Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Inteligência Artificial e Acessibilidade na Programação: Um Estudo Experimental com Estudantes Cegos

Naiara Silva dos Santos

Feira de Santana

2025

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Naiara Silva dos Santos

Inteligência Artificial e Acessibilidade na Programação: Um Estudo Experimental com Estudantes Cegos

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Claudia Pinto Pereira

Feira de Santana

2025

Ficha Catalográfica – Biblioteca Central Julieta Carteado

Santos, Naiara Silva dos
S236i Inteligência Artificial e acessibilidade na programação: um estudo experimental com estudantes cegos./ Naiara Silva dos Santos, 2025.
101f.: il.

Orientadora: Claudia Pinto Pereira

Dissertação (mestrado) – Universidade Estadual de Feira de Santana. Programa de Pós-Graduação em Astronomia, 2025.

1.Inteligencia Artificial – Acessibilidade. 2.Programação. 3.Leitores de tela. 4.Ambientes de Desenvolvimento Integrado (IDEs). I.Pereira, Claudia Pinto, orient. II.Universidade Estadual de Feira de Santana. III.Titulo.

CDU : 004.14

Maria de Fátima de Jesus Moreira - Bibliotecária - CRB-5/1120

Naiara Silva dos Santos

Inteligência Artificial e Acessibilidade na Programação: Um Estudo Experimental com Estudantes Cegos

Dissertação apresentada à Universidade Estadual. de Feira de Santana como parte dos requisitos para. a obtenção do título de Mestre em Ciência da Computação.

Feira de Santana, 20 de agosto de 2025

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br
CLAUDIA PINTO PEREIRA
Data: 29/01/2026 16:49:11-0300
Verifique em https://validar.iti.gov.br

Dra. Claudia Pinto Pereira (Presidente)
Universidade Estadual de Feira de Santana (UEFS)

Documento assinado digitalmente
gov.br
NELMA DE CASSIA SILVA SANDES GALVAO
Data: 29/01/2026 12:22:05-0300
Verifique em https://validar.iti.gov.br

Dra. Nelma de Cássia Silva Sandes Galvão
Universidade Federal do Recôncavo da Bahia (UFRB)

Documento assinado digitalmente
gov.br
GABRIELA RIBEIRO PEIXOTO REZENDE
Data: 29/01/2026 16:20:39-0300
Verifique em https://validar.iti.gov.br

Dra. Gabriela Ribeiro Peixoto Rezende Pinto
Universidade Estadual de Feira de Santana (UEFS)

Abstract

Artificial Intelligence (AI) has been increasingly applied in programming environments to optimize code development and improve accessibility. However, for blind students, integrating AI into programming tools still presents challenges, especially regarding compatibility with screen readers and user interaction. This study evaluates the impact of AI-assisted coding environments on the productivity and accessibility of blind programming students. Through experiments comparing code development with and without AI support, we analyze performance, error correction efficiency, and user perception. The results indicate that AI significantly reduces debugging time and enhances accessibility; however, difficulties persist in interpreting AI-generated suggestions due to a lack of adaptation for assistive technologies. The study highlights the need for improvements in the integration of AI with screen readers and proposes adjustments to enhance accessibility in development environments. These findings contribute to the discussion on inclusive education in computing and the development of more effective AI-driven solutions for blind programmers.

Keywords: Artificial Intelligence, Accessibility, Programming, Screen Readers, Integrated Development Environments (IDEs).

Resumo

A Inteligência Artificial tem sido cada vez mais aplicada em ambientes de programação para otimizar o desenvolvimento de código e melhorar a acessibilidade. No entanto, para estudantes cegos, a integração da IA nas ferramentas de programação ainda apresenta desafios, especialmente em relação à compatibilidade com leitores de tela e à interação do usuário. Este estudo avalia o impacto de ambientes de programação assistidos por IA na produtividade e acessibilidade de estudantes cegos. Através de experimentos comparando o desenvolvimento de código com e sem suporte de IA, analisamos o desempenho, a eficiência na correção de erros e a percepção dos usuários. Os resultados indicam que a IA reduz significativamente o tempo de depuração e melhora a acessibilidade; contudo, ainda há dificuldades na interpretação das sugestões geradas pela IA, devido à falta de adaptação para tecnologias assistivas. O estudo destaca a necessidade de melhorias na integração da IA com leitores de tela e propõe ajustes para tornar os ambientes de desenvolvimento mais acessíveis. As descobertas contribuem para a discussão sobre inclusão digital no ensino da computação e o desenvolvimento de soluções mais eficazes baseadas em IA para programadores cegos.

Palavras-chave: Inteligência Artificial, Acessibilidade, Programação, Leitores de Tela, Ambientes de Desenvolvimento Integrado (IDEs).

