Inteligência Artificial e Acessibilidade: Uma Experiência de Inclusão para Programadores Cegos em Ambientes de Desenvolvimento

Naiara Silva dos Santos[1,2], **Danyele de Oliveira Santana**[1,3], **Claudia Pinto Pereira**[1]

[1]Programa de Pós Graduação em Ciência da Computação – Universidade Estadual de Feira de Santana (UEFS) 44036-900 – Feira de Santana – BA – Brazil

[2]Departamento de Ciências e Tecnologias – Universidade Estadual do Sudoeste da Bahia – Jequié, BA.

[3]Instituto Federal de Educação, Ciência e Tecnologia Baiano – Bom Jesus da Lapa- BA.

nssantos@uesb.edu.br, danyele.santana@ifbaiano.edu.br, claudiap@uefs.br

Abstract. *This article explores the use of Artificial Intelligence (AI) tools to enhance accessibility and productivity in teaching programming to blind students. Using environments such as VS Code and Repl.it, the intervention assessed challenges and successes through questionnaires and interviews with blind programmers, comparing problem-solving with and without Artificial Intelligence. The results identified barriers, such as navigation difficulties and limitations in screen readers, alongside suggestions for improvements. This report provides insights for educators to replicate or adapt the intervention, encouraging the use of AI for more inclusive programming education.*

Resumo. *Este artigo explora o uso de ferramentas de Inteligência Artificial (IA) para aumentar a acessibilidade e a produtividade no ensino de programação para estudantes cegos. Utilizando ambientes como VS Code e Repl.it, a intervenção avaliou desafios e sucessos por meio de questionários e entrevistas com programadores cegos, comparando a resolução de problemas com e sem Inteligência Artificial. Os resultados apontaram barreiras, como dificuldades de navegação e limitações em leitores de tela, além de sugestões para melhorias. Este relato oferece lições para que educadores possam replicar ou adaptar a intervenção, incentivando o uso de IA para um ensino de programação mais inclusivo.*

1. Introdução

A inclusão digital no ensino de programação continua sendo um desafio significativo, especialmente para estudantes com deficiência visual. A programação, que frequentemente depende de interfaces gráficas e ferramentas complexas, apresenta barreiras substanciais para esse público, devido à ausência de diretrizes padronizadas e limitações em tecnologias assistivas como leitores de tela [Sribunruangrit et al. 2004]. Essas dificuldades impactam diretamente a acessibilidade e a produtividade de estudantes cegos, restringindo sua participação plena no aprendizado de habilidades essenciais para o mundo contemporâneo [Sribunruangrit et al. 2004].

Nesse contexto, o avanço da Inteligência Artificial (IA) emerge como uma oportunidade promissora. Ferramentas baseadas em IA, como assistentes de código, sistemas de autocompletar e soluções de depuração, oferecem potencial para mitigar barreiras, fornecendo suporte em tempo real, melhorando a acessibilidade e promovendo uma experiência educacional mais equitativa e inclusiva [Veiderma Holmberg 2021]. Esse potencial torna o tema altamente relevante para o ensino de programação.

A crescente presença da programação em diversas áreas do conhecimento exige que ambientes de desenvolvimento sejam acessíveis a todos, incluindo programadores cegos. Nos últimos anos, avanços em tecnologias assistivas, como leitores de tela e editores de código adaptáveis, têm melhorado a experiência desses usuários [Pandey et al. 2021]. No entanto, barreiras persistem, especialmente na depuração de código e na compreensão de mensagens de erro, dificultando a independência de programadores cegos [Mountapmbeme et al. 2022]. Nesse cenário, a IA surge como uma ferramenta potencial para reduzir essas limitações, oferecendo suporte automatizado e ampliando a autonomia desses usuários. Entretanto, a aplicação da IA na acessibilidade para programadores cegos ainda é pouco explorada, como aponta [Pandey 2023], que identificou uma escassez de pesquisas investigando o uso da IA para aprimorar a experiência de desenvolvedores com deficiência visual. Isso torna essencial a investigação sobre o impacto real dessas tecnologias no aprendizado e na produtividade desses programadores.

Este relato de experiência descreve uma intervenção educacional que explorou o uso de tecnologias assistivas e ferramentas de IA para apoiar estudantes cegos no aprendizado de programação. Realizada em ambientes de desenvolvimento como Visual Studio Code (VS Code), Repl.it e Google Colab, a intervenção permitiu que os participantes resolvessem problemas de programação, tanto de forma autônoma quanto com o suporte de ferramentas baseadas em IA. A experiência foi estruturada para analisar comparativamente as abordagens com e sem o uso de IA, buscando compreender como essas tecnologias podem impactar a eficiência, a produtividade e a acessibilidade no ensino de programação.

Os objetivos deste trabalho são, portanto, analisar os desafios enfrentados por estudantes cegos na programação, avaliar a eficácia de ferramentas de IA para superar essas barreiras e refletir sobre o impacto educacional dessa abordagem. Além disso, o relato busca oferecer subsídios para que outros educadores possam adaptar ou replicar essa intervenção em seus próprios contextos, promovendo um ensino de programação mais inclusivo. Ao longo do texto, detalharemos as atividades realizadas, os desafios encontrados, os resultados alcançados e as lições aprendidas, com o intuito de contribuir para a construção de práticas pedagógicas mais acessíveis e equitativas no ensino de computação.

