, n. 6, p. 73–79, DOI: 10.1109/MS.2022.3152629, 2022.

ANTHROPIC. The Claude 3 Model Family: Opus, Sonnet, Haiku, 2024. Disponível em: https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf. .

ARAÚJO, Fabrício Viero de; FALKEMBACH, Gilse Antoninha Morgental. Experiências no Aprendizado de Algoritmos Utilizando um Ambiente de Aprendizagem. RENOTE, Porto Alegre, v. 4, n. 1, DOI: https://doi.org/10.22456/1679-1916.13865, 2006.

EADVISOR. Home | eAdvisor. [S. l.], 2024. Disponível em: https://eadvisor.asu.edu/. Acesso em: 3 maio 2024.

AVCI, Görkem et al. Intelligent tutoring systems. In: OWUSU-BOATENG (ED), Obeng. Deep Science Publishing. [S. l.]: Deep Science Publishing, p. 46–57, DOI: https://doi.org/10.70593/978-93-49307-53-7_4, 2025.

AYYAMPERUMAL, S. G.; GE, L. Current state of LLM Risks and AI Guardrails. [S. l.]: arXiv, 2024. Disponível em: https://arxiv.org/abs/2406.12934. Acesso em: 26 set. 2024.

AZAIZ, Imen; DECKARM, Oliver; STRICKROTH, Sven. AI-Enhanced Auto-Correction of Programming Exercises: How Effective is GPT-3.5? International Journal of Engineering Pedagogy (iJEP), v. 13, n. 8, p. 67–83, 2023. Disponível em: https://online-journals.org/index.php/i-jep/article/view/45621.

BACICH, Lilian; MORAN, José. Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática. Penso Editora, 2017.

BANGOR, Aaron; KORTUM, Philip T.; MILLER, James T. An Empirical Evaluation of the System Usability Scale. International Journal of Human–Computer Interaction, [s. l.], v. 24, n. 6, p. 574–594, 2008. DOI: https://www.tandfonline.com/doi/full/10.1080/10447310802205776.

BANIĆ, Bojan; KONECKI, Mario; KONECKI, Mladen. Pair Programming Education Aided by ChatGPT. In: 2023 46th MIPRO ICT and Electronics Convention (MIPRO). Opatija, Croatia: IEEE, p. 911–915, DOI: 10.23919/MIPRO57284.2023.10159727, 2023.

BARDIN, Laurence. Análise de conteúdo. São Paulo: Edições 70, 2016.

BECK L., CHIZHIK A. Cooperative learning instructional methods for CS1: Design, implementation, and evaluation. ACM Transactions on Computing Education (TOCE), 13(3), 1–21. DOI: https://doi.org/10.1145/2499947.2492686, 2013.

BECKER, Brett A.; DENNY, Paul; FINNIE-ANSLEY, James; *et al*. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. Toronto ON Canada: ACM, p. 500–506. DOI: 10.1145/3545945.3569759, 2023.

BEKIARIDIS, George. Supplement to the DigCompEDU Framework: Outlining the Skills and Competences of Educators Related to AI in Education. [S. l.]: Universität Bremen, DOI: 10.26092/elib/2708, 2024.

BENNEDSEN, Jens; CASPERSEN, Michael E. Failure rates in introductory programming. ACM SIGCSE Bulletin, [s. l.], v. 39, n. 2, p. 32–36, 2007.

BENNEDSEN, Jens; CASPERSEN, Michael E. Failure rates in introductory programming: 12 years later. ACM Inroads, [s. l.], v. 10, n. 2, p. 30–36, DOI: https://doi.org/10.1145/332488, 2019.

BERSSANETTE, J. H.; FRANCISCO, A. C. de . Metodologias Ativas de Aprendizagem no Contexto de Ensino-Aprendizagem de Programação de Computadores: uma revisão sistemática da literatura. Educitec - Revista de Estudos e Pesquisas sobre Ensino Tecnológico, Manaus, Brasil, v. 7, p. e159821, DOI: 10.31417/educitec.v7.1598, 2021.

BLOOM, Benjamin S. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. Educational Researcher, [s. l.], v. 13, n. 6, p. 4–16, 1984.

BOSSE, Yorah; GEROSA, Marco Aurélio. Reprovações e Trancamentos nas Disciplinas de Introdução à Programação da Universidade de São Paulo: Um Estudo Preliminar. Anais do Workshop sobre Educação em Computação (WEI 2015). Brasil: Sociedade Brasileira de Computação - SBC, p. 426–435. DOI: 10.5753/wei.2015.10259, 2015.

BOSSE, Yorah. Padrões de dificuldades relacionadas com o aprendizado de programação. 2020. Tese (Doutorado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, DOI:10.11606/T.45.2020.tde-14072020-172808, 2020.

BREZOLIN, Carmen Vera Scorsatto; SILVEIRA, Milene Selbach. Panorama Brasileiro de Uso de Ferramentas para Desenvolvimento do Pensamento Computacional e Ensino de Programação. Anais do XXIX Workshop sobre Educação em Computação (WEI 2021). Brasil: Sociedade Brasileira de Computação, p. 398–407, DOI: https://doi.org/10.5753/wei.2021.15931, 2021.

BROOKE, J., SUS-A quick and dirty usability scale. In: JORDAN, P. W., THOMAS, B., MCCLELLAND, I. L., et al. (Org.), Usability evaluation in industry, [S.l.], CRC Press, 1996. p. 189–194. Disponível em: https://www.crcpress.com/product/isbn/9780748404605.

CALDERON, Ivanilse; SILVA, Williamson; FEITOSA, Eduardo. Active Learning Methodologies for Teaching Programming in Undergraduate Courses: A Systematic Mapping Study. Informatics in Education, DOI: https://doi.org/10.15388/infedu.2024.11, 2023.

CARBONELL, Jaime. On Man-Computer Interaction: A Model and Some Related Issues. IEEE Transactions on Systems Science and Cybernetics, [s. l.], v. 5, n. 1, p. 16–26, DOI: 10.1109/TSSC.1969.300239, 1969.

CASPERSEN, Michael E. Principles of Programming Education. In: Computer Science Education: Perspectives on Teaching and Learning in School, p. 219-236, 2023.

CC2020, Task Force. Computing Curricula 2020: Paradigms for Global Computing Education. New York, NY, USA: ACM, 2020. Disponível em: https://dl.acm.org/doi/book/10.1145/3467967. Acesso em: 24 set. 2023.

CHANG, Chiung-Sui; CHUNG, Chih-Hung; CHANG, Julio Areck. Influence of problem-based learning games on effective computer programming learning in higher education. Educational Technology Research and Development, [s. l.], v. 68, n. 5, p. 2615–2634, DOI: https://doi.org/10.1007/s11423-020-09784-3, 2020.

CHEN, M., TWOREK, J., JUN, H., *et al*. Evaluating Large Language Models Trained on Code. DOI: 10.48550/arXiv.2107.03374, 2021.

CHEN, Bei, *et al*. CodeT: Code Generation with Generated Tests. DOI: https://doi.org/10.48550/arXiv.2207.10397, 2022.

CHEN, J. Study on Teaching Programming in Adult Colleges Based on the Cultivation of Computational Thinking. In: 2023 5th International Conference on Computer Science and Technologies in Education (CSTE). Xi'an, China: IEEE, p. 81–85, DOI: 10.1109/CSTE59648.2023.00021, 2023.

CHEN, Tianyi. The Impact of AI-Pair Programmers on Code Quality and Developer Satisfaction: Evidence from TiMi studio. In: Proceedings of the 2024 International Conference on Generative Artificial Intelligence and Information Security. New York, NY, USA: Association for Computing Machinery. p. 201–205, DOI: 10.1145/3665348.3665383, 2024.

CHO, Young Hoan; LIM, Kenneth Y. T. Effectiveness of collaborative learning with 3 D virtual worlds. British Journal of Educational Technology, [s. l.], v. 48, n. 1, p. 202–211, 2017.

CHUKHLOMIN, Valeri. Socratic Prompts: Engineered Dialogue as a Tool for AI-Enhanced Educational Inquiry. Latin American Business and Sustainability Review, [s. l.], v. 1, n. 1, p. 1–13, DOI: https://doi.org/10.70469/labsreview.v1i1.10, 2024.

CIPRIANO, Bruno Pereira; ALVES, Pedro. GPT-3 vs Object Oriented Programming Assignments: An Experience Report. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. Turku Finland: ACM, 2023, p. 61–67. Disponível em: https://dl.acm.org/doi/10.1145/3587102.3588814.

CODEGEMMA TEAM *et al*. CodeGemma: Open Code Models Based on Gemma. [S. l.]: arXiv, 2024. Disponível em: https://arxiv.org/abs/2406.11409. Acesso em: 2 maio 2025.

COELHO, Marcos Da Conceição; GUEDES, Albertina Marília Alves. Aprendizagem Baseada em Problemas aplicada à Programação de Computadores: Um Mapeamento Sistemático. RENOTE, v. 18, n. 2, p. 570–580, DOI: https://doi.org/10.22456/1679-1916.110298, 2021.

COSTA, Dóris Anita Freire. Superando limites: a contribuição de Vygotsky para a educação especial. Revista Psicopedagogia, [s. l.], v. 23, n. 72, p. 232–240, 2006. Disponível em: http://pepsic.bvsalud.org/scielo.php?script=sci_abstract&pid=S0103-84862006000300007&lng=pt&nrm=iso&tlng=pt. Acesso em: 29 abr. 2024.

CRESWELL, John W. Educational research: planning, conducting, and evaluating quantitative and qualitative research. 4. ed. Boston: Pearson, 2012.

DECI, E. L., RYAN, R. M. The" what" and" why" of goal pursuits: Human needs and the self-determination of behavior. Psychological Inquiry, 11(4), 227-268, 2000.

DENNY, P., PRATHER, J., BECKER, B. A., FINNIE-ANSLEY, J., HELLAS, A., LEINONEN, J., LUXTON-REILLY, A., REEVES, B. N., SANTOS, E. A., SARSA, S. Computing Education in the Era of Generative AI. Communications of the ACM, v. 67, n. 2, p. 56–67. DOI: https://doi.org/10.1145/3624720, 2024.