Prefácio

Esta dissertação de mestrado foi submetida à Universidade Estadual de Feira de Santana (UEFS) como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

A dissertação foi desenvolvida no Programa de Pós-Graduação em Ciência da Computação (PGCC), tendo como orientadora a Profa. Dra. **Claudia Pinto Pereira**.

Agradecimentos

Gostaria de expressar minha mais profunda gratidão a todos que, de alguma forma, fizeram parte desta caminhada. Aos participantes da pesquisa, meu sincero reconhecimento por compartilharem seu tempo, suas experiências e, acima de tudo, por contribuírem para que este trabalho se tornasse possível e significativo. À minha orientadora, professora Claúdia Pinto Pereira, agradeço pela paciência, pela escuta atenta, pelos conselhos sábios e por acreditar no potencial desta pesquisa mesmo diante dos desafios. À minha família, que com tanto carinho e compreensão soube respeitar meus momentos de ausência e cansaço, sendo meu porto seguro em cada etapa desta jornada. Ao meu irmão Rian Silva Carvalho Santos, minha inspiração diária, que me motiva a buscar sempre o melhor de mim e a nunca desistir dos meus objetivos. Aos colegas de curso, que estiveram ao meu lado nos momentos mais difíceis e também nos mais felizes, oferecendo apoio, incentivo e partilhando conhecimentos e risadas, meu muito obrigada. Cada um de vocês fez parte não apenas de um capítulo acadêmico, mas de uma história de aprendizado, superação e afeto que levarei para toda a vida.

Sumário

Alinhamento com a Linha de Pesquisa

Linha de Pesquisa: Software e Sistemas Computacionais

A presente dissertação está alinhada com a Linha de Pesquisa: Software e Sistemas Computacionais, pois investiga o impacto da Inteligência Artificial (IA) na acessibilidade e produtividade de estudantes programadores cegos. O estudo está alinhado com pesquisas voltadas ao desenvolvimento e avaliação de software, incluindo processos de concepção, teste e adaptação de ferramentas tecnológicas para públicos específicos. A abordagem adotada permite não apenas mensurar o impacto do uso de assistentes de código, mas também contribuir para o aprimoramento das soluções existentes, promovendo uma melhor integração entre os desenvolvedores cegos e os ambientes computacionais nos quais atuam.

Produções Bibliográficas, Produções Técnicas e Premiações

Evento: Simpósio Brasileiro de Educação em Computação (EduComp 2024)

Título: (In)visibilidade da Diversidade nos Cursos Presenciais de Computação e Tecnologias da Informação e Comunicação: Um Panorama das Universidades Públicas da Bahia

DOI: DOI: 10.5753/educomp.2024.237512

Autores: Claudia Pinto Pereira, José Solenir Lima Figuerêdo, Thiago Ribeiro Alves, Naiara Silva dos Santos, Nelma de Cássia S. Sandes Galvão e Teófilo Alves Galvão Filho

---

Evento: XIII Congresso Brasileiro de Informática na Educação (CBIE 2024) I Workshop Uma Tarde na Urca: Encontro Filosófico sobre Informática na Educação (URCA 2024)

Título: Impactos da IA Generativa na Inclusão de Estudantes Programadores Cegos: Desafios e Oportunidades no Processo e Avaliação da Aprendizagem

DOI: 10.5753/urca.2024.245620

Autores: Naiara Silva dos Santos e Claudia Pinto Pereira

Evento: Simpósio Brasileiro de Educação em Computação (EduComp 2025)

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

DOI: https://doi.org/10.5753/educomp.2025.5385

Autores: Naiara Silva dos Santos, Danyele de Oliveira Santana e Claudia Pinto Pereira

Lista de Tabelas

Lista de Figuras

Lista de Abreviações

Abreviação	Descrição
CAST	Center for Applied Special Technology
CEP	Comitê de Ética e Pesquisa
DV	Deficiente Visual
IA	Inteligência Artificial
IDE	Ambiente de Desenvolvimento Integrado (Integrated Development Environment)
JAWS	Job Access With Speech
NVDA	NonVisual Desktop Access
OCR	Optical Character Recognition (Reconhecimento Óptico de Caracteres)
SBC	Sociedade Brasileira de Computação
SI	Sistema de Informação
STI	Sistemas Tutores Inteligentes
TA	Tecnologia Assistiva
TTS	Text-to-Speech (Texto para Fala)
UDI	Universal Design for Instruction)
UDL	Universal Design for Learning)
VC	Visão Computacional

VS Code Visual Studio Code

WCAG Acessibilidade para Conteúdo Web

Capítulo 1

Introdução

"A tecnologia é melhor quando une as pessoas."