2. Fundamentação Teórica

A inclusão digital é uma prioridade no contexto contemporâneo, especialmente em áreas como a programação, nas quais a habilidade de desenvolver soluções tecnológicas é cada vez mais essencial [Pandey et al. 2021]. No entanto, estudantes cegos enfrentam desafios significativos no aprendizado de programação, devido à dependência de interfaces gráficas e à ausência de ferramentas acessíveis plenamente integradas com tecnologias assistivas [Albusays et al. 2017]. Este trabalho é fundamentado em teorias sobre acessibi-

lidade [Potluri et al. 2018], o uso de Inteligência Artificial (IA) na educação e práticas pedagógicas inovadoras [Alizadehsani et al. 2022], que sustentam a proposta de intervenção descrita.

A acessibilidade, conforme definido pelas diretrizes da *Web Accessibility Initiative* (WAI), busca garantir que pessoas com deficiência tenham acesso igualitário a tecnologias e conteúdos digitais [Khasawneh 2023]. No caso de estudantes cegos, leitores de tela, como NVDA e JAWS, são ferramentas indispensáveis para a interação com ambientes de programação [Amin et al. 2024]. É fundamental ressaltar que a possibilidade de pessoas cegas programarem autonomamente só se tornou viável devido ao auxílio dos leitores de tela. No entanto, a integração desses leitores com plataformas de desenvolvimento ainda apresenta lacunas, especialmente na interpretação de mensagens de erro, sugestões de código e estruturas visuais [dos Santos Soares et al. 2024]. Estudos na área de inclusão digital [dos Santos Soares et al. 2024], [Khasawneh 2023], [Pandey et al. 2021] apontam que a personalização dessas tecnologias, alinhada a práticas pedagógicas específicas, pode minimizar barreiras e promover maior autonomia no aprendizado.

A Inteligência Artificial tem se mostrado um recurso promissor na educação, permitindo a personalização de processos de ensino e o suporte em tempo real para tarefas complexas [Chemnad and Othman 2024]. Ferramentas como GitHub Copilot e ChatGPT são exemplos de como a IA pode ser utilizada para apoiar programadores, fornecendo sugestões de código, identificando erros e otimizando algoritmos [Wermelinger 2023]. Segundo [Papert 1980], a tecnologia educacional deve atuar como mediadora entre o estudante e o conhecimento, facilitando a construção do aprendizado. Nesse sentido, a IA pode desempenhar um papel fundamental ao ajudar estudantes cegos a superar barreiras técnicas, promovendo uma experiência mais inclusiva e equitativa.

Segundo [Inhelder and Piaget 1976], o aprendizado ocorre de forma mais eficaz quando o estudante tem a oportunidade de interagir ativamente com o conteúdo. Nesse contexto, a programação, por sua natureza prática e iterativa, oferece um ambiente ideal para a aplicação dessas teorias. No entanto, para que estudantes cegos possam participar plenamente, é necessário que as ferramentas sejam adaptadas às suas necessidades, proporcionando um ambiente acessível e inclusivo.

Estudantes cegos enfrentam múltiplas barreiras ao aprender programação, que vão além da complexidade lógica e sintática da atividade [Petrausch and Loitsch 2017]. A interação com ambientes de desenvolvimento, frequentemente, exige navegação visual intensa, dificultando a leitura de códigos longos, a identificação de erros e a depuração eficiente com leitores de tela [Albusays et al. 2017]. Além disso, a falta de integração adequada entre tecnologias assistivas e editores de código impacta negativamente a experiência desses estudantes, tornando tarefas simples, como acompanhar sugestões de autocompletar ou interpretar mensagens de erro, mais demoradas e propensas a falhas [Seo and Rogge 2023]. Do ponto de vista pedagógico, essas dificuldades podem comprometer a autonomia no aprendizado e aumentar a dependência de suporte externo. Diante desse cenário, torna-se essencial investigar soluções que mitiguem essas barreiras, permitindo que estudantes cegos tenham um acesso mais equitativo ao ensino de programação, seja por meio de adaptações metodológicas ou pelo uso de tecnologias emergentes, como a Inteligência Artificial.

Trabalhos anteriores na área de ensino de programação para pessoas com deficiência visual destacam os desafios relacionados à acessibilidade e à navegação em ambientes de desenvolvimento. Estudos como o de [Petrausch and Loitsch 2017] analisaram a acessibilidade da IDE Eclipse para pessoas com deficiência visual, identificando desafios de usabilidade por meio de questionários e propondo melhorias, como tutoriais inclusivos e diretrizes para desenvolvimento de software acessível. Além disso, pesquisas como [Seo and Rogge 2023] avaliaram a eficácia do GitHub Copilot no suporte a programadores cegos, conduzindo uma intervenção prática que demonstrou a capacidade da ferramenta de reduzir significativamente o tempo de depuração e melhorar a produtividade. No entanto, os autores também apontaram limitações relacionadas à integração dessas tecnologias com leitores de tela, reforçando a necessidade de avanços para um uso mais fluido. Diferentemente desses estudos, este trabalho avança ao propor uma metodologia que compara diretamente as experiências de programação com e sem o uso de IA, respeitando a escolha de IDEs e linguagens pelos participantes. Essa abordagem permite explorar como diferentes ferramentas se adaptam às preferências e necessidades de programadores cegos.