DEEPSEEK-AI et al. DeepSeek-Coder-V2: Breaking the Barrier of Closed-Source Models in Code Intelligence. [S. l.]: arXiv, 2024. Disponível em: https://arxiv.org/abs/2406.11931. Acesso em: 2 maio 2025.

DESAI, Ankur; DEO, Atul. Introducing Amazon CodeWhisperer, the ML-powered coding companion | Artificial Intelligence. [S. l.], 2022. Disponível em: https://aws.amazon.com/blogs/machine-learning/introducing-amazon-codewhisperer-the-ml-powered-coding-companion/. Acesso em: 2 maio 2024.

DILLENBOURG, Pierre. What do you mean by collaborative learning? In: Collaborative-learning: Cognitive and computational approaches, p. 1-19. Oxford: Elsevier, 1999.

DOS SANTOS, Otávio Lube; CURY, Davidson. Challenging the Confirmation Bias: Using ChatGPT as a Virtual Peer for Peer Instruction in Computer Programming Education. In: 2023 IEEE Frontiers in Education Conference (FIE). College Station, TX, USA: IEEE, p. 1 –7. DOI: 10.1109/FIE58773.2023.10343247, 2023.

DOUGHTY, Jacob et al. A Comparative Study of AI-Generated (GPT-4) and Human-crafted MCQs in Programming Education. In: Proceedings of the 26th Australasian Computing Education Conference. Sydney NSW Australia: ACM, 2024, p. 114–123. Disponível em: https://dl.acm.org/doi/10.1145/3636243.3636256.

DUNDER, Nora et al. Kattis vs ChatGPT: Assessment and Evaluation of Programming Tasks in the Age of Artificial Intelligence. In: Proceedings of the 14th Learning Analytics and Knowledge Conference. Kyoto Japan: ACM, 2024, p. 821–827. Disponível em: https://dl.acm.org/doi/10.1145/3636555.3636882.

DURAN, Rodrigo *et al*. Potential Factors for Retention and Intent to Drop-out in Brazilian Computing Programs. ACM Transactions on Computing Education, [s. l.], v. 23, n. 3, p. 1 –33, DOI: https://dl.acm.org/doi/10.1145/3607537, 2023.

ECKERDAL, Anna ; THUNÉ, Michael. Novice Java Programmers' Conceptions of "Object" and "Class", and Variation Theory. In Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (Caparica, Portugal) (ITiCSE '05). ACM, NY NY, USA, 89–93. https://doi.org/10.1145/1067445.1067473, 2005.

ELLIS, Amanda R.; SLADE, Emily. A New Era of Learning: Considerations for ChatGPT as a Tool to Enhance Statistics and Data Science Education. Journal of Statistics and Data Science Education, v. 31, n. 2, p. 128–133, DOI: https://doi.org/10.1080/26939169.2023.2223609, 2023.

FAN, Aysa et al. Exploring the Potential of Large Language Models in Generating Code-Tracing Questions for Introductory Programming Courses. In: Findings of the Association for Computational Linguistics: EMNLP 2023. Singapore: Association for Computational Linguistics, 2023, p. 7406–7421. Disponível em: https://aclanthology.org/2023.findings-emnlp.496.

FAKOUR, Hoda; IMANI, Moslem. Socratic wisdom in the age of AI: a comparative study of ChatGPT and human tutors in enhancing critical thinking skills. Frontiers in Education, [s. l.], v. 10, p. 1528603, DOI: https://doi.org/10.3389/feduc.2025.1528603, 2025.

FARREL, Joyce. Lógica e design de programação. Cengage Learning, 2010.

FAVERO, Lucile et al. Enhancing Critical Thinking in Education by means of a Socratic Chatbot. [S. l.]: arXiv, DOI: 10.48550/arXiv.2409.05511, 2024.

FENG, Zhangyin *et al.* CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In: Online: Association for Computational Linguistics, p. 1536–1547, 10.18653/v1/2020.findings-emnlp.139, 2020.

FERNANDEZ, Amanda S.; CORNELL, Kimberly A. CS1 with a Side of AI: Teaching Software Verification for Secure Code in the Era of Generative AI. In: CS1 WITH A SIDE OF AI, 2024, Portland OR USA. Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. Portland OR USA: ACM, 2024. p. 345–351. Disponível em: https://dl.acm.org/doi/10.1145/3626252.3630817. Acesso em: 4 maio 2025.

FERRAZ, Ana Paula Do Carmo Marcheti; BELHOT, Renato Vairo. Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais. Gestão & Produção, [s. l.], v. 17, n. 2, p. 421–431, 2010.

FERREIRA, Vinicius Hartmann. A Aprendizagem de Programação Apoiada pelo Modelo Social Aberto do Estudante. Tese de Doutorado, Universidade Federal do Rio Grande do Sul - UFRGS, 2019.

FERREIRA, Vinicius Hartmann; REATEGUI, Eliseo. A Aprendizagem de Programação Apoiada pelo Modelo Social Aberto do Estudante. Revista Latinoamericana de Tecnología Educativa - RELATEC, [s. l.], v. 19, n. 2, p. 83–99, DOI: 10.17398/1695-288X.19.2.83, 2020.

FINNIE-ANSLEY, James et al. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In: Proceedings of the 24th Australasian Computing Education Conference. Virtual Event Australia: ACM, 2022, p. 10–19. Disponível em: https://dl.acm.org/doi/10.1145/3511861.3511863.

FINNIE-ANSLEY, James et al. My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In: Proceedings of the 25th Australasian Computing Education Conference. Melbourne VIC Australia: ACM, 2023, p. 97–104. Disponível em: https://dl.acm.org/doi/10.1145/3576123.3576134.

FLORIDI, L.; CHIRIATTI, M. GPT-3: Its Nature, Scope, Limits, and Consequences. Minds & Machines 30, 681–694, DOI:https://doi.org/10.1007/s11023-020-09548-1, 2020.

FRANÇA, Rozelma; TEDESCO, Patrícia. Desafios e oportunidades ao ensino do pensamento computacional na educação básica no Brasil. In: I Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação, 2015, Maceió, Alagoas, Brasil. Anais [...]. Maceió, Alagoas, Brasil: [s. n.], p. 1464, 2015.

FREITAS, Tiago Carvalho et al. NLP/AI Based Techniques for Programming Exercises Generation. p. 12 pages, 622938 bytes, 2023. Disponível em: https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.ICPEC.2023.9.

GARCIA, Msc Angela Villy Cervantes; ATOCHE, César Balladares. Competencias digitales: lenguaje de programación y rendimiento académico. Ciencia Latina Revista Científica Multidisciplinar, [s. l.], v. 6, n. 1, p. 530–543, DOI: https://doi.org/10.37811/cl_rcm.v6i1.1516, 2022.

GAŠEVIĆ, Dragan; SIEMENS, George; SADIQ, Shazia. Empowering learners for the age of artificial intelligence. Computers and Education: Artificial Intelligence, [s. l.], v. 4, p. 100130, DOI: https://doi.org/10.1016/j.caeai.2023.100130, 2023.

GIL, A. C. Métodos e Técnicas de Pesquisa Social. 7. ed. São Paulo: Atlas, 2019.

GOMES, Anabela; MENDES, Antonio. A teacher's view about introductory programming

teaching and learning: Difficulties, strategies and motivations. In: IEEE Frontiers in Education Conference (FIE) Proceedings. Madrid, Spain: IEEE, p. 1–8. DOI: 10.1109/FIE.2014.7044086, 2014.

GOOGLE. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. [S. l.], 2025. Disponível em: https://arxiv.org/html/2507.06261v1. Acesso em: 2 maio 2026.

GOYAL, Harshita *et al*. Sakshm AI: Advancing AI-Assisted Coding Education for Engineering Students in India Through Socratic Tutoring and Comprehensive Feedback. ACM Journal on Computing and Sustainable Societies, [s. l.], v. 4, n. 2, p. 1–38, DOI: https://doi.org/10.1145/3788679, 2026.

GUIMARAES JUNIOR, José Carlos et al. Ética e privacidade no uso de inteligência artificial na sala de aula: como preparar os educadores?. CONTRIBUCIONES A LAS CIENCIAS SOCIALES, [s. l.], v. 18, n. 1, p. e14350, DOI: https://doi.org/10.55905/revconv.18n.1-050, 2025.

HAINDL, Philipp; WEINBERGER, Gerald. Students' Experiences of Using ChatGPT in an Undergraduate Programming Course. IEEE Access, v. 12, p. 43519–43529, 2024. Disponível em: https://ieeexplore.ieee.org/document/10478015/.

HARSLEY, R., DI EUGENIO, B., GREEN, N., & FOSSATI, D. Collaborative Intelligent Tutoring Systems: Comparing Learner Outcomes Across Varying Collaboration Feedback Strategies In Smith, B. K., Borge, M., Mercier, E., and Lim, K. Y. (Eds.). Making a Difference: Prioritizing Equity and Access in CSCL, 12th International Conference on Computer Supported Collaborative Learning (CSCL), Volume 2. Philadelphia, PA: International Society of the Learning Sciences, 2017.

HARTLEY, K.; HAYAK, M.; KO, U. H. Artificial Intelligence Supporting Independent Student Learning: An Evaluative Case Study of ChatGPT and Learning to Code. Education Sciences, [s. l.], v. 14, n. 2, p. 120, 2024. Disponível em: https://www.mdpi.com/2227-7102/14/2/120.

HELLAS, Arto et al. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In: Proceedings of the 2023 ACM Conference on International Computing Education Research V.1. Chicago IL USA: ACM, 2023, p. 93–105. Disponível em: https://dl.acm.org/doi/10.1145/3568813.3600139.

HENKLAIN, Marcelo H. Oliveira; CARVALHO, Leandro Silva Galvão De; FEITOSA, Eduardo Luzeiro. Dificuldades no aprendizado de programação: Um exame baseado em mapeamento sistemático da literatura e interpretação analítico-comportamental. Revista Brasileira de Informática na Educação, [s. l.], v. 33, p. 1484–1521, DOI: https://doi.org/10.5753/rbie.2025.5272, 2025.