– Matt Mullenweg

Vivemos em uma sociedade cada vez mais mediada por tecnologias digitais, nas quais saber programar deixou de ser apenas uma habilidade técnica para se tornar um diferencial educacional e profissional em diversas áreas (Caldeira et al., 2025). A programação está presente em soluções que envolvem saúde, educação, mobilidade, comunicação e entretenimento, configurando-se como uma ferramenta poderosa de participação social, inovação e autonomia. No entanto, para que essas oportunidades sejam acessíveis a todas as pessoas, é imprescindível que o ensino de programação seja pensado sob a ótica da inclusão. Nesse contexto, a acessibilidade, entendida como a capacidade de garantir que todas as pessoas, independentemente de suas limitações físicas, sensoriais ou cognitivas, possam interagir com produtos e ambientes em igualdade de condições (Web Accessibility Initiative (WAI), 2022) assume um papel central. No campo da computação, isso implica repensar ambientes de desenvolvimento, linguagens de programação e metodologias pedagógicas para que estudantes cegos, por exemplo, possam superar barreiras visuais que historicamente os excluíram de percursos formativos na área (Albusays et al., 2017) (Mountapmbeme et al., 2022). Como aponta Bonito (2015), garantir acessibilidade no desenvolvimento tecnológico é essencial para reduzir barreiras, independentemente do envolvimento direto dos indivíduos com a programação.

No entanto, o campo da programação de software, que frequentemente se apoia em interfaces gráficas e ambientes de desenvolvimento visuais, apresenta desafios significativos para a participação de pessoas cegas (Sribunruangrit et al., 2004). A ausência de padrões definidos e diretrizes específicas para codificação acessível dificulta a navegação e a compreensão desses ambientes, ampliando as barreiras para aqueles que dependem de tecnologias assistivas, como leitores de tela (Albusays et al.,

2017). Além disso, a crescente complexidade dos Ambientes de Desenvolvimento Integrado (IDE) e o uso intensivo de ferramentas visuais tornam esses desafios ainda mais expressivos para programadores cegos (Storer et al., 2021).

Apesar dos avanços em tecnologias assistivas, como leitores de tela e editores de código adaptáveis, programadores cegos ainda enfrentam dificuldades, especialmente na depuração de código e na compreensão de mensagens de erro, impactando sua autonomia no desenvolvimento de software (Pandey et al., 2021) (Mountapmbeme et al., 2022).

Diante dessas dificuldades, pesquisadores têm buscado soluções que reduzam as barreiras enfrentadas por programadores cegos. Uma dessas alternativas é a aplicação de Inteligência Artificial (IA) em IDEs, possibilitando a automação de tarefas e o suporte interativo no desenvolvimento de código (Veiderma Holmberg, 2021).

De acordo com Zawacki-Richter et al. (2019), a IA tem um papel essencial na personalização do aprendizado, utilizando Sistemas Tutores Inteligentes para adaptar materiais e trajetórias educacionais às necessidades individuais dos estudantes. Esses sistemas, baseados em modelos de aprendizado, algoritmos e redes neurais, são capazes de tomar decisões sobre os conteúdos mais adequados para cada estudante, fornecendo suporte cognitivo e promovendo a interação durante o processo de aprendizagem. Além disso, a IA pode contribuir para a aprendizagem colaborativa ao facilitar a formação de grupos adaptativos, auxiliar na interação online e fornecer resumos de discussões que ajudam os tutores humanos a orientar os estudantes de forma mais eficaz. Outro aspecto relevante é o uso da realidade virtual inteligente, que permite experiências imersivas de aprendizado por meio de agentes virtuais que atuam como professores, facilitadores ou colegas de estudo. Dessa forma, a IA amplia as possibilidades educacionais, personalizando não apenas o ensino individual, mas também promovendo a colaboração e o engajamento dos estudantes (Zawacki-Richter et al., 2019).

Em programação, a IA impulsiona a eficiência por meio de assistentes virtuais que sugerem código, identificam erros e facilitam a compreensão de conceitos complexos (Philbin, 2023). Ferramentas baseadas em IA (Pudari, 2022) oferecem soluções inovadoras, como sistemas de sugestão de código baseados em aprendizado de máquina, que proporcionam recursos de autocompletar e correções em tempo real. Essas ferramentas não apenas agilizam o processo de codificação, mas também tornam a navegação eficiente, permitindo que os programadores cegos tenham uma experiência mais fluida e independente. Além disso, a capacidade da IA em traduzir texto para fala e vice-versa oferece uma interação mais acessível com o código, enquanto tecnologias de reconhecimento de imagem possibilitam descrições auditivas detalhadas de elementos visuais (Khan et al., 2020; Abhishek et al., 2022). A personalização da experiência do usuário, adaptando a interface com base nas preferências individuais, demonstra o potencial da IA em criar ambientes de estudo mais inclusivos e adaptados às necessidades específicas dos estudantes com deficiência visual (Vinaykarthik et al., 2022).

