FERRAMENTAS DE ASSISTÊNCIA IA: ANÁLISE DOS IMPACTOS NA APRENDIZAGEM EM PROGRAMAÇÃO

IA ASSISTANCE TOOLS: ANALYSIS OF THE IMPACT ON PROGRAMMING LEARNING.

Lucas de Souza Moraes[1]
Pedro Henrique Pereira Rodrigues[2]
Mariângela Ferreira Fuentes Molina[3]

RESUMO: Este artigo apresenta um estudo sobre o impacto de ferramentas de assistência, baseadas em inteligência artificial (IA), como o CodeWhisperer, no aprendizado e no desenvolvimento de habilidades de programação de alunos do curso de Análise e Desenvolvimento de Sistemas. A metodologia de pesquisa utilizada tem um caráter quantitativo de natureza exploratório e foi dividido em duas fases, uma pesquisa bibliográfica e um experimento. Ao final deste experimento, os resultados indicaram que, embora o CodeWhisperer tenha facilitado a codificação e proporcionado uma experiência mais rápida, não contribuiu, significativamente, para a compreensão profunda de conceitos como expressões lambda, sugerindo uma possível dependência que pode prejudicar a formação de uma base sólida de conhecimento. O estudo sugere que, no ensino de programação, o uso de tais ferramentas devem ser equilibrado com práticas pedagógicas que incentivem o pensamento crítico e a autonomia dos alunos.
Palavras-chave: Assistência IA; CodeWhisperer; Desenvolvimento de *Software*; Ensino de Programação; Ferramentas de Codificação; Inteligência Artificial.

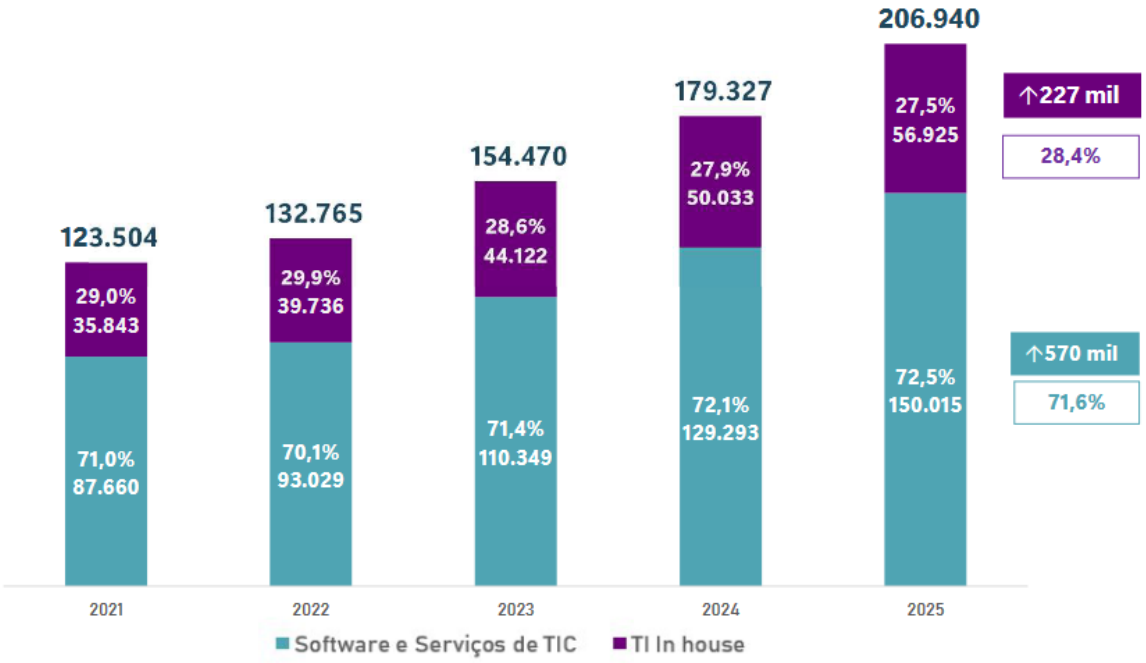
ABSTRACT: This article presents the impact of AI-based assistance tools, such as CodeWhisperer, on the learning and development of programming skills among students in the Systems Analysis and Development course. The research methodology used is quantitative and exploratory in nature and was divided into two phases: a bibliographical survey and an experiment. At the end of the experiment, the results indicated that although CodeWhisperer facilitated coding and provided a faster experience, it did not significantly contribute to a deep understanding of concepts like lambda expressions, suggesting a potential dependency that could hinder the formation of a solid knowledge base. The study suggests that, in programming education, the use of such tools should be balanced with pedagogical practices that encourage critical thinking and student autonomy.
Keywords: AI Assistance; CodeWhisperer; Software Development; Programming Education; Coding Tools; Artificial Intelligence.

Graduando, Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia de Mogi das Cruzes FATEC - MC. E-mail: lucas.moraes35@fatec.sp.gov.br[1]
Graduando, Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia de Mogi das Cruzes FATEC - MC. E-mail: pedro.rodrigues18@fatec.sp.gov.br[2]
Docente Curso Superior de Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Mogi das Cruzes FATEC – MC. Email: mariangela.molina@fatec.sp.gov.br[3]

1 INTRODUÇÃO

A área de desenvolvimento de sistemas tem passado por um constante crescimento nos últimos anos, aumentando o mercado de trabalho para esse setor, oferecendo oportunidades, ocasionando um crescente interesse dos profissionais em adentrar nessa área. Segundo os dados da Brasscom (2021), Associação das Empresas de Tecnologia da Informação e Comunicação, o número de vagas no mercado de tecnologia da informação é grande em comparação com a quantidade de mão de obra especializada disponível no mercado, ocasionando em um *déficit* de profissionais neste setor. O gráfico a seguir demonstra a situação:

Gráfico 1: Demanda de Talentos Em TIC E Estratégia.