3. Metodologia

Este relato de experiência utilizou uma abordagem experimental e qualitativa, com entrevistas e questionários como instrumentos principais para avaliar as barreiras enfrentadas, as soluções existentes e o impacto de ferramentas assistidas por Inteligência Artificial (IA) no aprendizado de programação para estudantes cegos. A metodologia foi concebida para permitir uma análise comparativa entre o desenvolvimento de algoritmos com e sem o suporte de IA, além de identificar estratégias e propor melhorias para um ensino mais inclusivo e acessível. O trabalho buscou oferecer informações suficientes para que outros educadores possam replicar ou adaptar a intervenção em seus próprios contextos.

A intervenção foi realizada com quatro participantes cegos, com idades entre 25 e 53 anos, oriundos de diferentes regiões do Brasil, incluindo Bahia, São Paulo e Pernambuco. Todos apresentavam cegueira total, sendo dois participantes cegos de nascença e dois que adquiriram a deficiência ao longo da vida. O grupo era composto por três homens e uma mulher, com experiências variadas em programação. Dois participantes estavam cursando Sistemas de Informação, enquanto os outros eram egressos de cursos na área de Tecnologia da Informação, e estão cursando pós-graduação em Arquitetura e Desenvolvimento Java.

O experimento foi conduzido de forma remota, utilizando a plataforma Google Meet para facilitar a interação e gravação das atividades, com o consentimento livre e esclarecido dos participantes. Os experimentos foram realizados em seus próprios computadores, configurados com ferramentas e leitores de tela que utilizavam rotineiramente, como NVDA e JAWS. O pesquisador observou as sessões em tempo real e gravou a tela de cada transmissão para análise posterior. Essas configurações garantiram que o ambiente fosse o mais próximo possível do cotidiano de cada participante, permitindo uma análise realista das dificuldades enfrentadas e do impacto das ferramentas de IA no suporte ao aprendizado e produtividade. Além disso, o uso remoto da plataforma Google Meet e das gravações assegurou a coleta de dados detalhada e completa para análise qualitativa e quantitativa, respeitando as condições éticas do estudo.

A intervenção foi estruturada em cinco etapas principais, sendo a **primeira** dedicada à apresentação e contextualização do experimento. Durante os 15 minutos iniciais, os participantes foram informados sobre os objetivos da pesquisa, que visava avaliar a acessibilidade e a produtividade de ferramentas de IA no suporte à programação para pessoas cegas. Foram explicadas as regras do estudo, incluindo a distinção entre as duas atividades: a primeira sem o uso de IA e a segunda com o suporte dessas ferramentas. Para garantir a integridade do experimento, foi estabelecido que, caso a IDE utilizada possuísse funcionalidades de IA já integradas, os participantes deveriam escrever o código inicialmente em um bloco de notas, assegurando que fosse desenvolvido do zero. Além disso, foram apresentados os leitores de tela e as ferramentas de IA que poderiam ser utilizadas, como o NVDA, JAWS e GitHub Copilot. A metodologia adotada permitiu que os participantes gerenciassem livremente o tempo para a execução das tarefas, garantindo que pudessem concluir as atividades sem interferências externas, enquanto o pesquisador permanecia disponível apenas para suporte técnico quando solicitado.

Na **segunda etapa**, os participantes desenvolveram um algoritmo para organizar uma sequência numérica em ordem crescente, sem o auxílio de IA. Eles podiam optar por trabalhar com uma lista predefinida ou permitir que o usuário inserisse os valores manualmente. Durante essa fase, foi permitido o uso de pesquisas online para consultas técnicas, desde que não envolvessem soluções baseadas em IA. O objetivo era avaliar a experiência de programação manual, analisando os desafios enfrentados, o tempo necessário para a implementação e os tipos de erros cometidos ao longo do processo.

Na **terceira etapa**, os participantes resolveram novamente o mesmo problema que envolvia a ordenação de uma lista de números, mas desta vez utilizando ferramentas assistidas por IA, como sistemas de autocompletar e geração de código, incluindo o ChatGPT. Com isso, puderam tanto revisar e aprimorar o código que haviam produzido na etapa anterior quanto explorar novas formas de implementação sugeridas pela IA. O foco dessa etapa foi explorar como a IA poderia ajudar na identificação e correção de erros do algoritmo já implementado na etapa anterior. Os participantes foram incentivados a usar a IA para analisar o código produzido, identificar problemas e propor soluções automatizadas. O objetivo principal era avaliar como as ferramentas baseadas em IA impactavam a eficiência na depuração, o tempo de execução e a qualidade geral dos códigos ajustados. Embora o tempo para a realização das atividades fosse gerenciado livremente pelos participantes, ele foi monitorado para análise comparativa, considerando que a dificuldade estava não na criação do algoritmo, mas na melhoria e correção do código com o suporte da IA.

Após as atividades práticas, foi conduzida uma **quarta etapa** de coleta de informações e discussão, com duração de 30 minutos. Foram realizadas entrevistas semiestruturadas e aplicados questionários acessíveis para registrar as percepções dos participantes sobre as atividades realizadas. Essa etapa permitiu compreender as barreiras específicas enfrentadas, os benefícios percebidos no uso da IA e as dificuldades relacionadas à integração de leitores de tela com as ferramentas de programação.