Assim, a IA não só permite a superação de barreiras tradicionais, como também pode promover uma educação em programação mais acessível e inclusiva. 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. Essa lacuna impede uma compreensão mais aprofundada sobre o papel dessas tecnologias no desenvolvimento de habilidades programacionais para esse público específico.

1.1 Problema de Pesquisa

Dentro desse contexto, a acessibilidade para estudantes programadores cegos se destaca como um desafio persistente, enquanto a IA surge como uma potencial solução para superar essas barreiras. Este estudo visa explorar a interseção desses desafios, examinando como soluções de IA podem ser aplicadas para aprimorar a acessibilidade e a produtividade na programação para estudantes com deficiência visual. Ao examinar essa interação complexa, buscamos não apenas compreender os desafios enfrentados, mas também identificar oportunidades para criar ambientes de programação mais inclusivos e eficazes.

As pesquisas preexistentes de Nascimento e Brandão (2019), Pandey et al. (2021), Huff et al. (2020), Pandey et al. (2022) e Schanzer et al. (2019) abordam os desafios enfrentados por indivíduos com deficiência visual na programação, proporcionando uma compreensão das dificuldades encontradas por esses profissionais, especialmente no ambiente de trabalho. Embora esses estudos explorem os desafios enfrentados por profissionais com deficiência visual na programação, uma lacuna persiste em relação à experiência desses indivíduos durante sua formação acadêmica. A transição para o mercado de trabalho e o impacto das barreiras educacionais e de formação são áreas pouco examinadas, o que limita nossa compreensão de como preparar adequadamente esses estudantes para desafios profissionais futuros.

A literatura existente (Nascimento e Brandão, 2019; Pandey et al., 2021) frequentemente foca nas barreiras profissionais, deixando uma lacuna significativa no que tange às dificuldades educacionais e de formação. Assim, nossa investigação visa preencher essa lacuna, analisando tanto as barreiras quanto as soluções potenciais para a formação de programadores com deficiência visual.

Diante desse cenário, o uso de ferramentas de IA integradas a IDEs, como assistentes de código e sistemas de autocompletar, surge como uma possibilidade de reduzir essas barreiras, oferecendo suporte automatizado à escrita, depuração e compreensão de código. No entanto, ainda são escassos os estudos que analisam de forma empírica o impacto efetivo dessas tecnologias sobre a acessibilidade e a produtividade de estudantes cegos em ambientes reais de desenvolvimento. Dessa forma, torna-se necessário compreender em que medida a incorporação da IA nesses ambientes pode contribuir para a inclusão digital e para o aprimoramento das práticas de ensino de

programação voltadas a esse público. Com base nessas lacunas, esta pesquisa busca responder às seguintes perguntas de pesquisa:

- Q1: Como o uso de ferramentas de Inteligência Artificial (IA) em Ambientes de Desenvolvimento Integrado (IDEs) impacta a acessibilidade de estudantes programadores cegos?
- Q2: De que forma essas ferramentas influenciam a produtividade e o tempo de execução das tarefas de programação realizadas por esses estudantes?
- Q3: Quais são as percepções e desafios relatados pelos participantes em relação ao uso de IA em comparação ao desenvolvimento sem suporte automatizado?

1.2 Relevância

A inclusão de pessoas com deficiência visual no campo da programação não é apenas uma questão de equidade, mas uma necessidade social urgente diante da crescente demanda por diversidade e inovação na área de tecnologia. A exclusão de profissionais com habilidades diversas, como os programadores cegos, representa não apenas uma perda de talentos valiosos, mas também limita a pluralidade de perspectivas no desenvolvimento de soluções tecnológicas. A falta de acessibilidade em ambientes de desenvolvimento de software compromete não apenas a aprendizagem, mas também as oportunidades acadêmicas e profissionais de estudantes com deficiência visual. Garantir que esses estudantes possam programar de forma autônoma e produtiva é um passo essencial para a promoção da equidade educacional e para a ampliação da diversidade nos cursos e profissões da área de Computação.

Do ponto de vista científico e tecnológico, a pesquisa se justifica pela escassez de estudos empíricos que avaliem o impacto real da IA na acessibilidade de IDEs. Embora existam investigações sobre leitores de tela e ferramentas assistivas, ainda há pouca compreensão sobre como os recursos baseados em IA, como assistentes de código e sistemas de autocompletar, podem mitigar barreiras enfrentadas por programadores cegos e transformar sua experiência de aprendizado. Dessa forma, este estudo contribui para preencher uma lacuna na literatura, fornecendo dados e análises que podem orientar o desenvolvimento de tecnologias mais inclusivas.