Fonte: Brasscom(,2021)

Por consequência, muitos entusiastas buscam maneiras de aprender conceitos e fundamentos essenciais à área, de maneira mais rápida e eficiente, para que se consiga a vaga esperada dentro do mercado de TI. Nesse contexto, surgem diversos modos de apresentar e desenvolver conhecimento da programação, seja através de guias, cursos ou ferramentas. Por exemplo, ao utilizar editores de código para desenvolver programas de computador, uma série de ferramentas de auxílio estão disponíveis. E diante do exposto, tendo em vista a utilização dessas ferramentas, surgem diversas opções, que inicialmente buscavam auxiliar o desenvolvedor a digitar mais rápido, consultar de maneira ágil a sintaxe de uma linguagem e até realizar

referências ao seu código de maneira mais eficaz. Contudo, com o passar do tempo, esses recursos se tornaram mais avançados, podendo sugerir lógicas e fundamentos de realização de um código com pequenos passos.

Uma dessas soluções mais avançadas está sendo desenvolvida pelo GitHub e se chama GitHub Copilot, a qual utiliza inteligência artificial para sugerir trechos de código e funções em tempo real, possibilitando um desenvolvimento mais eficiente e acessível às pessoas. Além do GitHub Copilot, atualmente, diversas outras ferramentas estão sendo apresentadas como TabNine e o CodeWhisperer, sendo esta a ferramenta utilizada nesta pesquisa.

Com essas possibilidades e os recursos que oferecem, surge a dúvida, esse processo torna o aprendizado consistente, ou apenas torna o desenvolvimento algo mais automático evitando a utilização do pensamento lógico necessário na área?

Neste contexto, este artigo tem como objetivo analisar os impactos do uso da ferramenta de assistência por inteligência artificial CodeWhisperer no processo de aprendizagem de programação, por meio da comparação entre grupos de estudantes submetidos a diferentes condições de uso da ferramenta, com base em análise quantitativa dos resultados de desempenho e percepção dos alunos.

A importância desta pesquisa está presente no fato de que o desenvolvimento e a utilização deste tipo de recurso é algo ainda novo, mas já é possível apontar os prováveis impactos que essas ferramentas podem ocasionar, sobretudo no processo de aprendizagem. Logo realizar um estudo pragmático sobre a utilização desses meios e como eles funcionam, possibilita introduzir aos próprios futuros desenvolvedores os riscos ou garantias que a utilização de ferramentas como CodeWhisperer podem trazer.

2 METODOLOGIA

No presente trabalho foi realizado uma pesquisa quantitativa com caráter exploratório. A intenção é analisar essa recente tecnologia e sua constante evolução. Para alcançar os resultados e possíveis respostas sobre essa questão, foram utilizados dois procedimentos principais: um levantamento bibliográfico e estudo de caso elaborados através de exercícios práticos e questionários.

O levantamento bibliográfico foi composto por uma seleção de artigos científicos e matérias especializadas, em plataformas como IEEE e Google

Acadêmico, que forneceram exemplos, orientações e perspectivas futuras sobre as ferramentas de assistência IA e o processo de aprendizagem na programação. Essa revisão bibliográfica tem como objetivo criar um embasamento teórico para a realização da pesquisa prática e a apresentação das conclusões.

3 REFERENCIAL TEÓRICO

Nessa seção são apresentados os estudos sobre as ferramentas de assistência em programação, como elas funcionam e um estudo sobre o processo de aprendizagem de programação. Esse estudo foi necessário para dar o embasamento teórico essencial para a conclusão da pesquisa proposta.

3.1 FERRAMENTAS DE ASSISTÊNCIA EM PROGRAMAÇÃO E USO DE INTELIGÊNCIA ARTIFICIAL NA CRIAÇÃO DE CÓDIGOS.

As ferramentas de assistência em programação são todas de *software,* atualmente, utilizadas para auxiliar no processo da codificação de programas de computador. Elas são conhecidas como assistentes de codificação, visando trazer benefícios em produtividade e qualidade de código gerado pelo programador.

Inicialmente, essas ferramentas eram consideradas algo complementar. As chamadas IDEs, um acrônimo para a sigla em inglês *Integrated Development Environment* (Ambiente de desenvolvimento integrado) fornecia recursos como o preenchimento automático do código, consulta rápida sobre sintaxe ou preparação de estruturas mais comuns como condicionais ou loopings.

Nos dias de hoje, essas ferramentas têm se tornado cada vez mais importantes, devido a melhorias nelas implementadas, como a utilização de inteligência artificial, para dar sugestões de melhorias no código ou agir, diretamente, criando trechos de código a partir de alguns comandos (Pearce,2021).

Esses aperfeiçoamentos estão sendo possíveis pela capacidade de analisar e compreender padrões de código que uma inteligência artificial pode possuir. Seja utilizando algoritmos de aprendizado de máquina e técnicas de processamento da linguagem natural, processando grandes quantidades de dados, reconhecendo seus padrões e resultando em sugestões e geração de trechos de código.