A **última etapa** consistiu na proposição de melhorias, com duração de 15 minutos. Os participantes foram incentivados a sugerir mudanças que poderiam facilitar a integração entre leitores de tela e ferramentas de IA. Essa etapa forneceu subsídios importantes para o aperfeiçoamento das tecnologias utilizadas, com foco em garantir aces-

sibilidade e usabilidade mais eficientes.

Os dados coletados foram analisados qualitativamente, a partir dos relatos dos participantes, e quantitativamente, comparando o tempo de execução das tarefas e o número de erros corrigidos nas atividades com e sem IA, ver tabela 2. Essa abordagem permitiu identificar as limitações e os benefícios das ferramentas adotadas, além de fornecer dados para a replicação e adaptação dessa prática em outros contextos educacionais.

Este trabalho foi desenvolvido em conformidade com os princípios éticos aplicáveis à pesquisa envolvendo seres humanos. O projeto foi submetido para o Comitê de Ética em Pesquisa (CEP), registrado na Plataforma Brasil sob o número de CAAE: 81692324.0.0000.0053, com data de submissão em 19/07/2024. Todos os participantes foram devidamente informados sobre os objetivos do estudo, os procedimentos adotados e os possíveis benefícios e riscos associados à participação. O Termo de Consentimento Livre e Esclarecido foi obtido de todos os envolvidos antes do início das atividades, assegurando o respeito à autonomia e aos direitos dos participantes.

4. Relato de Experiência

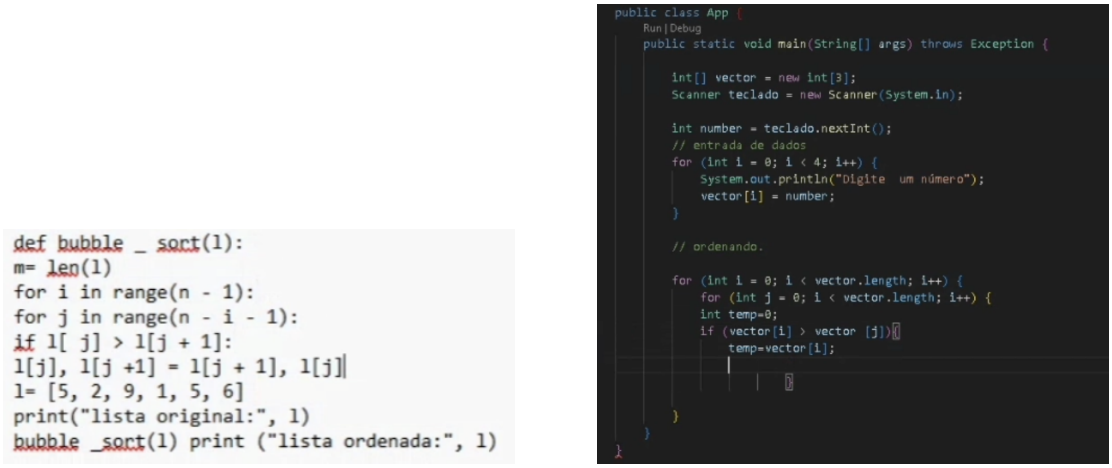
A intervenção foi desenvolvida ao longo de cinco etapas, conforme explicitado na metodologia, envolvendo atividades práticas, reflexões e coleta de dados. O relato detalha como as ações foram implementadas na prática, destacando as interações dos participantes, decisões tomadas durante o processo e os desafios enfrentados.

A experiência começou com uma apresentação e contextualização. Durante essa etapa inicial, os participantes foram recepcionados e introduzidos aos objetivos do estudo. O ambiente foi preparado para que os participantes se sentissem à vontade e tivessem clareza sobre as atividades. Nesse momento, as ferramentas que seriam utilizadas foram configuradas conforme as preferências individuais, e os leitores de tela foram testados para garantir pleno funcionamento. Decisões importantes foram tomadas, como permitir o uso de ferramentas com as quais os participantes já estavam familiarizados, como o NVDA e o JAWS, evitando uma curva de aprendizado desnecessária para novas tecnologias.

Na etapa seguinte, os participantes enfrentaram o desafio de resolver um problema de programação sem o auxílio de IA. Cada participante teve a liberdade de escolher a linguagem de programação que desejava utilizar para a resolução do problema, permitindo que trabalhassem com tecnologias com as quais já estavam familiarizados. A tarefa proposta envolvia a implementação de um algoritmo de ordenação para uma lista de números, o que revelou barreiras técnicas e cognitivas específicas para cada linguagem escolhida.

O Participante P1 (Tabela 2), ao programar em Python, enfrentou dificuldades com a indentação do código (Figura 1(a)), resultando em erros persistentes de execução que demandaram tempo significativo para serem corrigidos. Outro participante, o P2 (Tabela 2), como pode ser visto na Figura 1 (b), utilizando Java, cometeu erros que comprometeram a execução do algoritmo de ordenação. Primeiramente, a declaração do vetor foi feita com um tamanho fixo de 3 posições, mas a estrutura de repetição tenta acessar um índice fora dos limites, gerando uma exceção de *ArrayIndexOutOfBoundsException*. Além disso, na entrada de dados, a variável *number* é lida antes da iteração e atribuída diretamente a *vector[i]*, sem atualizar seu valor dentro do laço, resultando na repetição