HOLMES, Wayne; BIALIK, Maya; FADEL, Charles. Artificial intelligence in education. In: STÜCKELBERGER, Christoph; DUGGAL, Pavan (org.). Data ethics : building trust : how digital technologies can serve humanity. [S. l.]: Globethics Publications, p. 621–653, DOI: 10.58863/20.500.12424/4276068, 2023.

HU, Minjie; ASSADI, Tony; MAHROEIAN, Hamid. Explicitly Introducing ChatGPT into First-year Programming Practice: Challenges and Impact. In: 2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE). Auckland, New Zealand: IEEE, 2023, p. 1–6. Disponível em: https://ieeexplore.ieee.org/document/10398297/.

HUSAIN, Anas. Potentials of ChatGPT in Computer Programming: Insights from Programming Instructors. Journal of Information Technology Education: Research, v. 23, p. 002, 2024. Disponível em: https://www.informingscience.org/Publications/5240.

ILXOMOVNA, Kamalova Nilufar; AKMALOVNA, Ilyosova Mashhura. USING THE PROBLEM-BASED DESIGN METHOD IN TEACHING PROGRAMMING LANGUAGES. Academicia Globe: Inderscience Research, [s. l.], v. 4, n. 12, p. 1–7, 2023. Disponível em: https://agir.academiascience.org/index.php/agir/article/view/1455. Acesso em: 30 abr. 2024.

IKONOMOU, Maria-Nicole. Coursera announces new AI content and innovations to help HR and learning leaders drive organizational agility amid relentless disruption. 2023. Disponível em: https://blog.coursera.org/trusted-content-and-ai-innovations-to-drive-organizational-agility-for-learning-leaders/. Acesso em: 3 maio 2024.

ISHIZUE, Ryosuke et al. Improved Program Repair Methods using Refactoring with GPT Models. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. Portland OR USA: ACM, 2024, p. 569–575. Disponível em: https://dl.acm.org/doi/10.1145/3626252.3630875.

ISOTANI, S., BITTENCOURT, I. I., CHALLCO, G. C., DERMEVAL, D., MELLO, R. F. AIED Unplugged: Leapfrogging the Digital Divide to Reach the Underserved. In: International Conference on Artificial Intelligence in Education (pp. 772-779). Cham: Springer Nature Switzerland, DOI: https://doi.org/10.1007/978-3-031-36336-8_118, 2023.

JALIL, S., RAFI, S., LATOZA, T. D., MORAN, K., LAM, W. ChatGPT and software testing education: Promises and perils. In Proc. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Dublin, Ireland, p. 4130–4137, DOI: 10.1109/ICSTW58534.2023.00078, 2023.

JING, Yuhui et al. What factors will affect the effectiveness of using ChatGPT to solve programming problems? A quasi-experimental study. Humanities and Social Sciences Communications, v. 11, n. 1, p. 319, 2024. Disponível em: https://www.nature.com/articles/s41599-024-02751-w.

JONSSON, Martin; THOLANDER, Jakob. Cracking the code: Co-coding with AI in creative programming education. In: Creativity and Cognition. Venice Italy: ACM, 2022, p. 5–14. Disponível em: https://dl.acm.org/doi/10.1145/3527927.3532801.

JORDAN, Mollie; LY, Kevin; SOOSAI RAJ, Adalbert Gerald. Need a Programming Exercise Generated in Your Native Language? ChatGPT's Got Your Back: Automatic Generation of Non-English Programming Exercises Using OpenAI GPT-3.5. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. Portland OR USA: ACM, 2024, p. 618–624. Disponível em: https://dl.acm.org/doi/10.1145/3626252.3630897.

JOVANOVIC, Mladan; CAMPBELL, Mark. Generative Artificial Intelligence: Trends and Prospects. Computer, [s. l.], v. 55, n. 10, p. 107–112, DOI: 10.1109/MC.2022.3192720, 2022.

KADAR, Rozita; ABDUL WAHAB, Naemah; OTHMAN, Jamal; *et al*. A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review. International Journal of Academic Research in Progressive Education and Development, v. 10, n. 3, p. Pages 591-605, DOI: 10.6007/IJARPED/v10-i3/11100, 2021.

KARLING, Daniel Antonio; RIZZI, Claudia Brandelero; RIZZI, Rogério Luis. Avaliação Preliminar de Ambiente Online Baseado em Jogos Digitais: Um Estudo de Caso na Aprendizagem de Algoritmos. Anais do XXIX Workshop sobre Educação em Computação (WEI 2021). Brasil: Sociedade Brasileira de Computação, p. 288–297, DOI: https://doi.org/10.5753/wei.2021.15920, 2021.

KARNALIM, Oscar *et al*. Plagiarism and AI Assistance Misuse in Web Programming: Unfair Benefits and Characteristics. In: 2023 IEEE International Conference on Teaching,

Assessment and Learning for Engineering (TALE). Auckland, New Zealand: IEEE, 2023, p. 1–5. Disponível em: https://ieeexplore.ieee.org/document/10398397/.

KASNECI, Enkelejda; SESSLER, Kathrin; KÜCHEMANN, Stefan; *et al*. ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, v. 103, p. 102274, DOI: 10.1016/j.lindif.2023.102274, 2023.

KHAN ACADEMY. Khan Academy - About. 2024. Disponível em: https://www.khanacademy.org/about. Acesso em: 29 abril de 2024.

KHANMIGO. Meet Khanmigo: Khan Academy's AI-powered teaching assistant & tutor. [S. l.], 2024. Disponível em: https://www.khanmigo.ai. Acesso em: 3 maio 2024.

KIESLER, Natalie; LOHR, Dominic; KEUNING, Hieke. Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. In: 2023 IEEE Frontiers in Education Conference (FIE). College Station, TX, USA: IEEE, 2023, p. 1–5. Disponível em: https://ieeexplore.ieee.org/document/10343457/.

KIM, Jinhee; LEE, Hyunkyung; CHO, Young Hoan. Learning design to support student-AI collaboration: perspectives of leading teachers for AI in education. Education and Information Technologies, v. 27, n. 5, p. 6069–6104, 2022. Disponível em: <https://link.springer.com/10.1007/s10639-021-10831-6>. Acesso em: 20 set. 2024.

KOLB, David A. Experiential learning: experience as the source of learning and development. New Jersey: Prentice-Hall, 1984.

KOORSSE, Melisa; CILLIERS, Charmain; CALITZ, André. Programming assistance tools to support the learning of IT programming in South African secondary schools. Computers & Education, v. 82, p. 162-178, DOI: https://doi.org/10.1016/j.compedu.2014.11.020, 2015.

KWAK, Myungjae; JENKINS, Jonathan; KIM, Joobum. Adaptive Programming Language Learning System Based On Generative AI. Issues In Information Systems, 2023. Disponível em: https://iacis.org/iis/2023/3_iis_2023_222-231.pdf.

LAMATTINA, Alexandre de Araújo. Educação 4.0: transformando o ensino na era digital. Formiga, MG : Editora Union, 2023.

LAU, Sam; GUO, Philip. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In: Proceedings of the 2023 ACM Conference on International Computing Education Research V.1. Chicago IL USA: ACM, 2023, p. 106–121. Disponível em: https://dl.acm.org/doi/10.1145/3568813.3600138.

LEINONEN, Juho et al. Comparing Code Explanations Created by Students and Large Language Models. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. Turku Finland: ACM, 2023, p. 124–130. Disponível em: https://dl.acm.org/doi/10.1145/3587102.3588785.

LELLI, V., SANTOS, I. S., SOUSA, F., BRAIDE, L. Aplicação do POGIL no ensino de Computação. In: SIMPÓSIO BRASILEIRO DE EDUCAÇÃO EM COMPUTAÇÃO (EDUCOMP), 4., 2024, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, p. 224-233. DOI: https://doi.org/10.5753/educomp.2024.237541, 2024.

LEMOS, Marilza Antunes de. Uma Abordagem Baseada em Padrões Elementares para Aprendizado de Programação. Tese de Doutorado. Universidade de São Paulo, São Paulo, 2004.

LEWIS, James R.; SAURO, Jeff. Item benchmarks for the system usability scale. J. Usability Studies 13, 3 (May 2018), 158–167. 2018.

LI, Yujia *et al*. Competition-level code generation with AlphaCode. Science, [s. l.], v. 378, n.

6624, p. 1092–1097, DOI: 10.1126/science.abq1158, 2022.

LIMA, Jefferson Ribeiro De; MENEZES, Crediné Silva De. As dificuldades enfrentadas pelos estudantes na aprendizagem de programação de computadores: Uma Revisão Sistemática da Literatura. RENOTE, [s. l.], v. 22, n. 1, p. 130–140, DOI: https://doi.org/10.22456/1679-1916.141538, 2024.

LIRA, W. A. L., SANTOS NETO, P. de A. dos; OSÓRIO, L. F. M. Uma análise do uso de ferramentas de geração de código por alunos de Computação. In: SIMPÓSIO BRASILEIRO DE EDUCAÇÃO EM COMPUTAÇÃO (EDUCOMP), 4., Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, p. 63-71. DOI: https://doi.org/10.5753/educomp.2024.237427, 2024.

LIU, Jiangyue; LI, Siran. Toward Artificial Intelligence-Human Paired Programming: A Review of the Educational Applications and Research on Artificial Intelligence Code-Generation Tools. Journal of Educational Computing Research, [s. l.], v. 62, n. 5, p. 1385–1415, DOI: 10.1177/07356331241240460, 2024.

LO, Chung Kwan. What Is the Impact of ChatGPT on Education? A Rapid Review of the Literature. Education Sciences, [s. l.], v. 13, n. 4, p. 410, DOI:10.3390/educsci13040410, 2023.