Estes assistentes são treinados com grandes repositórios de código, como por exemplo, trechos de código aberto vindos de repositórios públicos do GitHub. Podem também utilizar códigos específicos, preparados pela própria instituição criadora da ferramenta. Por conta deste tipo de treinamento, em contato direto com códigos já existentes, é possível que essas ferramentas tragam em sua lógica, o reconhecimento de muitos padrões comuns no desenvolvimento e realizam recomendações que podem corresponder às melhores práticas e técnicas de codificação comumente usadas (Chen,2021).

Em uma visão geral das ferramentas de assistência que utilizam de recursos IA, há como principais o TabNine, GitHub Copilot, CodeWhisperer. Além disso, há algumas ferramentas com capacidade de geração de código, mas que não possuem como principal objetivo o desenvolvimento de *software,* como ChatGPT, Microsoft Copilot, Gemini.

Apesar de muitos pontos positivos serem apresentados por essas ferramentas, ainda possuem alguns pontos de alerta, como o fato, desses recursos serem treinados com uma base de dados diversa, proveniente, de ambientes abertos como o próprio GitHub, e como resultado, também podem ser obtidos padrões de codificação não seguros, bugs e códigos não eficientes (Pearce,2021).

É recomendado que o usuário possua, ao menos, um conhecimento básico em programação, para utilizar a ferramenta de maneira plena, consciente e responsável. Avaliando a relação de benefícios versus riscos que a utilização desses recursos oferece, e caso se opte por utilizar, é importante realizar validações para garantir a segurança e qualidade do código.

3.2 ASPECTOS FUNDAMENTAIS DAS FERRAMENTAS DE ASSISTÊNCIA IA

Os recursos avançados destas ferramentas de assistência de codificação, são baseadas em modelos de inteligência artificial, em específico sistemas de processamento de linguagem natural, capazes de analisar entradas de texto e gerar respostas com base no que for escrito, sendo assim capaz de compreender e manipular a linguagem (Brown,2020).

Entretanto, para ocorrer o funcionamento dessas ferramentas, se torna necessário a criação de modelos capazes de compreender, além da linguagem natural, também compreender aspectos mais técnicos voltados ao ambiente da

programação. Para atingir esta compreensão, estes modelos especializados são preparados com base em repositórios de código aberto, como o GitHub, ou respostas selecionadas do Stack Overflow.

Um exemplo de modelo especializado, seria o Codex, que é uma criação baseada no sistema de processamento de linguagem natural “Generative Pre-Training Transformer 3 (GPT-3)”, que já era capaz de gerar códigos de programação desde sua criação, entretanto com a especialização do Codex se tornou possível respostas melhores e apropriadas para uso de ferramentas como o GitHub Copilot (Chen,2021).

Porém, tanto o Codex quanto outros modelos, estão suscetíveis a certas limitações que são notadas desde o processo de treinamento. No caso do Codex, foi observado que, inicialmente, ocorria uma baixa eficiência em relação à quantidade de amostras necessárias para treiná-lo, sendo necessário muitas amostras para um resultado significativo, e a sua capacidade reduzida em produzir códigos a partir de requisitos complexos e abstratos, resultando em recomendações inadequadas, códigos indefinidos ou até mesmo fora do escopo dos requisitos.

Além dessas limitações, equipes de pesquisa e desenvolvimento voltadas a essas ferramentas, como a Open AI, apresentaram em análises destes recursos, a possibilidade de riscos e impactos que devem ser considerados com a utilização de seus modelos e suas ferramentas, tais como:

- O excesso de confiança que a ferramenta pode ocasionar: podem ser sugeridas soluções que podem estar corretas superficialmente, mas no final não representam a intenção e objetivo que o usuário solicita;
- Implicações de segurança: estes recursos podem gerar códigos vulneráveis em relação a *cyber* segurança. Além do fato, destas ferramentas serem treinadas com base em repositórios públicos, sendo possível a divulgação de dados sensíveis que estavam presentes nestes repositórios;
- Implicações legais: possui riscos de se realizar plagio através do código gerado;
- Impactos econômicos: podem provocar transformações significativas no mercado de trabalho em tecnologia. Por um lado, há o risco de substituição de determinadas cargos que realizam, cotidianamente, tarefas repetitivas e de baixa complexidade. Por outro lado, essas ferramentas podem reduzir a barreira de entrada de novos programadores no setor de desenvolvimento de software, tendendo a um

aumento da produtividade, aceleração no ciclo de desenvolvimento e surgimento de novos modelos de negócio (Chen,2021).

É importante para a análise destes recursos, a demonstração tanto dos possíveis impactos positivos quanto negativos, pois incentivam a pesquisa referente ao assunto e a busca por novos indícios de ambos os tipos. Como essas ferramentas ainda são recentes, seu impacto real permanece incerto. Portanto, investigações em diferentes áreas, tanto técnicas quanto sociais, são essenciais para uma compreensão mais ampla e fundamentada de suas implicações.

3.3 APRENDIZAGEM NA ÁREA DA PROGRAMAÇÃO: O QUE OCORRE E QUAIS SÃO SUAS DIFICULDADES

O ensino da ciência da computação enfrenta desafios constantes, especialmente, na formação de novos estudantes programadores. Apesar de muitas ferramentas e recursos, muitas dificuldades persistem, impactando neste processo de aprendizagem. A programação é um exemplo desta dificuldade, pois é uma atividade complexa que exige um esforço contínuo, uma abordagem específica e o desenvolvimento de diversas habilidades em vários aspectos.