Sob a perspectiva ética, a pesquisa também se insere em um debate mais amplo sobre o papel da tecnologia na construção de uma sociedade mais justa e inclusiva. A ética da inclusão exige que se promovam ambientes educacionais e profissionais equitativos, em que as diferenças sejam respeitadas e as barreiras estruturais sejam ativamente enfrentadas (Alves et al., 2025). Por outro lado, a ética no uso da Inteligência Artificial também deve ser considerada: embora a IA ofereça grandes oportunidades de apoio e personalização para pessoas com deficiência, seu uso precisa ser orientado por princípios de transparência, respeito à autonomia e não discriminação. Neste contexto, explorar o potencial da IA como recurso assistivo implica também refletir sobre seu uso responsável, inclusivo e comprometido com o

bem comum (Brotosaputro et al., 2024). Além disso, a formação de programadores cegos apresenta uma relevância estratégica. Apoiar esses estudantes desde sua trajetória educacional é essencial para capacitá-los a ocupar espaços profissionais de forma autônoma e produtiva. Trata-se não apenas de remover barreiras técnicas (Brotosaputro et al., 2024), mas de reconhecer a importância de um processo formativo acessível, sensível às suas necessidades e potencialidades.

1.3 Objetivos

Diante dos desafios enfrentados por estudantes cegos no ensino de programação, especialmente no uso de Ambientes de Desenvolvimento Integrado (IDEs) e ferramentas baseadas em Inteligência Artificial (IA), este estudo propõe um olhar sobre os aspectos de acessibilidade, usabilidade e autonomia no contexto educacional.

O presente estudo tem como objetivo geral analisar de que forma ferramentas de Inteligência Artificial (IA), quando integradas a Ambientes de Desenvolvimento Integrado (IDEs), podem contribuir para a acessibilidade e a produtividade de estudantes programadores cegos, a partir da realização de um experimento comparativo entre práticas de programação com e sem o uso de IA. Além disso, busca compreender as percepções, desafios e estratégias de uso dessas tecnologias no contexto educacional, de modo a subsidiar o desenvolvimento de práticas pedagógicas e soluções tecnológicas mais inclusivas.

No grupo dos objetivos de pesquisa, busca-se:

1. compreender as principais barreiras de acessibilidade enfrentadas por estudantes cegos em ambientes de programação;
2. investigar a integração entre leitores de tela e ferramentas de IA, avaliando seus impactos na compreensão do código;
3. analisar as percepções dos participantes sobre o uso dessas ferramentas, considerando aspectos como confiabilidade, clareza das sugestões e dependência de ajuda externa;
4. explorar as estratégias individuais adotadas por esses estudantes para superar limitações técnicas e cognitivas impostas pelas ferramentas utilizadas.

Já os objetivos de intervenção visam:

1. desenvolver e aplicar um experimento estruturado que compare o desempenho dos estudantes com e sem o suporte de IA;
2. testar o uso de diferentes ferramentas de programação e IA sob a ótica da acessibilidade; e
3. propor recomendações técnicas e pedagógicas que favoreçam a construção de ambientes de aprendizagem mais inclusivos e eficazes, com o uso de IA, especialmente no ensino de programação para pessoas com deficiência visual.

Essa distinção entre investigação e aplicação permite maior clareza metodológica e contribui para que os resultados do estudo possam ser replicados, adaptados e discutidos por educadores, desenvolvedores e pesquisadores da área da acessibilidade digital.

1.4 Hipóteses

O presente estudo parte de um conjunto de hipóteses que orientam a análise dos dados e a discussão dos resultados, a partir do qual formulam-se as seguintes hipóteses:

- A primeira hipótese (H1) considera que o uso de ferramentas de IA integradas a leitores de tela proporciona um aumento significativo na acessibilidade dos ambientes de desenvolvimento, permitindo maior fluidez na navegação e compreensão do código pelos participantes.
- A segunda hipótese (H2) propõe que o uso dessas ferramentas de IA contribui para ampliar a autonomia dos estudantes cegos durante o processo de programação.
- A terceira hipótese (H3) parte do reconhecimento de que ainda persistem barreiras significativas de acessibilidade, mesmo com o uso de IA. Essas limitações são atribuídas à baixa usabilidade de algumas interfaces, à dificuldade de leitura de sugestões de código não verbalizadas e à integração parcial entre as ferramentas de IA e os leitores de tela.
- A quarta hipótese (H4) considera que a percepção de produtividade e a eficácia no uso das ferramentas de IA variam conforme o nível de familiaridade dos estudantes com a linguagem de programação, com o ambiente de desenvolvimento e com as próprias tecnologias assistivas.
- Por fim, a quinta hipótese (H5) assume que a comparação entre as etapas com e sem o uso de IA revelará diferenças relevantes no desempenho dos estudantes, especialmente quanto ao tempo de execução das tarefas, à correção dos erros e à qualidade final dos códigos produzidos.