LUCAS, Margarida; MOREIRA, António; TRINDADE, Anícia Rebelo. DigComp 2.2: Quadro Europeu de Competência Digital para Cidadãos: com exemplos de conhecimentos, capacidades e atitudes. [S. l.]: UA Editora, DOI: https://doi.org/10.48528/4w7y-j586, 2022.

MEC, Ministério da Educação. Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação (DCN16). Resolução CNE/CES nº 5, de 16 de novembro de 2016. 2016. Disponível em: https://www.gov.br/mec/pt-br/cne/pdf/normas-classificadas-por-assunto/diretrizes-curriculares-cursos-de-graduacao/rces005_16.pdf. Acesso em: 27/03/2023.

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, [s. l.], v. 62, n. 2, p. 77–90, DOI: 10.1109/TE.2018.2864133, 2019.

MEDEIROS, Rodrigo Pessoa; FALCÃO, Taciana Pontual; RAMALHO, Geber Lisboa. Ensino e Aprendizagem de Introdução à Programação no Ensino Superior Brasileiro: Revisão Sistemática da Literatura. Anais do Workshop sobre Educação em Computação (WEI 2020). Brasil: Sociedade Brasileira de Computação - SBC, p. 186–190. DOI: https://doi.org/10.5753/wei.2020.11155, 2020.

MELLO, Rafael Ferreira et al. Enhancing Instructors' Capability to Assess Open-Response Using Natural Language Processing and Learning Analytics. In: HILLIGER, Isabel et al. (org.). Educating for a New Future: Making Sense of Technology-Enhanced Learning Adoption. Cham: Springer International Publishing, 2022. v. 13450, p. 102–115. Disponível em: https://link.springer.com/10.1007/978-3-031-16290-9_8.

MELLO, Rafael Ferreira; FREITAS, Elyda; PEREIRA, Filipe Dwan; *et al*. Education in the age of Generative AI: Context and Recent Developments. 2023.

MISHRA, Mayank *et al*. Granite Code Models: A Family of Open Foundation Models for Code Intelligence. [S. l.]: arXiv, 2024. Disponível em: https://arxiv.org/abs/2405.04324. Acesso em: 2 maio 2025.

MOLENAAR, Inge *et al*. Artificial Intelligence and Education: Different Perceptions and Ethical Directions. In: SMUHA, Nathalie A. (org.). The Cambridge Handbook of the Law, Ethics and Policy of Artificial Intelligence. 1. ed. [S. l.]: Cambridge University Press, p.

261–282, DOI: https://doi.org/10.1017/9781009367783.017, 2025.

MOLLICK, E. R.; MOLLICK, L. Assigning AI: Seven Approaches for Students, with Prompts. SSRN Electronic Journal, [s. l.], 2023. Disponível em: https://www.ssrn.com/abstract=4475995. Acesso em: 26 set. 2024.

MORADI DAKHEL, Arghavan et al. GitHub Copilot AI pair programmer: Asset or Liability? Journal of Systems and Software, v. 203, p. 111734, 2023. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0164121223001292.

MORAIS, Ceres Germanna Braga. Ensino e aprendizagem de programação: estudo de caso no Ensino Superior. Tese de Doutorado. Escola de Engenharia. Universidade do Minho. Braga, Portugal, 2022.

OECD. OECD Digital Education Outlook 2026: Exploring Effective Uses of Generative AI in Education. [S. l.]: OECD Publishing (OECD Digital Education Outlook), 2026. Disponível em: https://www.oecd.org/en/publications/oecd-digital-education-outlook-2026_062a7394-en.html. Acesso em: 3 mar. 2026.

OECD. Recommendation of the Council on Artificial Intelligence. 2023. Disponível em: https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449. Acesso em: 15 de dezembro de 2024.

OLIVEIRA, Katyeudo Karlos de Sousa; MARCOLINO, Anderson da Silva; FALCÃO, Taciana Pontual; BARBOSA, Ellen Francine. Ensino e Aprendizagem de Programação na Educação 4.0: Um Mapeamento Sistemático da Literatura. In: Simpósio Brasileiro de Educação em Computação (EDUCOMP), 4. , 2024, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, p. 245-255. DOI: https://doi.org/10.5753/educomp.2024.237363, 2024.

OPENAI. Conheça o ChatGPT. [S. l.], 2022. Disponível em: https://openai.com/pt-BR/index/chatgpt/. Acesso em: 2 maio 2024.

OPENEDX. Open edX Platform: The Open-Source Platform for High-Impact Learning. [S. l.], 2024. Disponível em: https://openedx.org/platform/. Acesso em: 20 de maio de 2024.

ORYNBASSAROVA, Dilara; DE LASALA PORTA, Santiago. Implementing the Socratic Method with AI: Opportunities and Challenges of Integrating ChatGPT into Teaching Pedagogy. In: IMPLEMENTING THE SOCRATIC METHOD WITH AI, 2024, Stary Smokovec, Slovakia. 2024 International Conference on Emerging eLearning Technologies and Applications (ICETA). Stary Smokovec, Slovakia: IEEE, p. 526–532. DOI: 10.1109/ICETA63795.2024.10850795, 2024.

OTHMAN, Mahfudzah; ROSMANI, Arifah Fasha; MOHD FAUZI, Shukor Sanim; *et al*. The Impact of Pair Programming on Students Logical Thinking_A Case Study on Higher Academic Institution. Social and Management Research Journal, v. 16, n. 1, p. 85, DOI:10.24191/smrj.v16i1.6085, 2019.

PASSERINO, Liliana, SANTAROSA, L. M. C. Uma visão sócio-histórica da interação dentro de ambientes computacionais. In: : V Congresso Ibero-Americano de Informática na Educação - RIBIE 2000, 2000. Anais [...] Viña del Mar - Chile, [s.n.], 2000.

PAUL, Richard; ELDER, Linda. The Thinker’s Guide to The Art of Socratic Questioning. [S. l.]: The Foundation for Critical Thinking, 2016.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: PROCEEDINGS OF THE 12th INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING. Anais...: EASE’08, GBR: BCS Learning & Development Ltd., 2008, p. 68-77

PHUNG, Tung et al. Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation. In: Proceedings of the 14th Learning Analytics and Knowledge Conference. Kyoto Japan: ACM, 2024, p. 12–23. Disponível em: https://dl.acm.org/doi/10.1145/3636555.3636846.

PIAGET, J. Epistemologia Genética. Coleção Os Pensadores. Abril Cultural, 1970.

PIMENTEL, Johnny Félix Farfán et al. Digital Competence And Cooperative Learning In High School Students. Educational Administration: Theory and Practice, [s. l.], v. 30, n. 4, p. 5582–5592, 2024. Disponível em: https://kuey.net/index.php/kuey/article/view/2250. Acesso em: 23 set. 2024.

PRATHER, James *et al*. "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. ACM Transactions on Computer-Human Interaction, v. 31, n. 1, p. 1–31, DOI: https://dl.acm.org/doi/10.1145/3617367, 2024.

PREUSS, Evandro. NIDABA: Plataforma Digital para Produção de Recursos Educacionais Inclusivos Baseados em Mesa Tangível. 2021. Tese de Doutorado - Universidade Federal do Rio Grande do Sul - UFRGS, [s. l.], 2021.

PRINCE, Michael. Does Active Learning Work? A Review of the Research. Journal of Engineering Education, v. 93, n. 3, p. 223–231, DOI: https://doi.org/10.1002/j.2168-9830.2004.tb00809.x, 2004.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico. [S. l.]: Universidade Feevale, 2013.

QWEN *et al*. Qwen2.5 Technical Report. [S. l.]: arXiv, 2024. Disponível em: https://arxiv.org/abs/2412.15115. Acesso em: 2 maio 2025.

RAHMAN, Md. Mostafizer; WATANOBE, Yutaka. ChatGPT for Education and Research: Opportunities, Threats, and Strategies. Applied Sciences, v. 13, n. 9, p. 5783, 2023. Disponível em: https://www.mdpi.com/2076-3417/13/9/5783.

REEVES, Brent et al. Evaluating the Performance of Code Generation Models for Solving Parsons Problems With Small Prompt Variations. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. Turku Finland: ACM, 2023, p. 299–305. Disponível em: https://dl.acm.org/doi/10.1145/3587102.3588805.

ROZIÈRE, Baptiste et al. Code Llama: Open Foundation Models for Code. [S. l.]: arXiv, 2023. Disponível em: https://arxiv.org/abs/2308.12950. Acesso em: 2 maio 2025.

SÁNCHEZ-GORDÓN, Mary; TOVAR, Edmundo; COLOMO-PALACIOS, Ricardo; et al. Educating Augmented Programmers. Computer, v. 56, n. 12, p. 100–104, 2023. Disponível em: <https://ieeexplore.ieee.org/document/10319928/>.

SARKAR, A., GORDON, A. D., NEGREANU, C., POELITZ, C., RAGAVAN, S. S., ZORN, B. What is it like to program with artificial intelligence?, 2022. Disponível em: https://doi.org/10.48550/ARXIV.2208.06213

SARSA, Sami; DENNY, Paul; HELLAS, Arto; et al. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1. Lugano and Virtual Event Switzerland: ACM, 2022, p. 27–43. Disponível em: https://dl.acm.org/doi/10.1145/3501385.3543957.

SAVELKA, Jaromir et al. Large Language Models (GPT) Struggle to Answer Multiple-Choice Questions About Code: In: Proceedings of the 15th International Conference on Computer Supported Education. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2023a, p. 47–58. Disponível em:

https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0011996900003470.

SAVELKA, Jaromir et al. Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses? In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. Turku Finland: ACM, 2023b, p. 117–123. Disponível em: https://dl.acm.org/doi/10.1145/3587102.3588792.

SAVERY, J. R. Overview of Problem-based Learning: Definitions and Distinctions. Interdisciplinary Journal of Problem-Based Learning, vol 1(1). DOI: https://doi.org/10.7771/1541-5015.1002, 2006.

SCHULHOFF, Sander et al. The Prompt Report: A Systematic Survey of Prompting Techniques. [S. l.]: arXiv, 2024. Disponível em: https://arxiv.org/abs/2406.06608. Acesso em: 26 set. 2024.