É considerado, que o processo de obtenção dessas competências relacionadas a programação, seja algo tedioso, algo exigente, vindo de constantes tentativas com seus erros e acertos, se tornando, por vezes, desmotivador aos estudantes. Conhecer as sintaxes de uma linguagem e a própria lógica de programação pertencente ao desenvolvimento, são apenas os primeiros passos.

Portanto, para a resolução de problemas reais complexos é necessário um entendimento de conceitos abstratos e estratégicos. Sendo um obstáculo significativo para muitos alunos, que possuem dificuldades para entender conceitos abstratos de programação, muitas vezes devido a uma falta de consolidação de conhecimentos fundamentais, como o conhecimento sintático, o conhecimento conceitual e o conhecimento estratégico (Cheah,2020).

Além disso, as dificuldades podem decorrer da ausência de habilidades mais fundamentais, como a resolução de problemas e o raciocínio lógico, o que pode agravar as dificuldades durante o estágio inicial do aprendizado. De modo que, o conhecimento matemático, principalmente, lógico está diretamente relacionado ao

desempenho na programação, sendo que lacunas neste conhecimento podem prejudicar a capacidade de abstração e formulação de soluções computacionais.

Diante deste contexto, fatores psicológicos podem influenciar neste processo de aprendizagem. Para os alunos, a programação pode adquirir uma percepção de que possui uma excessiva dificuldade, uma percepção negativa, sendo influenciada, por comentários e experiências negativas de outros estudantes. Podendo levar à desmotivação e ao abandono da disciplina. Como resultado, a falta de incentivo nestes momentos, podem afetar o engajamento dos alunos, impactando seu desempenho, sua autoconfiança e comprometendo seu desempenho acadêmico (Cheah,2020).

De acordo com a pesquisa de Ismail, Ngah e Irfan Naufal (2010), são indicados quatro principais desafios no ensino de programação. (1) dificuldades na análise de problemas e compreensão de conceitos abstratos, (2) uso ineficaz de técnicas de apresentação para a solução de problemas, como pseudocódigo e fluxogramas, não adequadas para abordagens mais predominantes nas linguagens atuais como o ensino de programação orientada a objetos, (3) dificuldades enfrentadas pelos alunos devido ao uso de metodologias pouco eficazes para ensino de resolução de problemas e técnicas de codificação, principalmente, para o ensino de programação orientada a objetos e (4) baixo envolvimento dos alunos durante as aulas práticas, muitas vezes, decorrente de dificuldades técnicas e psicológicas.

A identificação e compreensão desses fatores é essencial para a melhoria do ensino da computação. Entretanto, as pesquisas relevantes sobre esse tópico não estão tão desenvolvidas no campo da educação em ciência da computação, quanto na educação em matemática e outras ciências. Portanto, é fundamental aprofundar os estudos sobre as dificuldades enfrentadas pelos alunos, suas concepções equivocadas e o desenvolvimento de estratégias pedagógicas mais eficazes e inovadoras, que respondam às demandas específicas do ensino de programação e consigam estimular o desenvolvimento de habilidades básicas necessárias a área. (Qian,2017).

Nesse contexto, ferramentas de assistência baseadas em inteligência artificial surgem como alternativas para apoiar o aprendizado. Além de correções de sintaxe, elas também podem contribuir para o aprendizado conceitual, facilitando a transcrição da linguagem natural para o código e abordando aspectos técnicos, desde a construção de estruturas convencionais à formulação de soluções mais complexas.

Estas ferramentas podem influenciar diretamente o aprendizado diante dos desafios apresentados, podendo facilitar este processo, mas também, gerar dependência e comprometer a autonomia do aluno no desenvolvimento lógico. Sendo necessário, que o uso desses recursos seja analisado de maneira criteriosa, considerando seus efeitos no desenvolvimento das habilidades cognitivas e formação de futuros programadores.

4 CONDUÇÃO DO EXPERIMENTO

Para investigar o impacto das ferramentas de assistência IA no aprendizado e no desenvolvimento de habilidades de programação, foi conduzido um experimento prático com estudantes da disciplina Engenharia de Software II (3º semestre) do curso de Análise e Desenvolvimento de Sistemas da FATEC Mogi das Cruzes.

Antes do experimento, metade das máquinas dos alunos participantes foi equipada com o CodeWhisperer, escolhido por ser gratuito. Garantiu-se que todos os alunos desse grupo tivessem uma conta ativa e que a ferramenta estivesse devidamente integrada ao Visual Studio Code (VS Code).

A divisão igualitária da turma, foi realizada em dois grupos, escolhidos, aleatoriamente, conforme os alunos chegavam:

- Grupo Experimental: utilizou a ferramenta, o CodeWhisperer durante o experimento;
- Grupo Controle: não utilizou a ferramenta.

Inicialmente, foi feita uma apresentação detalhada sobre o funcionamento e as potencialidades do CodeWhisperer. Em seguida, ambos os grupos responderam um questionário inicial, que coletou informações sobre os participantes e avaliou brevemente seus conhecimentos.

Na sequência, os alunos receberam uma tarefa preparada pelo professor da disciplina, já trabalhada em aula, mas que exigia a implementação de métodos simples usando expressões lambda[1]. Esta tarefa em específico propõe aos participantes concluírem a implementação de um sistema de controle de eleições para a Fatec.