Essas hipóteses sustentam o processo investigativo desta dissertação, oferecendo um quadro interpretativo que orienta a análise dos dados coletados e a proposição de melhorias concretas para a acessibilidade no ensino de programação.

1.5 Contribuições

Diante do delineamento desta pesquisa e da realização de um experimento com estudantes cegos, este trabalho apresenta contribuições relevantes para o campo da acessibilidade no ensino de programação. Ao analisar o uso de ferramentas de IA em conjunto com IDEs, a pesquisa avalia como essas tecnologias impactam a

usabilidade, a autonomia e a experiência prática de estudantes cegos ao programar. A partir dessa abordagem, o estudo contribui para a compreensão mais aprofundada das limitações técnicas e pedagógicas enfrentadas por esse público.

A investigação também permite identificar barreiras específicas como falhas na leitura de sugestões de código, baixa integração com leitores de tela e dificuldades de navegação por teclado e documentar estratégias espontâneas adotadas pelos participantes para contornar esses desafios. Ao reunir esses dados, a pesquisa fornece subsídios para o aprimoramento de ferramentas educacionais e para a formação docente, com foco em práticas mais inclusivas. As percepções dos participantes oferecem uma perspectiva rica e pouco explorada na literatura, reforçando a importância de abordagens centradas no usuário com deficiência.

Por fim, destaca-se como contribuição metodológica a estruturação de um procedimento experimental acessível, baseado em tarefas práticas, entrevistas e observações remotas, que pode ser replicado por outros pesquisadores. Ao propor recomendações técnicas e pedagógicas a partir de dados empíricos, este trabalho avança no debate sobre inclusão digital e equidade no acesso à tecnologia, demonstrando que a IA, quando acessível e bem implementada, pode ampliar significativamente a autonomia e o protagonismo de estudantes cegos no processo de aprendizagem em Computação.

1.6 Organização do Trabalho

Este trabalho está estruturado em sete capítulos. O Capítulo 1 apresenta a introdução, destacando o problema de pesquisa, a relevância da pesquisa, os objetivos, as hipóteses e as contribuições esperadas. No Capítulo 2, é realizada uma revisão bibliográfica, abordando tecnologias assistivas para programadores cegos e soluções baseadas em Inteligência Artificial (IA) em Ambientes de Desenvolvimento Integrado (IDEs). O Capítulo 3 detalha a metodologia utilizada, incluindo o levantamento bibliográfico, os critérios de seleção de participantes e a condução dos experimentos. Além disso, são apresentados os fundamentos teóricos do estudo de caso e da análise de conteúdo. Em seguida, no Capítulo 4, são apresentados os resultados da pesquisa, com análises quantitativas e qualitativas sobre o impacto da IA na acessibilidade e produtividade de estudantes programadores cegos. O Capítulo 5 discute os achados do estudo, relacionando-os com trabalhos correlatos e destacando desafios e oportunidades para a melhoria das ferramentas de IA. O Capítulo 6 apresenta diretrizes técnicas e pedagógicas para promover a acessibilidade e a autonomia de estudantes cegos no ensino de programação, com foco no uso de leitores de tela, sistemas operacionais, IDEs e ferramentas de IA. Por fim, o Capítulo 7 conclui a pesquisa, sintetizando os principais resultados e sugerindo direções para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

"Inclusão é simplesmente fazer tudo pensando nas pessoas que existem. E não considerando pessoas que você gostaria que existissem. "

– Claudia Werneck

O presente capítulo tem como objetivo fundamentar teoricamente os principais conceitos que sustentam esta pesquisa. Para isso, são discutidos temas como acessibilidade digital, recursos de tecnologia assistiva voltados para pessoas com deficiência visual, o uso de inteligência artificial no contexto educacional, especialmente no ensino de programação, e a acessibilidade em ambientes de desenvolvimento integrado (IDEs). Além disso, são apresentadas experiências educacionais de estudantes cegos e estratégias pedagógicas, com ênfase na construção de sequências didáticas inclusivas, que contribuem para a formação de um referencial sólido e alinhado aos objetivos da pesquisa.

2.1 Acessibilidade Digital e Inclusão Educacional

A acessibilidade digital é definida como a prática de projetar sistemas e serviços digitais de modo que possam ser utilizados por todas as pessoas, inclusive aquelas com deficiências visuais, auditivas, motoras ou cognitivas (Chemnad e Othman, 2024). De acordo com a *Web Accessibility Initiative (WAI) (2022)*, isso implica garantir que esses usuários possam acessar, navegar, perceber e interagir com o conteúdo digital de forma plena e autônoma. Os princípios da acessibilidade digital envolvem a criação de ambientes inclusivos, sem barreiras, que promovam equidade no acesso à informação e à participação.