SHAHADE, Aniket K; DESHMUKH, Priyanka V. Enhancing Natural Language Processing: A Comprehensive Review of Retrieval Augmented Generation. In: ENHANCING NATURAL LANGUAGE PROCESSING, 2024, Kaski, Nepal. 2024 4th International Conference on Sustainable Expert Systems (ICSES). Kaski, Nepal: IEEE, 2024. p. 609–611. Disponível em: https://ieeexplore.ieee.org/document/10763224/. Acesso em: 5 maio 2026.

SHEARD, Judy et al. Instructor Perceptions of AI Code Generation Tools - A Multi-Institutional Interview Study. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. Portland OR USA: ACM, 2024, p. 1223–1229. Disponível em: https://dl.acm.org/doi/10.1145/3626252.3630880.

SILVA JUNIOR, Silvino Marques Da; FREITAS, Robson Almeida Borges De; MORAIS, Márcio Aurélio Carvalho De; *et al.* ChatGPT no auxílio da aprendizagem de programação: Um estudo de caso. In: Anais do XXXIV Simpósio Brasileiro de Informática na Educação (SBIE 2023). Brasil: Sociedade Brasileira de Computação - SBC, p. 1375–1384, DOI: https://doi.org/10.5753/sbie.2023.234873, 2023.

SILVA, Carlos Alexandre Gouvea Da et al. ChatGPT: Challenges and Benefits in Software Programming for Higher Education. Sustainability, v. 16, n. 3, p. 1245, 2024. Disponível em: https://www.mdpi.com/2071-1050/16/3/1245.

SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor; TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da. Inteligência artificial generativa no ensino de programação: um mapeamento sistemático da literatura. RENOTE, Porto Alegre, v. 22, n. 1, p. 262–272, DOI: 10.22456/1679-1916.141553, 2024a.

SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor; TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da. Potencialidades do uso de Inteligência Artificial Generativa como apoio ao Ensino de Programação. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 35. , 2024, Rio de Janeiro/RJ. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, p. 1942-1956. DOI: https://doi.org/10.5753/sbie.2024.242711, 2024b.

SILVA, Teresinha Letícia da; TAROUCO, Liane Margarida Rockenbach. Development Of Computational Thinking: Goals And Strategies. In: 14th International Conference on Education and New Learning Technologies. Valência, Espanha: International Academy of Technology, Education and Development (IATED), p. 5171-5177, DOI: 10.21125/edulearn.2022.1355, 2022.

SIMON *et al*. Pass Rates in Introductory Programming and in other STEM Disciplines. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education. NY, USA: ACM, p. 53–71. DOI: https://doi.org/10.1145/3344429.3372502, 2019.

SOBRAL, S. 30 Years Of CS1: Programming Languages Evolution. 9197–9205. 2019. https://doi.org/10.21125/iceri.2019.2214

SOBRAL, Sónia Rolland. Strategies on Teaching Introducing to Programming in Higher Education. In: ROCHA, Álvaro et al. (org.). Trends and Applications in Information Systems and Technologies. Cham: Springer International Publishing, v. 1367, p. 133–150, DOI: 10.1007/978-3-030-72660-7_14, 2021a.

SOBRAL, Sónia Rolland. Project Based Learning with Peer Assessment in an Introductory Programming Course. International Journal of Information and Education Technology, [s. l.], v. 11, n. 7, p. 337–341, 2021b.

STEPHAN, João; OLIVEIRA, Alessandreia; RENHE, Marcelo Caniato. O Uso de Jogos para Apoiar o Ensino e Aprendizagem de Programação. Anais do XXXI Simpósio Brasileiro de Informática na Educação (SBIE 2020). Brasil: Sociedade Brasileira de Computação, p. 381 –390, DOI: https://doi.org/10.5753/cbie.sbie.2020.381, 2020.

STYVE, Arne; VIRKKI, Outi T.; NAEEM, Usman. Developing Critical Thinking Practices Interwoven with Generative AI Usage in an Introductory Programming Course. 2024, Kos Island, Greece. 2024 IEEE Global Engineering Education Conference (EDUCON). Kos Island, Greece: IEEE, 2024. p. 01–08. Disponível em: https://ieeexplore.ieee.org/document/10578746/. Acesso em: 4 maio 2025.

SUN, Dan *et al*. Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. International Journal of Educational Technology in Higher Education, v. 21, n. 1, p. 14, DOI: https://doi.org/10.1186/s41239-024-00446-5, 2024.

TA, Nguyen Binh Duong; NGUYEN, Hua Gia Phuc; GOTTIPATI, Swapna. ExGen: Ready-t ExGen: Ready-to-use ex o-use exercise gener cise generation in intr ation in introductory programming courses. In: Proceedings of the 31st International Conference on Computers in Education Conference. Japan, 2023. Disponível em: https://ink.library.smu.edu.sg/sis_research/8466/.

TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da; SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor. Digital competence formation of the citizen for the creation in the digital world using programming. Concilium, v. 23, n. 14, p. 443–470, https://doi.org/10.53660/CLM-1648-23J45, 2023. Disponível em: https://clium.org/index.php/edicoes/article/view/1648. Acesso em: 23 set. 2024.

TAVARES, Paula Correia. O impacto da animação e da avaliação automática na motivação para o ensino da programação. Tese de Doutorado. Escola de Engenharia. Universidade do Minho. Braga, Portugal, 2018.

THINAKARAN, Rajermani; ALI, Rosmah. Work in progress: An initial review in programming tutoring tools. In: 2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE). Zhuhai, China: IEEE, p. 1–4, DOI: 10.1109/TALE.2015.7386006, 2015.

THOMPSON, Errol, Whalley, Jacqueline, Lister, Raymond, and Simon, Beth. Code Classification as a Learning and Assessment Exercise for Novice Programmers. In 19th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2006). National Advisory Committee on Computing Qualifications, Wellington, New Zealand, 291–298, 2006.

TORRES, Nicolás. Do Robots Dream of Passing a Programming Course? In: 2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC). Concepcion, Chile: IEEE, 2023, p. 1–8. Disponível em: https://ieeexplore.ieee.org/document/10315744/.

TSAI, Chun-Yen. Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. Computers in Human Behavior, vol 95, p. 224-232, https://doi.org/10.1016/j.chb.2018.11.038, 2019.

UMAPATHY, Karthikeyan; RITZHAUPT, Albert D. A Meta-Analysis of Pair-Programming in Computer Programming Courses: Implications for Educational Practice. ACM Transactions on Computing Education, [s. l.], v. 17, n. 4, p. 1–13, DOI: https://dl.acm.org/doi/10.1145/2996201, 2017.

UNESCO. Guidance for generative AI in education and research. [s.l.]: UNESCO, DOI: https://doi.org/10.54675/EWZM9535, 2023.

VAITHILINGAM, P.; ZHANG, T.; GLASSMAN, E. L. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In: CHI '22: CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2022, New Orleans LA USA. CHI Conference on Human Factors in Computing Systems Extended Abstracts. New Orleans LA USA: ACM, 2022. p. 1–7. Disponível em: https://dl.acm.org/doi/10.1145/3491101.3519665.

VASWANI, Ashish *et al.* Attention Is All You Need. Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Red Hook, NY, USA: Curran Associates Inc., p. 6000–6010, 2017.

VICARI, R., BRACKMANN, C., MIZUSAKI, L., LOPES, D., BARONE, D., CASTRO, H. Referencial Curricular: Inteligência Artificial no Ensino Médio. ISBN 978-65-00-58427-1, DOI 10.13140/RG.2.2.23179.98089, http://inf.ufrgs.br/ciars. 2022.

VYGOTSKY, L. S. Mind in Society - The Development of Higher Psychological Processes. Cambridge MA: Harvard University Press, DOI: https://doi.org/10.2307/j.ctvjf9vz4, 1978.

VYGOTSKY, L. S. A formação social da mente: o desenvolvimento dos processos psicológicos superiores. São Paulo: Martins Fontes. 1989.

WANG, Feng Hsu. Efficient generation of text feedback in object-oriented programming education using cached performer revision. Machine Learning with Applications, v. 13, p. 100481, 2023. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S2666827023000348.

WANG, Tianjia et al. Exploring the Role of AI Assistants in Computer Science Education: Methods, Implications, and Instructor Perspectives. In: 2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). Washington, DC, USA: IEEE, 2023, p. 92–102. Disponível em: https://ieeexplore.ieee.org/document/10305701/.

WANG, Shan *et al*. Artificial intelligence in education: A systematic literature review. Expert Systems with Applications, [s. l.], v. 252, p. 124167, DOI: https://doi.org/10.1016/j.eswa.2024.124167, 2024.

WATSON, Christopher; LI, Frederick W.B. Failure rates in introductory programming revisited. In: THE 2014 CONFERENCE, 2014, Uppsala, Sweden. Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE'14. Uppsala, Sweden: ACM Press, p. 39–44, https://doi.org/10.1145/2591708.2591749, 2014.

WEIZENBAUM, Joseph. ELIZA - a computer program for the study of natural language communication between man and machine. Communications of the ACM, [s. l.], v. 9, n. 1, p. 36–45, DOI: https://dl.acm.org/doi/10.1145/365153.365168, 1966.

WERMELINGER, Michel. Using GitHub Copilot to Solve Simple Programming Problems. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. Toronto ON Canada: ACM, 2023, p. 172–178. Disponível em: https://dl.acm.org/doi/10.1145/3545945.3569830.

WILLIAM, Linda et al. Effectiveness of Supply Chain Games in Problem-Based Learning Environment. In: IFENTHALER, Dirk; KIM, Yoon Jeon (org.). Game-Based Assessment Revisited. Cham: Springer International Publishing, p. 257–280, DOI: 10.1007/978-3-030-15569-8_13, 2019.

WILLIAMS, Laurie et al. Strengthening the case for pair programming. IEEE Software, v. 17, n. 4, p. 19–25, DOI: https://doi.org/10.1109/52.854064, 2000.