[1] Expressões lambda são funções que não precisam ser declaradas com um nome, que podem ser definidas e utilizadas de maneira concisa, eliminando a necessidade de criar funções nomeadas para tarefas simples, possibilitando um código mais limpo e eficiente. No contexto deste experimento, foram empregadas para promover a aplicação de conceitos de programação funcional, exigindo que os alunos utilizassem essas estruturas em substituição a comandos tradicionais de repetição, como for e while.

O sistema já estava, parcialmente, desenvolvido por alunos anteriores, mas alguns métodos da classe Eleição precisavam ser finalizados. A atividade exigia a implementação dos seguintes métodos usando expressões lambda:

- quemGanhouParaCargo → Retorna os candidatos vencedores de um cargo, com base no número de votos;
- quantosVotosTeve → Retorna à quantidade de votos que um candidato recebeu;
- getRegistrosVotacaoSemRepeticao → Filtra registros de votação duplicados;
- getEleitoresPorSexo → Retorna todos os eleitores de um determinado sexo;
- vencedoresPorCargo → Retorna um mapa com os cargos e seus respectivos vencedores;

A principal restrição era de que os alunos não poderiam utilizar estruturas tradicionais de repetição, devendo resolver a atividade, exclusivamente, com expressões lambda.

O tempo para a realização foi limitado, e as condições de trabalho foram as seguintes:

- Grupo Experimental: pôde utilizar o CodeWhisperer para obter sugestões de código, além de realizar pesquisas na internet;
- Grupo Controle: realizou apenas pesquisas, sem assistência da ferramenta.

Durante o experimento, os participantes puderam tirar dúvidas com o professor sobre o exercício e, no caso do Grupo Experimental, também com os instrutores sobre o uso da ferramenta.

Após o experimento, ambos os grupos responderam o segundo questionário, que avaliou:

- O nível de compreensão, de ambos os grupos, sobre as expressões lambda, após a atividade;
- A percepção do Grupo Experimental sobre o uso do CodeWhisperer e sua influência no desenvolvimento da tarefa.

Além disso, para medir a aprendizagem dos participantes, os exercícios foram corrigidos e analisados junto às respostas dos questionários. Isso permitiu verificar o desempenho individual e comparar a qualidade das soluções entre os dois grupos,

identificando possíveis diferenças no aprendizado, entre aqueles que utilizaram a ferramenta de assistência e aqueles, que resolveram a tarefa sem suporte automatizado.

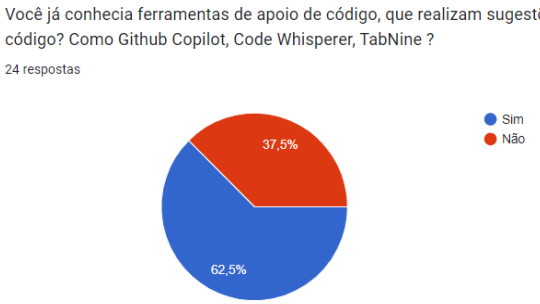
5 RESULTADOS E DISCUSSÕES

O objetivo deste capítulo é apresentar os dados coletados da pesquisa e realizar uma análise de seus resultados. Foram feitos comentários sobre os resultados com base na revisão bibliográfica apresentada ao longo de todo o trabalho.

5.1 RESULTADOS DO QUESTIONÁRIO INICIAL

Inicialmente, foi questionado a todos participantes do experimento sobre o conhecimento prévio sobre o uso de ferramentas de assistência IA e expressões lambda. Na primeira questão foi possível observar que a maior parte dos alunos conhecem alguma ferramenta de apoio de código (62,5%), contra uma minoria que não tinha esse conhecimento (37,5%), conforme mostrado no Gráfico 2.

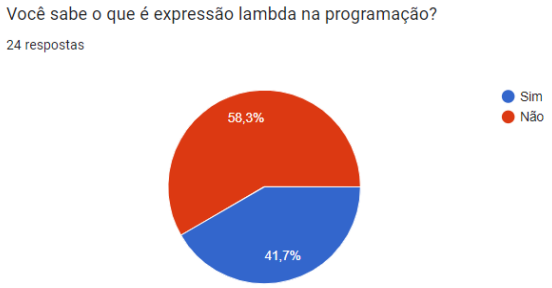
Gráfico 2 – Conhecimento de ferramentas de apoio.



Fonte: Elaborado pelos autores (2025).

A segunda questão refere-se ao conhecimento sobre o que é expressão lambda na programação, onde 58,3% não conheciam e 41,7% conheciam, como mostrado no Gráfico 3.

Gráfico 3 – Conhecimento de expressão lambda.



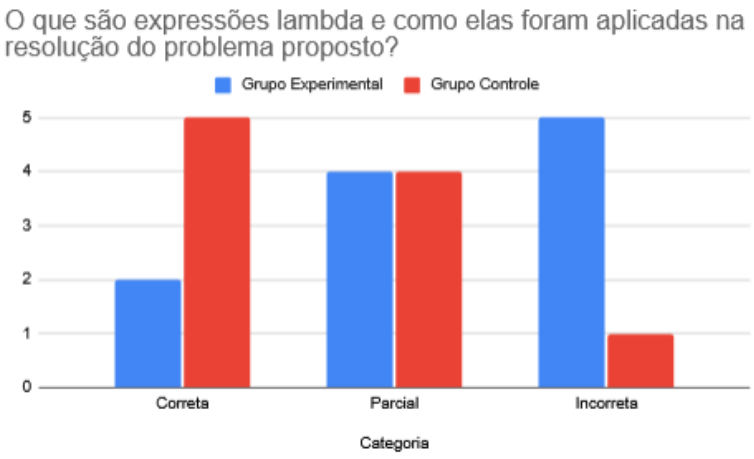
Fonte: Elaborado pelos autores (2025).