do mesmo número no vetor. No laço de ordenação, há um erro na variável de controle do segundo *for*, onde *i* é reutilizado em vez de *j*, o que impede a comparação correta entre os elementos do vetor. Por fim, a lógica de troca de valores dentro do *if* não está completa, pois o elemento maior é armazenado em *temp*, mas não ocorre a reatribuição correta aos índices do vetor, impossibilitando a ordenação correta dos números. Esses erros refletem dificuldades tanto na manipulação de estruturas de dados quanto na lógica de comparação e troca de valores dentro do algoritmo.



```
def bubble _ sort(1):
m= len(1)
for i in range(n - 1):
for j in range(n - i - 1):
if 1[ j] > 1[j + 1]:
1[j], 1[j +1] = 1[j + 1], 1[j]
1= [5, 2, 9, 1, 5, 6]
print("lista original:", 1)
bubble _sort(1) print ("lista ordenada:", 1)
```

```
public class App {
    Run|Debug
    public static void main(String[] args) throws Exception {

        int[] vector = new int[3];
        Scanner teclado = new Scanner(System.in);

        int number = teclado.nextInt();
        // entrada de dados
        for (int i = 0; i < 4; i++) {
            System.out.println("Digite  um número");
            vector[i] = number;
        }

        // ordenando.

        for (int i = 0; i < vector.length; i++) {
            for (int j = 0; i < vector.length; i++) {
            int temp=0;
            if (vector[i] > vector [j]){
                temp=vector[i];
                |
                    }

        }
    }
}
```

(a) Tela do Participante 1 **(b)** Tela do Participante 2

Figura 1. Telas dos Participantes P1 e P2

Apesar de permitido o uso de pesquisas online para auxiliar na resolução do problema, alguns participantes relataram dificuldades adicionais. Especificamente, os leitores de tela nem sempre conseguiam interpretar corretamente os conteúdos de determinados sites, comprometendo a eficiência das buscas por soluções e aumentando a dependência de estratégias manuais para depuração e correção de erros. Essas dificuldades destacaram a importância de melhorar a acessibilidade tanto em ambientes de desenvolvimento quanto em plataformas externas de suporte.

Com a introdução das ferramentas de IA na etapa seguinte, observamos mudanças significativas. Os participantes utilizaram sistemas como ChatGPT e Repl.it, que proporcionaram suporte durante a programação. Os resultados foram expressivos: códigos que anteriormente apresentavam erros estruturais foram corrigidos em segundos, e a geração de algoritmos funcionais tornou-se mais ágil. No entanto, mesmo com essas melhorias, surgiram novos desafios. As sugestões de código apresentadas em formato de texto fantasma (*ghost text*) não eram automaticamente lidas pelos leitores de tela, exigindo ajustes manuais que, em alguns casos, reduziram a fluidez do processo.

A repetição do mesmo exercício nas etapas sem IA e com IA foi uma escolha intencional para permitir uma comparação direta entre os métodos, focando na eficiência da codificação, nos desafios enfrentados e na correção de erros. Embora a familiaridade com o problema pudesse representar uma vantagem na segunda tentativa, essa abordagem isolou o impacto da IA, garantindo que as diferenças observadas fossem atribuídas ao uso da tecnologia assistiva e não à complexidade do desafio. Caso o objetivo fosse avaliar a resolução de problemas inéditos, exercícios distintos seriam mais adequados. No entanto, como o foco do estudo é medir o impacto da IA na otimização, depuração e eficiência da

codificação, a repetição permitiu uma análise mais precisa das melhorias proporcionadas. Os dados coletados (tempo de execução, erros corrigidos, percepções dos participantes) demonstraram que, apesar de ser o mesmo exercício, a IA influenciou significativamente a abordagem e o desempenho dos participantes. Ainda assim, surgiram novos desafios relacionados ao uso da IA no processo de programação, como a necessidade de adaptação às sugestões de código geradas automaticamente, dificuldades na interação com o leitor de tela para interpretar essas sugestões e ajustes na lógica para integrar corretamente os trechos sugeridos. Essas questões evidenciam tanto os benefícios da IA quanto as adaptações necessárias para maximizar seu potencial na acessibilidade e produtividade dos programadores cegos.

A etapa de coleta de informações e discussão foi especialmente rica. Durante entrevistas e questionários, os participantes refletiram sobre a experiência, destacando aspectos positivos, como a rapidez e a precisão proporcionadas pela IA, mas também apontaram limitações importantes. Um tema recorrente foi a dificuldade de integração entre leitores de tela e ferramentas de IA, reforçando a necessidade de avanços nesse campo. Conforme mencionado por P2: *"A IA já me ajudou muito, mas algumas sugestões simplesmente não são lidas pelo leitor de tela. Precisei explorar a tela manualmente para entender o que estava acontecendo"*.

Além disso, a experiência com IA ampliou a confiança dos participantes em suas habilidades de programação. Um deles, P1, afirmou: *"Com a ajuda da IA, eu consegui corrigir erros que antes eu levaria horas para encontrar. Isso realmente aumenta a minha confiança no que posso fazer como programador"*. Essas reflexões mostram que a tecnologia pode não apenas superar barreiras, mas também empoderar os usuários, promovendo maior autonomia e eficiência no aprendizado de programação.