No contexto educacional e tecnológico, essa preocupação torna-se ainda mais urgente. À medida que a educação e os serviços se digitalizam, torna-se essencial

assegurar que estudantes com deficiência não sejam excluídos das oportunidades de aprendizagem e desenvolvimento. A acessibilidade digital, portanto, além de representar uma responsabilidade social e uma exigência legal, constitui um pilar fundamental para a inclusão, o empoderamento e a autonomia de todos os indivíduos na era da inteligência artificial (Chemnad e Othman, 2024).

Particularmente no ensino superior, a acessibilidade deve ser compreendida como um direito fundamental e um requisito indispensável para a construção de ambientes educacionais verdadeiramente inclusivos (Wilkens et al., 2021). O *Universal Design for Learning* - UDL é um modelo pedagógico criado pela organização *Center for Applied Special Technology* - CAST[1] na década de 1980, segundo Espada-Chavarria et al. (2023) o UDL é uma abordagem focada no ensino, aprendizagem, desenvolvimento curricular e avaliação. Baseia-se em pesquisas sobre os processos cerebrais e no uso das Tecnologias de Informação e Comunicação - TIC, com o objetivo de responder às diferenças individuais na aprendizagem. Assim, o UDL é aplicado como uma estratégia de ensino que visa eliminar barreiras por meio de um modelo flexível e adaptável, que inclui todos os estudantes e estimula o desenvolvimento de suas habilidades (Bray et al., 2024). Essa abordagem reconhece a diversidade entre os estudantes e propõe, desde o planejamento, estratégias que favoreçam a participação de todos, por meio da flexibilidade e da oferta de escolhas na forma de engajamento e aprendizagem. Fundamentado na neurociência, o UDL valoriza a variabilidade dos aprendizes e busca promover a inclusão por meio da diversificação de recursos e métodos pedagógicos. Seu arcabouço é sustentado por três princípios centrais: múltiplos meios de engajamento, voltados à motivação e ao envolvimento ativo; múltiplos meios de representação, que garantem diferentes formas de acesso às informações; e múltiplos meios de ação e expressão, que possibilitam aos estudantes demonstrar seus conhecimentos de maneiras diversas (Bray et al., 2024).

Já o *Universal Design for Instruction* - UDI é um conceito relativamente recente na educação voltada para universidades. É definido como o modelo que desenvolve métodos instrucionais para que todos os estudantes com diversas necessidades de aprendizagem tenham acesso equitativo ao ensino (Espada-Chavarria et al., 2023). O UDI aplicado ao ensino universitário não se refere apenas à acessibilidade para pessoas com deficiência. É uma abordagem verdadeiramente universal porque considera as necessidades futuras de todos os estudantes ao projetar o conteúdo e o ensino. Este processo é usado para identificar e eliminar barreiras no ensino, mantendo o rigor acadêmico e impulsionando a aprendizagem dos estudantes, independentemente de seus conhecimentos e preferências, reduzindo ao mínimo a necessidade de adaptações especiais. É baseado em princípios como: os processos de ensino devem promover a interação e a comunicação entre estudantes e entre estudantes e professores, e o ensino deve ser projetado para ser acolhedor e inclusivo.

Instrumentos como o UDL e o UDI reforçam a importância de incorporar a acessibilidade desde a concepção das práticas pedagógicas, e não como um recurso comple-

[1] https://udlguidelines.cast.org

mentar. Essa abordagem assegura que as experiências de ensino sejam planejadas para contemplar a diversidade dos estudantes desde o início do curso, estendendo-se também aos espaços digitais. Nesse contexto, refletir sobre a diversidade implica, necessariamente, considerar a acessibilidade digital como parte integrante de um processo educacional inclusivo.

Garantir que ambientes virtuais de aprendizagem sejam acessíveis para pessoas com deficiência visual exige o uso de estratégias específicas que promovam a interação autônoma e significativa com plataformas e conteúdos digitais. Isso inclui o uso de recursos de tecnologia assistiva, como leitores de tela, linhas braille e softwares de ampliação, que dependem diretamente de um design digital fundamentado nas diretrizes internacionais de acessibilidade, como as Diretrizes de Acessibilidade para Conteúdo Web-WCAG [2]. Como aponta Cavalcante (2022), a ausência de elementos como estrutura semântica adequada, textos alternativos e navegação acessível compromete a usabilidade desses recursos, limitando a participação efetiva e o direito à educação. A autora enfatiza que a acessibilidade digital deve ser entendida como um direito humano, sendo essencial que professores, desenvolvedores e instituições sejam capacitados para planejar e implementar práticas acessíveis, possibilitando às pessoas com deficiência visual vivenciarem o espaço digital em igualdade de condições, com autonomia e protagonismo.