5.2 RESULTADOS DO QUESTIONÁRIO FINAL E IMPLEMENTAÇÕES

O segundo questionário, aplicado também a todos os participantes, teve como objetivo coletar informações após o exercício proposto na pesquisa (20 respostas totais, sendo 11 do Grupo experimental e 9 do grupo controle). A primeira questão confere se os participantes aprenderam o que são expressões lambda e a compreensão de sua aplicação no exercício.

O Grupo Controle demonstrou uma compreensão mais sólida do conceito de expressões lambda, com 5 respostas corretas contra apenas 2 do Grupo Experimental. Ambos os grupos tiveram 4 respostas parciais (respostas que demonstram algum conhecimento, porém não estão completas ou totalmente corretas), O Grupo Experimental apresentou um número alto de respostas incorretas em comparação com o Grupo de Controle, como pode ser observado no Gráfico 4.

Gráfico 4 – Entendimento e aplicação das expressões lambda.

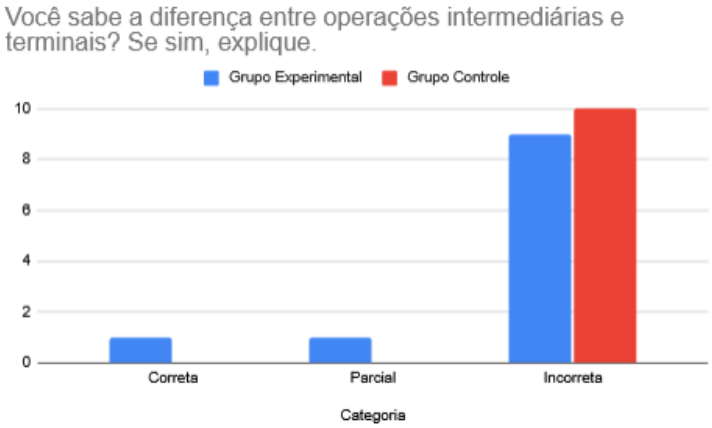


Fonte: Elaborado pelos autores (2025).

A segunda questão avalia o conhecimento específico sobre as expressões lambdas. Foi questionado se conhecem a diferença de operações intermediárias e terminais, e caso sim, foi pedido para explicar essa diferença.

Nenhum aluno do Grupo Controle alcançou uma resposta correta, enquanto 1 aluno do Grupo Experimental conseguiu. No entanto, o Grupo Experimental também apresentou uma resposta parcial, enquanto o Grupo Controle não apresentou nenhuma. A grande maioria dos alunos de ambos os grupos (9 do Experimental e 10 do Controle) não souberam responder corretamente a esta questão, indicando uma dificuldade geral com este conceito, como mostrado no Gráfico 5.

Gráfico 5 – Diferença de operações.

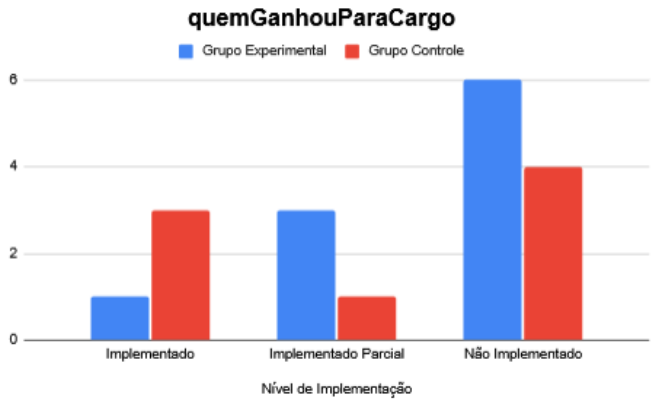


Fonte: Elaborado pelos autores (2025).

A partir das implementações realizadas pelos grupos de alunos, foi possível levantar os dados referentes a cinco métodos avaliados: **quemGanhouParaCargo**, **quantosVotosTeve**, **getRegistrosVotacaoSemRepeticao**, **getEleitoresPorSexo** e **vencedoresPorCargo**.

No método **quemGanhouParaCargo**, o Grupo Experimental apresentou 1 participante que implementou corretamente, 3 com implementação parcial e 6 que não implementaram. Já o Grupo Controle teve desempenho superior, com 3 implementações corretas, 1 parcial e 4 ausentes. O gráfico 6 mostra tal diferença.

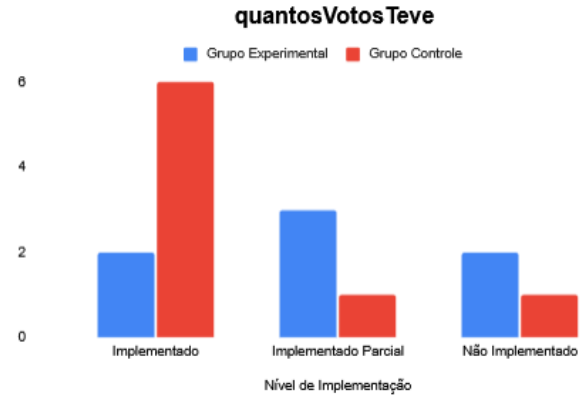
Gráfico 6 – quemGanhouParaCargo.