Na etapa final, dedicada à proposição de melhorias, os participantes sugeriram mudanças práticas que poderiam facilitar o uso das ferramentas por outros estudantes cegos. Entre as ideias apresentadas estavam a inclusão de notificações sonoras para indicar ações importantes, ajustes automáticos de indentação e melhorias na documentação das ferramentas para torná-las mais acessíveis. Essas contribuições reforçaram a importância de ouvir os usuários finais para promover uma inclusão mais efetiva.

Um dos participantes, P4, destacou: *"Quando a IA gera o código, não há nenhum retorno sonoro ou notificação que indique isso. Seria muito útil se tivesse um barulhinho ou mensagem indicando que o código foi gerado"*. Outro, P2, apontou a necessidade de posicionamento automático no início do código gerado: *"Quando a IA gera o código, seria ótimo que ela posicionasse o cursor no início do código gerado. Isso facilitaria a navegação e economizaria tempo"* Pois o leitor de tela começa a ler a partir do posicionamento do cursor. Além disso, foi mencionada a importância de interfaces mais intuitivas por P3: *"A interface precisa ser mais amigável, ainda está muito complicada para quem depende de leitores de tela"*. Essas sugestões, vindas da experiência prática, demonstram que, apesar dos avanços tecnológicos, ainda há lacunas significativas que precisam ser preenchidas para garantir que as ferramentas de IA sejam acessíveis e eficazes para todos.

A experiência revelou-se não apenas como um teste de ferramentas, mas como um processo de aprendizado mútuo. Os desafios enfrentados e superados proporcionaram dados que vão além da programação, destacando a complexidade de garantir acessibilidade

plena em um campo tão dinâmico quanto o da computação. A intervenção mostrou-se relevante ao combinar ferramentas de IA com práticas de inclusão, oferecendo um modelo replicável e inspirador para educadores interessados em promover um ensino de programação mais equitativo e acessível.

5. Resultados

Os resultados desta intervenção educacional evidenciam os desafios enfrentados por estudantes cegos no aprendizado de programação e destacam as contribuições das ferramentas baseadas em Inteligência Artificial (IA) para a superação de barreiras de acessibilidade e produtividade. As observações, análises de desempenho, permitiram identificar os impactos da metodologia aplicada e fornecer subsídios para a discussão de melhorias.

Os participantes enfrentaram desafios distintos ao longo das etapas do experimento. Na programação sem IA, os principais obstáculos envolveram erros de sintaxe, dificuldades com indentação, estruturas de controle e uso adequado de funções nas linguagens escolhidas. Além disso, a depuração manual exigiu um tempo significativo, especialmente para aqueles que não utilizaram pesquisas *online* como suporte. A falta de integração completa entre leitores de tela e ambientes de programação também representou uma barreira, dificultando a identificação de mensagens de erro e o acompanhamento do código. Na etapa assistida por IA, embora a produtividade tenha aumentado, novas dificuldades surgiram, como a dependência das sugestões da IA, a necessidade de interpretar e adaptar as recomendações ao contexto do problema e a acessibilidade limitada das ferramentas, com algumas sugestões de código não sendo lidas corretamente pelos leitores de tela. Esses desafios ressaltam a importância de aprimorar tanto o suporte educacional para a programação manual quanto a acessibilidade e a usabilidade das ferramentas assistidas por IA.

5.1. Experiência Sem o Uso de IA

Na etapa inicial, os participantes, Tabela 1, enfrentaram desafios significativos ao resolver problemas de programação sem o auxílio de ferramentas de Inteligência Artificial (IA). Os tempos de execução foram mais longos e os erros, recorrentes, demandaram esforço considerável para serem corrigidos, como podemos obeservar na Tabela 2.

Tabela 1. Informações dos Participantes

Participante	Status	Gênero	Linguagem
P1	EP	M	Python
P2	EG	M	Java
P3	EG	F	C++
P4	EP	M	Python

Notas: Px = Participante x, onde $x \geq 1$ e $x \leq 4$; F = Feminino; M = Masculino; EP = Estudante de Pós-Graduação; EG = Estudante de Graduação.

O participante P1 utilizando Python no bloco de notas e transferindo posteriormente o código para o VS Code, concluiu o algoritmo de ordenação em 17 minutos. No entanto, a correção dos erros detectados, como falta de indentação e um espaço indevido no nome de um método, levou 70 minutos adicionais. Esse participante não utilizou pesquisas online durante a tarefa e confiou exclusivamente em seu conhecimento prévio.

Tabela 2. Tempos e Desempenho na Programação com e sem IA

Participante	Sem uso da IA (min)			Com uso de IA	Erros Principais
	Codificação	Correção	Total	Ferramenta e Resultado	
P1	17	70	87	VS Code, código corrigido em 3 segundos	Falta de indentação, espaço no nome do método.
P2	61	67	128	ChatGPT, código gerado em segundos	Vetor fixo insuficiente, loop incompleto, lógica desalinhada.
P3	57	90	147	Repl.it, código corrigido em segundos	std::swap sem cabeçalho, função mal declarada, saída sem espaçamento.
P4	Não Finalizou	Não Finalizou	Não Aplicável	Google Colab, código gerado corretamente	Lista não declarada, ordem das operações errada.