WING, Jeannette M. Computational thinking. Communications of the ACM, [s. l.], v. 49, n. 3, p. 33–35, DOI://dl.acm.org/doi/10.1145/1118178.1118215, 2006.

XU, Fan; CORREIA, Ana-Paula. Adopting distributed pair programming as an effective team learning activity: a systematic review. Journal of Computing in Higher Education, [s. l.], v. 36, n. 2, p. 320–349, DOI: https://doi.org/10.1007/s12528-023-09356-3, 2024.

YAN, Wanxin; NAKAJIMA, Taira; SAWADA, Ryo. Benefits and Challenges of Collaboration between Students and Conversational Generative Artificial Intelligence in Programming Learning: An Empirical Case Study. Education Sciences, v. 14, n. 4, p. 433, 2024. Disponível em: <https://www.mdpi.com/2227-7102/14/4/433>. Acesso em: 20 set. 2024.

YANG, Xigui. A Historical Review of Collaborative Learning and Cooperative Learning. TechTrends, v. 67, n. 4, p. 718–728, 2023. Disponível em: <https://link.springer.com/10.1007/s11528-022-00823-9>. Acesso em: 20 set. 2024.

YILMAZ, Ramazan; KARAOGLAN YILMAZ, Fatma Gizem. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. Computers and Education: Artificial Intelligence, [s. l.], v. 4, p. 100147, DOI:10.1016/j.caeai.2023.100147, 2023.

YILMAZ, Ramazan; YILMAZ, Fatma Gizem Karaoglan. Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. Computers in Human Behavior: Artificial Humans, v. 1, n. 2, p. 100005, 2023. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S2949882123000051.

YUSOFF, Karimah Mohd; SAHARI, Noraidah; SITI, Tengku; *et al*. Analysis on the Requirements of Computational Thinking Skills to Overcome the Difficulties in Learning Programming. International Journal of Advanced Computer Science and Applications, v. 11, n. 3, DOI: 10.14569/IJACSA.2020.0110329, 2020.

ZHAO, W. X. *et al*. A Survey of Large Language Models. [s. l.], DOI: 10.48550/arXiv.2303.18223, 2023.

ZORZO, A. F.; NUNES, D.; MATOS, E.; STEINMACHER, I.; LEITE, J.; ARAUJO, R. M.; CORREIA, R.; MARTINS, S. Referenciais de Formação para os Cursos de Graduação em Computação. Sociedade Brasileira de Computação (SBC). 153p, DOI: https://doi.org/10.5753/sbc.ref.2017.134, 2017.

APÊNDICE I - TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Você está sendo convidado a participar da pesquisa intitulada Aprendizagem de Programação Apoiada por Inteligência Artificial Generativa: Um Estudo sobre Eficácia e Motivação com um Par de Programação Baseado em IA (AI Pair Programming). Esta pesquisa faz parte da tese de doutorado desenvolvida no programa de Pós-Graduação em Informática na Educação (PPGIE) – UFRGS, sob a orientação da Profª. Drª. Liane Margarida Rockenbach Tarouco e coorientação da Profª Drª. Patrícia Fernanda da Silva.

Deste modo, no caso de concordância em participar desta pesquisa, você deve assinar este termo para autorizar sua participação. Neste sentido, você estará ciente de que a partir da presente data:

- Os direitos da entrevista gravada ou respondidas (questionários) realizada pela pesquisadora, será utilizada integral ou parcialmente, sem restrições;

- estará assegurado o anonimato nos resultados dos dados obtidos, sendo que todos os registros ficarão de posse da pesquisadora por cinco anos e após esse período serão extintos.

Será garantido também:

- receber a resposta e/ou esclarecimento de qualquer pergunta e dúvida a respeito da pesquisa;

- poderá retirar seu consentimento a qualquer momento, deixando de participar do estudo, sem que isso traga qualquer tipo de prejuízo.

Assim, mediante termo de Consentimento Livre e Esclarecido, declaro que autorizo minha participação nesta pesquisa, por estar esclarecido e não me oferecer nenhum risco de qualquer natureza. Declaro, ainda, que as informações fornecidas nesta pesquisa podem ser usadas e divulgadas no programa de Pós-Graduação em Informática na Educação (PPGIE) – UFRGS, bem como nos meios científicos, publicações eletrônicas e apresentações profissionais.

_____

Participante da pesquisa (Nome completo e legível)

_____

Participante da pesquisa (Assinatura)

_____

Pesquisadora: Teresinha Letícia da Silva (leticiasilva.ufsm@gmail.com)

Frederico Westphalen, RS, _____ de _____.

APÊNDICE II - QUESTIONÁRIO DEMOGRÁFICO SOBRE O PERFIL DOS ESTUDANTES PARTICIPANTES DA ATIVIDADE

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO INTERDISCIPLINAR DE NOVAS TECNOLOGIAS NA EDUCAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA NA EDUCAÇÃO

Pesquisa: Aprendizagem de Programação Apoiada por Inteligência Artificial Generativa
Prezado(a) aluno(a),

Você está sendo convidado (a) a participar de uma pesquisa de doutorado sobre "Aprendizagem de Programação Apoiada por Inteligência Artificial Generativa". Esta pesquisa está sob a responsabilidade da aluna Teresinha Letícia da Silva, doutoranda do Programa de Pós-graduação em Informática na Educação da Universidade Federal do Rio Grande do Sul (PPGIE/UFRGS), orientada pela Profª. Drª. Liane Margarida Rockenbach Tarouco e coorientada: Profª Drª. Patrícia Fernanda da Silva.

Sua participação na pesquisa é livre, voluntária e suas respostas serão confidenciais, sendo utilizadas exclusivamente para fins de pesquisa acadêmica, segundo a Lei Geral de Proteção de Dados.

Seção 1: Informações Demográficas

1. Idade:
2. Gênero:
   - Feminino
   - Masculino
   - Prefiro não dizer
   - Outros…
3. Em qual semestre do curso você está?
   - 1º semestre
   - 2º semestre
   - 3º semestre
   - 4º semestre
   - 5º semestre
   - 6º semestre
   - 7º semestre
   - 8º semestre
   - 9º semestre

Seção 2: Conhecimentos em programação

4. Indique seu nível de conhecimento em programação.
   - Muito Suficiente
   - Suficiente
   - Nem Suficiente/Nem Insuficiente
   - Insuficiente
   - Muito Insuficiente
5. Em quais linguagens de programação você já programou?
   - Python
   - C++
   - Java
   - JavaScript
   - PHP
   - Outros…
6. Indique seu nível de conhecimento na linguagem de programação que será estudada nesta disciplina
   - Muito Suficiente
   - Suficiente
   - Nem Suficiente/Nem Insuficiente
   - Insuficiente
   - Muito Insuficiente

Seção 3: Uso de Ferramentas de IA Generativa

7. Você utiliza alguma ferramenta de IA generativa no seu aprendizado de programação?
   ( ) Sim
   ( ) Não

8. Se sim, quais ferramentas de IA generativa você usa? (Selecione todas que se aplicam)
   ( ) ChatGPT
   ( ) Gemini
   ( ) GitHub Copilot
   ( ) Repl.it
   ( ) OpenAI Codex
   ( ) Tabnine
   ( ) Code Llama
   ( ) DeepCode
   ( ) CodeWhisperer
   ( ) Outras (por favor, especifique)

9. Com que frequência você usa essas ferramentas?
   ( ) Sempre
   ( ) Frequentemente
   ( ) Ocasionalmente
   ( ) Raramente
   ( ) Nunca
10. Se não, por quais motivos você ainda não utilizou ferramentas de IA generativa? (Selecione todas que se aplicam)
   ( ) Falta de tempo
   ( ) Falta de interesse
   ( ) Falta de conhecimento sobre as ferramentas
   ( ) Dificuldade de acesso à tecnologia
   ( ) Preocupação com questões éticas e de privacidade
   ( ) Preocupação com plágio e falta de originalidade
   ( ) Outras (por favor, especifique)
11. Como você avalia a eficácia das ferramentas de IA generativa no auxílio ao aprendizado de programação?
   ( ) Muito eficaz
   ( ) Eficaz
   ( ) Moderadamente eficaz
   ( ) Pouco eficaz
   ( ) Ineficaz
12. Em que aspectos você acredita que as ferramentas de IA generativa mais ajudam no seu aprendizado de programação? (Selecione todas que se aplicam)
   ( ) Aceleração do processo de escrita de código
   ( ) Correção automática de erros
   ( ) Fornecimento de exemplos e exercícios personalizados
   ( ) Feedback em tempo real
   ( ) Pesquisa de informações
   ( ) Resolução de problemas complexos
   ( ) Estudo de conteúdos específicos
   ( ) Tirar dúvidas pontuais
   ( ) Outros (por favor, especifique)
13. Quais são as principais limitações ou desafios que você enfrenta ao usar ferramentas de IA generativa no aprendizado? (Selecione todas que se aplicam)
   ( ) Custo das ferramentas
   ( ) Dificuldade de aprendizado para usar as ferramentas
   ( ) Dependência excessiva da tecnologia
   ( ) Qualidade do conteúdo gerado pela IA
   ( ) Informações incorretas ou incompletas
   ( ) Falta de desenvolvimento do pensamento crítico
   ( ) Possível plágio
   ( ) Dificuldade em entender as respostas geradas pela ferramenta
   ( ) Outros (por favor, especifique)
14. Você percebe uma mudança na sua motivação para aprender programação ao usar ferramentas de IA generativa?
   ( ) Aumento significativo
   ( ) Aumento moderado
   ( ) Nenhuma mudança
   ( ) Redução moderada
   ( ) Redução significativa
15. Você observa uma melhoria no seu desempenho ao usar ferramentas de IA generativa?
   ( ) Sim, significativa
   ( ) Sim, moderada
   ( ) Não
   ( ) Redução no desempenho
16. Você acha que o uso dessas ferramentas facilita a resolução de problemas de programação?
   ( ) Sim, muito
   ( ) Sim, um pouco
   ( ) Não
   ( ) Dificulta a resolução de problemas
17. Você sente que depende excessivamente dessas ferramentas para completar suas tarefas de programação?
   ( ) Sim, frequentemente
   ( ) Sim, ocasionalmente
   ( ) Não