Fonte: Elaborado pelos autores (2025).

No método **quantosVotosTeve**, o Grupo Experimental contou com 2 implementações corretas, 3 parciais e 2 não implementadas. O Grupo Controle, por sua vez, obteve 6 implementações corretas, 1 parcial e apenas 1 não implementada, indicando um desempenho ,significativamente, melhor nesse caso, como pode ser visto no Gráfico 7.

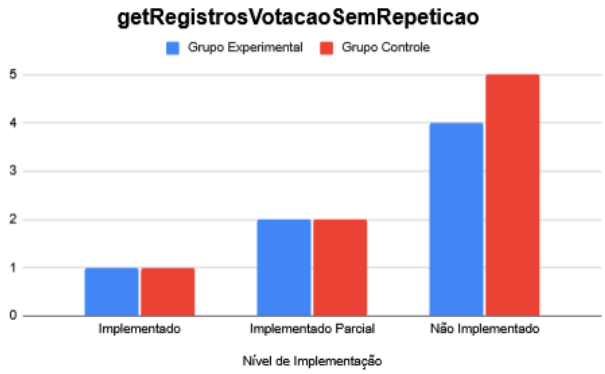
Gráfico 7 – quantosVotosTeve.



Fonte: Elaborado pelos autores (2025).

Para o método **getRegistrosVotacaoSemRepeticao**, ambos os grupos tiveram desempenho semelhante: o Grupo Experimental teve 1 implementação correta, 2 parciais e 4 não implementadas, enquanto o Grupo Controle contou com 1 correta, 2 parciais e 5 não implementadas, como mostrado no Gráfico 8.

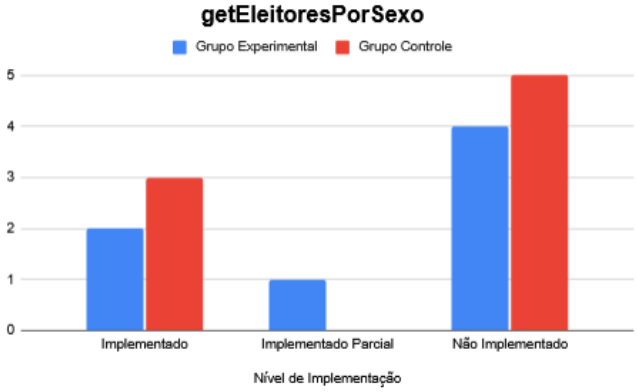
Gráfico 8 – getRegistrosVotacaoSemRepeticao



Fonte: Elaborado pelos autores (2025).

No método **getEleitoresPorSexo**, o Grupo Experimental apresentou 2 implementações corretas, 1 parcial e 4 não implementadas. O Grupo Controle teve um leve desempenho superior, com 3 implementações corretas e 5 não implementadas, sem registros de implementação parcial, como mostrado no Gráfico 9.

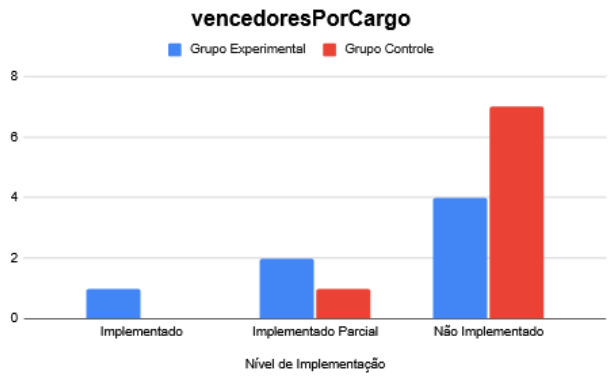
Gráfico 9 – getEleitoresPorSexo.



Fonte: Elaborado pelos autores (2025).

Por fim, no método **vencedoresPorCargo**, o Grupo Experimental teve 1 implementação correta, 2 parciais e 4 não implementadas. Já o Grupo Controle não apresentou nenhuma implementação correta, apenas 1 parcial e 7 não implementadas, indicando maior dificuldade nesse método por parte desse grupo, como visto no Gráfico 10.

Gráfico 10 – vencedoresPorCargo.



Fonte: Elaborado pelos autores (2025).

A partir dos dados obtidos nos questionários e nas implementações dos métodos solicitados, é possível refletir sobre os impactos da utilização da ferramenta CodeWhisperer no processo de aprendizagem de expressões lambda em Java.

Os dados demonstram que, embora a ferramenta tenha oferecido suporte ao desenvolvimento prático, o grupo que utilizou o CodeWhisperer (Grupo Experimental) apresentou um desempenho inferior em relação à compreensão conceitual das expressões lambda quando comparado ao Grupo Controle. Esta diferença sugere que a presença de uma ferramenta de assistência pode ter gerado uma dependência excessiva, como discutido por Pearce (2021), fazendo com que os alunos se concentrassem mais em obter soluções funcionais do que em compreender sua estrutura e lógica.