O participante P2, programando em Java no VS Code, levou 61 minutos para implementar o código inicial. Durante a depuração, enfrentou problemas relacionados à declaração de um vetor com tamanho insuficiente para os elementos necessários, o que poderia causar uma exceção de índice fora dos limites. Além disso, a lógica de entrada de dados estava incompleta, impossibilitando o preenchimento adequado do vetor, e o algoritmo de ordenação apresentava inconsistências no loop interno. A correção desses problemas levou mais 67 minutos. Esse participante utilizou consultas online, mas ainda assim teve dificuldades em adaptar as informações encontradas à tarefa.

O participante P3, utilizando C++ no VS Code e posteriormente no Repl.it, concluiu a implementação inicial do algoritmo em 57 minutos. Contudo, erros como o uso de um método sem incluir a biblioteca e problemas na declaração de uma função resultaram em falhas contínuas durante a execução. Além disso, a saída do array não incluía espaçamento adequado entre os valores, dificultando a interpretação dos resultados. A correção dos erros demandou mais 90 minutos de trabalho.

O último participante, P4, utilizando Python no Google Colab, não conseguiu finalizar o programa, mesmo com o tempo livre e desistiu de concluir a atividade. Problemas como a ausência de declaração prévia da lista de números e a ordem inadequada das

operações de conversão e ordenação resultaram em erros de execução. Embora este participante tenha utilizado pesquisas online com o WebVox, as dificuldades em compreender e implementar o algoritmo manualmente impediram a conclusão da tarefa.

Esses resultados revelam que, sem o suporte de IA, os participantes enfrentaram desafios relacionados à lógica de programação, organização do código e uso eficaz das ferramentas. Problemas de sintaxe, erros lógicos e dificuldades na interação com os leitores de tela comprometeram significativamente o desempenho. A experiência, confome Tabela 2, destacou a complexidade do processo manual e reforçou a necessidade de suporte adicional para aumentar a eficiência e a acessibilidade no ensino de programação

5.2. Experiência Com o Uso de IA

Com a introdução de ferramentas assistidas por IA, observou-se uma melhora expressiva na eficiência e na qualidade dos códigos desenvolvidos, conforme podemos observar na Tabela 2. As sugestões de código geradas automaticamente reduziram drasticamente o tempo necessário para resolver os problemas. Para o participante P1, a IA sugeriu uma solução funcional em apenas três segundos, corrigindo erros anteriores. Para os participantes P2 e P3 a IA foi capaz de identificar, explicar e corrigir problemas no código em poucos segundos, simplificando a tarefa de depuração. Um caso específico foi o de P4 que escolheu Python, mas não conseguiu concluir a implementação do algoritmo e optou por desistir após várias tentativas frustradas. Posteriormente, ele utilizou uma ferramenta de IA para gerar o código do algoritmo proposto, o que resultou em uma solução funcional e rápida.

A utilização de assistentes como ChatGPT e funcionalidades integradas a plataformas de programação permitiu gerar algoritmos completos e precisos em alguns segundos. Essa eficiência foi relatada como um fator determinante para superar os desafios enfrentados na etapa inicial, destacando o potencial dessas ferramentas para transformar a experiência de aprendizado.

5.3. Impacto Educacional e Acessibilidade

As ferramentas de IA mostraram um impacto significativo na produtividade e na acessibilidade do processo de programação. Além de reduzirem o tempo necessário para a execução das tarefas, elas desempenharam um papel crucial na correção de códigos, mantendo a lógica inicial implementada pelos participantes. A IA foi capaz de identificar erros, explicá-los de forma clara e propor soluções adequadas, muitas vezes apenas ajustando pequenos detalhes, como problemas de sintaxe ou lógica desalinhada. Essa funcionalidade não apenas tornou o processo mais eficiente, mas também contribuiu para o aprendizado.

Apesar desses avanços, desafios específicos de acessibilidade foram relatados, especialmente no que diz respeito à interação entre leitores de tela e as sugestões da IA. Todos os participantes relataram dificuldades em localizar as sugestões geradas pela IA no editor de código, já que os leitores de tela não conseguiam identificar automaticamente onde estavam as correções ou explicações fornecidas. Um dos participantes, P3, comentou: *"A IA explica bem os erros, mas eu tive dificuldade em encontrar na tela onde estavam as sugestões para o leitor de tela ler"*.

Todos os participantes dependem de leitores de tela para interagir com ferramentas de programação. Suas experiências com essas ferramentas mostram que, embora atendam parcialmente às suas necessidades, ainda enfrentam desafios relacionados à acessibilidade, como a dificuldade em revisar saídas nos terminais ou identificar erros de compilação.

Dois participantes, P1 e P2, já haviam experimentado ferramentas de IA, como ChatGPT, GitHub Copilot e Claude.ai, enquanto os participantes P3 e P4 nunca haviam utilizado tecnologias assistidas por IA antes do experimento. Aqueles que utilizaram IA relataram um ganho significativo de produtividade e destacaram a capacidade das ferramentas de desbloquear problemas que eles não conseguiam resolver sozinhos. O participante P1 afirmou: *"A IA nos ajuda a avançar quando estamos bloqueados, seja explicando conceitos ou sugerindo soluções diretamente."*

As interfaces gráficas de algumas IDEs *(Integrated Development Environments)* foram citadas como barreiras significativas, especialmente na manipulação de objetos e elementos visuais, como *labels* e botões. A falta de suporte adequado para identificar e corrigir erros de indentação e lógica foi outro ponto crítico, tornando o processo de depuração mais lento e menos acessível.