18. Quais melhorias você gostaria de ver nas ferramentas de IA generativa para uso educacional? (Resposta aberta)

APÊNDICE III - QUESTIONÁRIO DE MOTIVAÇÃO BASEADO NO IMI

1. Achei esta atividade muito interessante.
2. Esta atividade foi divertida para mim.
3. Achei esta atividade bastante envolvente.
4. Gostei muito de realizar esta atividade.
5. A atividade manteve minha atenção.
6. Eu faria essa atividade novamente por conta própria.
7. Me esforcei bastante para realizar esta atividade.
8. Achei importante fazer bem essa atividade.
9. Dediquei tempo e atenção para concluir a atividade.
10. Dei o meu melhor nesta atividade.
11. Fiquei motivado(a) a terminar toda a tarefa.
12. Acredito que esta atividade foi útil para meu aprendizado.
13. A atividade tem valor educacional para mim.
14. Vejo utilidade prática no que aprendi com esta atividade.
15. Fazer essa atividade contribuiu para meu desenvolvimento.
16. A atividade foi relevante para minha formação acadêmica.
17. Me senti competente durante a atividade.
18. Achei que fui bem na resolução da atividade.
19. Fiquei confiante com meu desempenho.
20. Me senti capaz de concluir as tarefas propostas.
21. Consegui resolver os desafios da atividade sem muita dificuldade.
22. Me senti pressionado(a) durante a atividade.
23. A atividade me deixou tenso(a).
24. Senti que precisava correr ou fazer rápido demais.
25. Fiquei estressado(a) ao realizar a atividade.

APÊNDICE IV - QUESTIONÁRIOS DE AVALIAÇÃO DE USABILIDADE

Questionário SUS - System Usability Scale:
Utilize a escala de 1 a 5, onde:
1 = Discordo totalmente
2 = Discordo
3 = Neutro
4 = Concordo
5 = Concordo totalmente

1. Acho que gostaria de usar o assistente de IA com frequência
2. Eu achei o assistente de IA muito complexo
3. Eu acho que o assistente de IA tem sido fácil de usar
4. Eu acho que precisaria de suporte técnico para usar o assistente de IA
5. Eu acho que as várias funções do sistema estão muito bem integradas
6. Eu acho que há muita inconsistência no assistente de IA
7. Eu acho que a maioria das pessoas aprenderá a usar o assistente de IA rapidamente
8. Eu acho estranho a forma como usar o assistente de IA
9. Eu me senti confiante ao usar o assistente de IA
10. Eu precisei aprender várias coisas novas antes de conseguir usar o assistente de IA

APÊNDICE V - QUESTIONÁRIO DE PERCEPÇÃO E MOTIVAÇÃO COM O PAR DE PROGRAMAÇÃO (IA OU HUMANO) - ESTUDO PRELIMINAR

Instruções: Responda às perguntas abaixo com base em sua experiência na atividade como seu par de programação (IA ou Colega). Utilize a escala de 1 a 5, onde:
1 = Discordo totalmente
2 = Discordo
3 = Neutro
4 = Concordo
5 = Concordo totalmente

Dimensão: Interesse e Prazer (Baseado no IMI)

1. Eu gostei de usar a IA como par de programação durante a resolução de exercícios. (1-5)
2. A experiência de usar a IA como assistente de programação foi interessante para mim. (1-5)
3. Eu me senti envolvido enquanto usava a IA para resolver problemas de programação. (1-5)
4. Trabalhar com a IA como par de programação me deu uma sensação de satisfação. (1-5)

Dimensão: Percepção de Competência (Baseado no IMI)

5. Usar a IA me ajudou a entender melhor os conceitos de programação. (1-5)
6. Eu sinto que minha habilidade de resolver problemas de programação melhorou ao usar a IA. (1-5)
7. A IA me ajudou a sentir mais confiança na minha capacidade de programar. (1-5)
8. Eu me senti mais competente ao resolver exercícios com a ajuda da IA. (1-5)

Dimensão: Percepção de Utilidade (Baseado no AMS)

9. A IA foi útil para meu aprendizado de programação. (1-5)
10. Eu acho que o uso da IA vai me ajudar a ter um melhor desempenho em programação no futuro. (1-5)
11. Eu acredito que a IA como par de programação é uma ferramenta importante para meus estudos. (1-5)

Dimensão: Autonomia (Baseado no IMI e AMS)

12. Eu me senti no controle do processo de aprendizagem enquanto usava a IA. (1-5)
13. O uso da IA me permitiu tomar decisões sobre como resolver os problemas de programação. (1-5)
14. Eu tive liberdade para usar a IA da maneira que achei mais adequada ao meu aprendizado. (1-5)

Dimensão: Engajamento e Motivação (Baseado no IMI)

15. Eu me senti motivado a continuar aprendendo programação com a IA. (1-5)
16. A IA tornou o processo de aprendizado mais interessante e me incentivou a continuar estudando. (1-5)
17. Eu gostaria de continuar usando a IA como parte do meu aprendizado de programação. (1-5)
18. O uso da IA aumentou meu interesse em programar fora das aulas. (1-5)

Dimensão: Relacionamento com o Assistente de IA (Adaptação própria)

19. Eu senti que a IA atuou como um bom parceiro de programação. (1-5)
20. Eu pude confiar nas orientações fornecidas pela IA durante o processo de programação. (1-5)

Questão aberta:

21. Comentários que achar pertinente sobre a atividade:

APÊNDICE VI - TABELA DE NOTAS DOS ALUNOS DO ESTUDO FINAL 1

A Tabela abaixo apresenta as notas de cada aluno no pré-teste, pós-teste e o ganho de aprendizagem para os grupos controle e experimental no Estudo Final 1.

id_aluno	grupo	pre_teste	pos_teste	ganho
AlunoControle_01	Controle	5	7	2
AlunoControle_02	Controle	0	1	1
AlunoControle_03	Controle	7,2	7,8	0,6
AlunoControle_04	Controle	3	8,2	5,2
AlunoControle_05	Controle	2,7	9,6	6,9
AlunoControle_06	Controle	4,5	7,4	2,9
AlunoControle_07	Controle	0	2,8	2,8
AlunoControle_08	Controle	6,6	5,2	-1,4
AlunoControle_09	Controle	7,5	9,8	2,3
AlunoControle_10	Controle	3,3	7,4	4,1
AlunoControle_11	Controle	6	4,7	-1,3
AlunoControle_12	Controle	1,2	2,1	0,9
AlunoControle_13	Controle	5	2,6	-2,4
AlunoExperimental_01	Experimental	2,4	6,6	4,2
AlunoExperimental_02	Experimental	6	9,7	3,7
AlunoExperimental_03	Experimental	3	10	7
AlunoExperimental_04	Experimental	2,4	5	2,6
AlunoExperimental_05	Experimental	2,4	7	4,6
AlunoExperimental_06	Experimental	3,9	7,5	3,6
AlunoExperimental_07	Experimental	1,5	5,7	4,2
AlunoExperimental_08	Experimental	2	3,3	1,3
AlunoExperimental_09	Experimental	5	9,4	4,4
AlunoExperimental_10	Experimental	3	4,2	1,2
AlunoExperimental_11	Experimental	3	4,5	1,5
AlunoExperimental_12	Experimental	3,3	4,9	1,6
AlunoExperimental_13	Experimental	1,8	7,6	5,8
AlunoExperimental_14	Experimental	1	8,9	7,9

APÊNDICE VII - TABELA DE NOTAS DOS ALUNOS DO ESTUDO FINAL 2

A Tabela abaixo apresenta as notas de cada aluno no pré-teste, pós-teste e o respectivo ganho de aprendizagem para os grupos controle e experimental do Estudo final 2.

id_aluno	grupo	pre_teste	pos_teste	ganho
AlunoControle_01	Controle	5	7	2
AlunoControle_02	Controle	2,8	9,2	6,4
AlunoControle_03	Controle	1	2,5	1,5
AlunoControle_04	Controle	1,3	5,1	3,8
AlunoControle_05	Controle	1	3,8	2,8
AlunoControle_06	Controle	2	6,7	4,7
AlunoControle_07	Controle	3,3	9,5	6,2
AlunoControle_08	Controle	2	9,3	7,3
AlunoControle_09	Controle	2,3	5,3	3
AlunoControle_10	Controle	1	5,8	4,8
AlunoControle_11	Controle	0	2,7	2,7
AlunoControle_12	Controle	1	2,8	1,8
AlunoControle_13	Controle	3,5	3	-0,5
AlunoControle_14	Controle	1	6,7	5,7
AlunoControle_15	Controle	1,8	7	5,2
AlunoExperimental_01	Experimental	3,5	7,4	3,9
AlunoExperimental_02	Experimental	0,3	1,5	1,2
AlunoExperimental_03	Experimental	3,5	9,6	6,1
AlunoExperimental_04	Experimental	2	8,8	6,8
AlunoExperimental_05	Experimental	2,8	8,8	6
AlunoExperimental_06	Experimental	1,5	6,4	4,9
AlunoExperimental_07	Experimental	1	7,9	6,9
AlunoExperimental_08	Experimental	1	8,4	7,4
AlunoExperimental_09	Experimental	0	6,4	6,4
AlunoExperimental_10	Experimental	2	5,4	3,4
AlunoExperimental_11	Experimental	2,5	7,4	4,9
AlunoExperimental_12	Experimental	0	8,4	8,4
AlunoExperimental_13	Experimental	1	7,3	6,3
AlunoExperimental_14	Experimental	2,5	7,8	5,3
AlunoExperimental_15	Experimental	6	8,7	2,7

APÊNDICE VIII - LISTA DE PUBLICAÇÕES REALIZADAS DURANTE O DESENVOLVIMENTO DA TESE

Artigos publicados em periódicos

SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor; TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da. **AI Pair Programming: Estudo sobre o uso do ChatGPT como par de programação**. RENOTE, 2025. DOI: https://doi.org/10.22456/1679-1916.153547. Disponível em: https://seer.ufrgs.br/renote/article/view/153547 (Qualis A4)