Adicionalmente, observou-se um número expressivo de resoluções parciais no Grupo Experimental, o que indica que a ferramenta pode acelerar a implementação de soluções em estudantes que já possuem certo domínio da linguagem e da lógica. Entretanto, para aqueles com pouca base conceitual, o assistente não foi suficiente para viabilizar soluções completas. Já no Grupo Controle, os resultados foram mais polarizados (a maioria dos estudantes ou sabia como resolver corretamente ou não conseguia avançar) refletindo com mais clareza o nível real de compreensão. Esses achados sugerem que a ferramenta tende a beneficiar apenas aqueles com conhecimento prévio, sem contribuir de forma significativa para quem ainda está em processo de construção das bases lógicas da programação.

No que se refere à questão sobre operações intermediárias e terminais (um tema de maior complexidade), a maioria dos estudantes de ambos os grupos não conseguiu responder corretamente, o que evidencia uma dificuldade comum entre os alunos na

assimilação de conceitos mais abstratos, independentemente, do uso da ferramenta. Isso corrobora com os desafios relatados por Ismail, Ngah e Irfan Naufal (2010), que destacam a dificuldade dos estudantes em internalizar conceitos estratégicos e abstratos no ensino de programação.

Dessa forma, os resultados evidenciam uma tensão entre a facilitação prática oferecida pela IA e o desenvolvimento efetivo do conhecimento conceitual, o que aponta para a necessidade de uma mediação pedagógica criteriosa no uso dessas ferramentas em ambientes educacionais. Como argumentado por Chen (2021), o uso dessas ferramentas deve ser acompanhado de estratégias que incentivem a reflexão sobre o código gerado, promovendo assim um aprendizado ativo e consciente.

6 CONSIDERAÇÕES FINAIS

O avanço das ferramentas de assistência em programação impulsionadas por inteligência artificial, como o CodeWhisperer, apresenta benefícios significativos, como o aumento da velocidade e da eficiência no processo de codificação. No entanto, seu uso também levanta preocupações, especialmente em relação à dependência tecnológica e à compreensão superficial de conceitos fundamentais.

Os resultados obtidos revelam que os alunos que utilizaram a ferramenta apresentaram desempenho inferior na compreensão conceitual de expressões lambda, em comparação ao grupo que não teve acesso ao assistente. A análise demonstrou que o grupo controle, mesmo sem suporte automatizado, foi capaz de formular mais respostas corretas e completas, tanto nos questionários quanto na implementação dos métodos propostos.

Isso indica que, no contexto educacional, o uso de ferramentas de assistência pode gerar um atalho cognitivo, desviando o foco da internalização dos conceitos para a simples resolução das tarefas. A dependência causada por esse tipo de recurso, como apontado na literatura, tende a favorecer estudantes que já possuem certa familiaridade com os fundamentos, mas oferece pouca contribuição para aqueles que ainda estão em processo de construção do raciocínio lógico e da compreensão abstrata.

Portanto, conclui-se que o uso de assistentes de codificação por IA no ensino de programação deve ser, cuidadosamente, planejado e vinculado a estratégias pedagógicas que priorizem a compreensão profunda dos conceitos. O CodeWhisperer

pode ser um recurso complementar valioso, desde que seu uso não comprometa o desenvolvimento das habilidades cognitivas e da autonomia dos estudantes. Para contextos educacionais, recomenda-se que essas ferramentas sejam utilizadas como apoio supervisionado, com ênfase na reflexão sobre o código sugerido e na construção ativa do conhecimento.

REFERÊNCIAS

BOSSE, Y.; GEROSA, M. A. **Why is programming so difficult to learn? Patterns of difficulties related to programming learning mid-stage**. ACM SIGSOFT Software Engineering Notes, v. 41, n. 6, p. 1-6, 2017. Disponível em: https://doi.org/10.1145/3011286.3011301.

BRASSCOM – ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO. **Demanda de talentos em TIC e estratégia.** Brasília: Brasscom, 2021. Disponível em: https://brasscom.org.br/pdfs/demanda-de-talentos-em-tic-e-estrategia-tcem/.

BROWN, Tom et al. **Language models are few-shot learners**. 22 maio 2020. Disponível em: https://arxiv.org/abs/2005.14165.

CHEAH, Chin Soon. **Factors contributing to the difficulties in teaching and learning of computer programming: a literature review**. Contemporary Educational Technology, v. 12, n. 2, p. ep272, 8 maio 2020. Disponível em: https://doi.org/10.30935/cedtech/8247.

CHEN, Mark et al. **Evaluating large language models trained on code**. 14 jul. 2021. Disponível em: https://arxiv.org/abs/2107.03374.

ISMAIL, Mohd Nasir; NGAH, Nor Azilah; Irfan Naufal. **Instructional strategy in the teaching of computer programming: A need assessment analyses.** The Turkish Online Journal of Educational Technology, v. 9, n. 2, 2010. Disponível em: http://tojet.net/articles/v9i2/9214.pdf.

PEARCE, Hammond et al. **Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions**. 16 dez. 2021. Disponível em: https://arxiv.org/abs/2108.09293.

QIAN, Yizhou; LEHMAN, James. **Students' misconceptions and other difficulties in introductory programming**. ACM Transactions on Computing Education, v. 18, n. 1, p. 1–24, 6 dez. 2017. Disponível em: https://doi.org/10.1145/3077618.

ROBINS, A. **Novice programmers and introductory programming.** In: FINCHER, S.; ROBINS, A. (org.). The Cambridge handbook of computing education research. Cambridge: Cambridge University Press, 2019. (Cambridge Handbooks in Psychology), p. 327-376.