6. Conclusões

Os resultados desta intervenção educacional revelaram que o uso de ferramentas baseadas em Inteligência Artificial (IA) pode ser um diferencial significativo no ensino de programação para estudantes cegos. As ferramentas assistidas por IA demonstraram ser eficazes na redução de barreiras de acessibilidade, aumentando a produtividade e promovendo maior autonomia entre os participantes. Problemas que anteriormente exigiam longos períodos para serem resolvidos manualmente foram solucionados de forma rápida e eficiente, contribuindo para uma experiência educacional mais inclusiva. Ao mesmo tempo, a intervenção destacou desafios importantes, como a integração limitada entre as funcionalidades das ferramentas de IA e os leitores de tela, além de dificuldades na navegação e acessibilidade.

Embora os participantes tenham afirmado que as ferramentas atualmente utilizadas atendem parcialmente às suas necessidades, eles reforçaram que a acessibilidade ainda é insuficiente. Essa situação cria uma dependência de recursos externos, como suporte técnico ou auxílio de colegas, limitando a autonomia dos programadores. Esses desafios destacam a urgência de melhorias em ferramentas de programação, com foco em maior integração com leitores de tela, estabilidade nas interfaces e suporte a processos de depuração e organização de código.

Para contextos futuros, uma extensão prática desta intervenção seria incorporar treinamentos voltados a estudantes e educadores para maximizar o uso das ferramentas de IA em ambientes educacionais. Programas de formação específicos poderiam abordar tanto as possibilidades quanto as limitações das tecnologias disponíveis, garantindo que estudantes cegos sejam capacitados para utilizá-las de forma eficaz. Além disso, colaborações com desenvolvedores de ferramentas assistivas poderiam acelerar a criação de soluções mais robustas e integradas, com base nos dados coletados durante experiências como esta.

7. Agradecimentos

Agradecemos o apoio do Programa Interno de Auxílio Financeiro aos Programas de Pós- Graduação Stricto Sensu (AUXPPG) da UEFS e do Programa de Apoio à Pós-Graduação(PROAP) da CAPES

Referências

Albusays, K., Ludi, S., and Huenerfauth, M. (2017). Interviews and observation of blind software developers at work to understand code navigation challenges. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 91–100.

Alizadehsani, Z., Gomez, E. G., Ghaemi, H., González, S. R., Jordan, J., Fernández, A., and Pérez-Lancho, B. (2022). Modern integrated development environment (ides). In Corchado, J. M. and Trabelsi, S., editors, *Sustainable Smart Cities and Territories*, pages 274–288, Cham. Springer International Publishing.

Amin, N., Saeed, A., Khalid, A., Usman, M., and Akram, F. (2024). Comparative study between jaws® and nvda® in academic performance of students with visual impairment. *British Journal of Visual Impairment*, 0(0):02646196241255889.

Chemnad, K. and Othman, A. (2024). Digital accessibility in the era of artificial intelligence—bibliometric analysis and systematic review. *Frontiers in Artificial Intelligence*, 7.

dos Santos Soares, M., Furukawa, C. A., Cagnin, M. I., and Paiva, D. M. B. (2024). Accessibility barriers for blind students in teaching-learning systems. *J. Univers. Comput. Sci.*, 30:1342–1370.

Inhelder, B. and Piaget, J. (1976). *Da lógica da criança à lógica do adolescente: ensaios sobre a construção das estruturas operatórias formais*.

Khasawneh, M. (2023). Digital inclusion: Analyzing social media accessibility features for students with visual impairments. *Studies in Media and Communication*, 12(1):71.

Mountapmbeme, A., Okafor, O., and Ludi, S. (2022). Addressing accessibility barriers in programming for people with visual impairments: A literature review. *ACM Transactions on Accessible Computing (TACCESS)*, 15(1):1–26.

Pandey, M. (2023). *Accessibility of Collaborative Programming for Blind and Visually Impaired Developers*. Master's thesis, University of Michigan.

Pandey, M., Kameswaran, V., Rao, H. V., O'Modhrain, S., and Oney, S. (2021). Understanding accessibility and collaboration in programming for people with visual impairments. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–30.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books, Inc., USA.

Petrausch, V. and Loitsch, C. (2017). Accessibility analysis of the eclipse ide for users with visual impairment. *Studies in Health Technology and Informatics*, 242:922–929.

Potluri, V., Vaithilingam, P., Iyengar, S., Vidya, Y., Swaminathan, M., and Srinivasa, G. (2018). Codetalk: Improving programming environment accessibility for visually

impaired developers. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–11.

Seo, J. and Rogge, M. (2023). Coding non-visually in visual studio code: Collaboration towards accessible development environment for blind programmers. In *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '23, New York, NY, USA. Association for Computing Machinery.

Sribunruangrit, N., Marque, C., Lenay, C., and Gapenne, O. (2004). Graphic-user-interface system for people with severely impaired vision in mathematics class. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 5145–5148.

Veiderma Holmberg, R. L. (2021). The impact of ai-based tools on software development work. H2 - master's degree (two years), Department of Computer Science, Lund University, Sweden. EDAM05 20211.

Wermelinger, M. (2023). Using github copilot to solve simple programming problems. SIGCSE 2023, page 172–178, New York, NY, USA. Association for Computing Machinery.