VIDOTTO, K. N. S., SILVA, T. L. da, & TAROUCO, L. M. R. **Programming virtual experiments as a teaching and learning strategy for engineering**. Cadernos De Educação Tecnologia E Sociedade, 18(3), 786-801. 2025. DOI: https://doi.org/10.14571/brajets.v18.n3.786-801. Disponível em: https://brajets.com/brajets/article/view/1410 (Qualis A4)

SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor; TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da. **Inteligência artificial generativa no ensino de programação: um mapeamento sistemático da literatura**. RENOTE, Porto Alegre, v. 22, n. 1, p. 262–272, 2024. DOI: 10.22456/1679-1916.141553. Disponível em: https://seer.ufrgs.br/index.php/renote/article/view/141553 (Qualis A4)

TAROUCO, Liane Margarida Rockenbach ; SILVA, Patrícia Fernanda Da; SILVA, Teresinha Letícia Da ; VIDOTTO, Kajiana Nuernberg Sartor. **Digital competence formation of the citizen for the creation in the digital world using programming**. CONCILIUM (ENGLISH LANGUAGE EDITION), v. 23, p. 443-470, 2023. DOI: 10.53660/CLM-1648-23J45 Disponível em: https://clium.org.uk/index.php/editions/article/view/1648 (Qualis A2)

ROCKENBACH TAROUCO, Liane Margarida; LICKS MISSEL MACHADO, Luís Antônio; SILVA, Teresinha Letícia da; TIMÓTEO, Dúlcio Joaquim Antonio. **Possibilidades Do Metaverso Como Recurso Educacional**. Revista da FUNDARTE, [S. l.], v. 56, n. 56, p. 1–22, 2023. DOI: 10.19179/rdf.v56i56.1268. Disponível em: https://seer.fundarte.rs.gov.br/index.php/RevistadaFundarte/article/view/1268 (Qualis A2)

MACHADO, L. A. L. M. ; SILVA, Teresinha Letícia Da ; TIMOTEO, D. J. A. ; TAROUCO, L. M. R. . **Recursos multimídia na educação sob o enfoque da teoria cognitiva de aprendizagem de Richard Mayer**. Redin - Revista Educacional Interdisciplinar, v. 12, p. 121-140, 2023. Disponível em: https://seer.faccat.br/index.php/redin/article/view/2855 (Qualis A4)

Trabalhos completos publicados em anais de congressos

SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor; TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da. **Análise do Uso de Ferramentas**

de IA Generativa no Ensino de Programação: Perspectivas de Estudantes do Curso de Sistemas de Informação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 33. , 2025, Maceió/AL. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2025. p. 1277-1288. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2025.9236. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/36259 (Qualis A4)

SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor; TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da. **Potencialidades do uso de Inteligência Artificial Generativa como apoio ao Ensino de Programação**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 35. , 2024, Rio de Janeiro/RJ. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 1942-1956. DOI: https://doi.org/10.5753/sbie.2024.242711. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/31369 (Qualis A3)

SILVA, Teresinha Letícia da; VIDOTTO, Kajiana Nuernberg Sartor; TAROUCO, Liane Margarida Rockenbach. **Investigando as percepções de estudantes e professores do ensino médio e técnico sobre o uso do ChatGPT em suas atividades escolares**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 35. , 2024, Rio de Janeiro/RJ. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 1851-1864. DOI: https://doi.org/10.5753/sbie.2024.242584. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/31362 (Qualis A3)

VIDOTTO, Kajiana Nuernberg Sartor; SILVA, Teresinha Letícia da; TAROUCO, Liane Margarida Rockenbach; SILVA, Patrícia Fernanda da. **Construção de um Ambiente em Realidade Virtual para o Estudo de Energia Elétrica Renovável**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 35. , 2024, Rio de Janeiro/RJ. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 2816-2825. DOI: https://doi.org/10.5753/sbie.2024.244905. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/31443 (Qualis A3)

ROCKENBACH TAROUCO, Liane Margarida ; SILVA, Teresinha Letícia da ; SILVA, Patrícia Fernanda da. **Assessment Using Active Student Participation**. In: 16th International Conference on Education and New Learning Technologies, 2024, Palma. org.crossref.xschema._1.Title@60bad562, 2024. p. 1048. DOI: https://doi.org/10.21125/edulearn.2024.0365. Disponível em: https://library.iated.org/view/ROCKENBACHTAROUCO2024ASS

SILVA, Patricia Fernanda ; SILVA, Teresinha Letícia ; TAROUCO, L. M. R. . **Mapeamento da Competência Digital de docentes do Ensino Superior**. In: 29º CIAED - Congresso Internacional ABED de Educação a Distância, 2024, Brasília, DF. Anais do 29º CIAED - Congresso Internacional ABED de Educação a Distância, 2024. Disponível em: https://proceedings.science/ciaed-2024/trabalhos/mapeamento-da-competencia-digital-de-docentes-do-ensino-superior-2?lang=pt-br

SILVA, Teresinha Letícia; VIDOTTO, Kajiana Nuernberg Sartor ; TAROUCO, L. M. R. ; SILVA, P. F. ; MACHADO, L. A. L. M. ; SCHMITT, M. A. R. . **Taxonomia para Apoiar a Promoção da Competência Digital de Educadores.** In: LACLO 2024: XIX Conferencia Latinoamericana de Tecnologías de Aprendizaje. Montevideo, Uruguai:

Montevideo: Universidad Nacional de La Plata (UNLP), 2024. p. 25-37. Disponível em: https://sedici.unlp.edu.ar/bitstream/handle/10915/179170/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

MACHADO, Luís Antônio Licks Missel; SILVA, Teresinha Letícia da; TAROUCO, Liane Margarida Rockenbach; HERPICH, Fabrício. **Metaverso para educação em desenvolvimento sustentável**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 34. , 2023, Passo Fundo/RS. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023. p. 536-547. DOI: https://doi.org/10.5753/sbie.2023.233539. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/26691 (Qualis A3)

SILVA, Teresinha Letícia da; DOS SANTOS, Gilson Saturnino . **Development Of One Game In Scratch To Assist In Teaching Computational Thinking**. In: 14th International Conference on Education and New Learning Technologies, EDULEARN22 Proceedings, 2022, Palma, 2022. p. 7707-7710. DOI: https://doi.org/10.21125/edulearn.2022.1793. Disponível em: https://library.iated.org/view/DASILVA2022DEV2

SILVA, Teresinha Letícia da; ROCKENBACH TAROUCO, Liane Margarida. **Development Of Computational Thinking: Goals And Strategies**. In: 14th International Conference on Education and New Learning Technologies, EDULEARN22 Proceedings, 2022, Palma, 2022. p. 5771-5777. DOI: 10.21125/edulearn.2022.1355. Disponível em: https://library.iated.org/view/DASILVA2022DEV

LIMA, Rafaela de Araujo Sampaio; SILVA, Francisco Xavier da; SILVA, Teresinha Letícia da; MENEZES, Crediné Silva de. **Uma arquitetura pedagógica para aprendizagem de paisagismo baseada em jogos digitais e pensamento computacional.** In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 33. , 2022, Manaus. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2022. p. 196-207. DOI: https://doi.org/10.5753/sbie.2022.225105. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/22408 (Qualis A3)

PEREIRA, Andréa; LIMA, Rafaela de Araujo Sampaio; SILVA, Teresinha Letícia da; MENEZES, Crediné Silva de. **Passeio no parque: uma arquitetura pedagógica para promover o desenvolvimento da estrutura de seriação**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 33. , 2022, Manaus. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2022. p. 220-231. DOI: https://doi.org/10.5753/sbie.2022.225118. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/22410 (Qualis A3)

Capítulo de livro

Liane Margarida Rockenbach Tarouco, Patrícia Fernanda da Silva e / and Teresinha Letícia da Silva. Competência digital no Brasil, In: Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros: TIC Domicílios 2023 [livro eletrônico] = Survey on the use of information and communication technologies in Brazilian households:

ICT Households 2023 / [editor] Núcleo de Informação e Coordenação do Ponto BR. -- 1. ed. -- São Paulo : Comitê Gestor da Internet do Brasil, 2024. ISBN 978-65-85417-31-0. Disponível em: https://cetic.br/media/docs/publicacoes/2/20241104102822/tic_domicilios_2023_livro_eletronico.pdf

Premiações

- O trabalho intitulado “Análise do Uso de Ferramentas de IA Generativa no Ensino de Programação: Perspectivas de Estudantes do Curso de Sistemas de Informação”, recebeu a premiação de Menção Honrosa como um dos 3 melhores artigos do Workshop sobre Educação em Computação (WEI), realizado durante o 45° Congresso da Sociedade Brasileira de Computação (CSBC 2025), no período de 20 a 24 de julho de 2025, em Maceió/AL.
- O trabalho intitulado “Mapeamento da Competência Digital de docentes do Ensino Superior” recebeu Menção Honrosa no 29o CIAED – Congresso Internacional ABED de Educação a Distância, realizado pela Associação Brasileira de Educação a Distância – ABED no período de 15 a 18 de setembro de 2024, em Brasília - DF.

NOTAS

Em conformidade com as diretrizes de integridade na atividade científica do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), declara-se que, durante a elaboração desta tese, foi utilizada uma ferramenta de Inteligência Artificial Generativa como apoio ao processo de escrita acadêmica.

A ferramenta utilizada foi o ChatGPT, da OpenAI, empregado exclusivamente como apoio à revisão textual e gramatical, à tradução de trechos, à organização de ideias e à reformulação de trechos redacionais. O uso da ferramenta teve caráter auxiliar e não substituiu a autoria, a análise crítica, as decisões metodológicas, a interpretação dos resultados ou a responsabilidade acadêmica da autora.

Todas as informações, argumentos, análises, resultados e conclusões apresentados nesta tese foram revisados, selecionados e validados pela autora, permanecendo sob sua inteira responsabilidade.