2.2 Recursos de Tecnologia Assistiva

Tecnologia Assistiva (TA) é um campo interdisciplinar que reúne produtos, serviços, práticas e estratégias com o objetivo de ampliar ou restaurar as habilidades funcionais de pessoas com deficiência, promovendo sua autonomia, independência e inclusão social. Esses recursos podem variar desde dispositivos simples até sistemas complexos, sempre com foco na melhoria da qualidade de vida e na participação ativa dos indivíduos em diferentes contextos sociais. Trata-se, portanto, de uma área que busca eliminar barreiras e favorecer a equidade no acesso a oportunidades e direitos (Bastos et al., 2023).

Existem alguns sinônimos para tratar da temática acerca de TA, como “Ajudas Técnicas”, “Tecnologia de Apoio”, “Tecnologia Adaptativa” e “Adaptações”, mas a expressão Tecnologia Assistiva é uma tradução brasileira do termo *Assistive Technology* a respeito do qual diversos países apresentam percepções e classificações diferentes, conforme citado por Galvão Filho (2009) apud Bastos et al. (2023).

A tecnologia assistiva assume muitas formas, desde cadeiras de rodas a aparelhos auditivos e próteses. No contexto da acessibilidade digital, refere-se a ferramentas de software e hardware que permitem que pessoas com deficiência acessem e interajam com plataformas web, aplicativos e dispositivos digitais, garantindo uma navegação mais autônoma e equitativa. Nesse escopo mais amplo de acessibilidade digital,

[2] https://www.w3c.br

destacam-se também os recursos de TA voltados especificamente para a área da programação. Tais ferramentas são projetadas para permitir que pessoas com deficiência não apenas acessem conteúdos, mas também possam interagir com ambientes de desenvolvimento e participar ativamente da construção de soluções computacionais (Eckhardt et al., 2019). Essas tecnologias ajudam a preencher a lacuna entre as habilidades dos indivíduos e as exigências das tarefas computacionais, promovendo a inclusão e a igualdade de oportunidades na indústria tecnológica.

Entre os diversos recursos, destaca-se o leitor de tela como uma das ferramentas mais essenciais para pessoas com deficiência visual, sendo sua compreensão indispensável para o desenvolvimento deste estudo.

2.2.1 Leitores de Tela

Os recursos de tecnologia assistiva para pessoas com deficiência visual na programação desempenham um papel crucial na promoção da inclusão digital e na facilitação do acesso ao desenvolvimento de *software*. Entre as ferramentas mais importantes estão os leitores de tela que, conforme Wataya (2006), descrevem o conteúdo exibido no monitor de computador ou ainda de um código-fonte de página da Internet. Esta tecnologia é chamada de *"text-to-speech"* ou TTS. Dentre os leitores de tela, os mais conhecidos são o *Job Access With Speech*[3] - JAWS (Ndlovu et al., 2023) e *NonVisual Desktop Access*[4] - NVDA (Amin et al., 2024), que permitem a interação com interfaces de desenvolvimento ao converter texto visual em descrições auditivas detalhadas. Esses leitores de tela tornam possível a navegação, a edição e a depuração de código, oferecendo uma interface auditiva que guia os programadores através dos elementos visuais (Amin et al., 2024).

Os leitores de tela são ferramentas essenciais para programadores cegos, desempenhando um papel crucial na sua capacidade de acessar, interagir e produzir código de maneira eficiente, permitindo que indivíduos com deficiência visual utilizem computadores e dispositivos móveis de forma autônoma (Amin et al., 2024), ou seja, sem o leitor de tela, a pessoa com deficiência visual não teria acesso às informações visuais necessárias para programar autonomamente. Cabe destacar que esses leitores de tela não foram originalmente concebidos para a leitura de código de programação, mas, com a devida configuração, é possível utilizá-los de forma eficiente. No Capítulo 6, traremos configurações e comandos essenciais para programadores com deficiência visual.

Os leitores de tela, amplamente utilizados em nível internacional, são o NVDA e o JAWS, mas é importante destacar o Dosvox, que é um sistema para microcomputadores da linha PC que se comunica com o usuário através de síntese de voz, desenvolvido no Brasil pela Universidade Federal do Rio de Janeiro (UFRJ) (dos Santos Borges, 2000). O Dosvox é um sistema que oferece um ambiente próprio

[3]https://support.freedomscientific.com/Downloads/JAWS
[4]https://www.nvaccess.org/

e simplificado para pessoas cegas interagirem com o computador por meio de voz, diferindo dos leitores de tela, como o JAWS, que apenas interpretam e narram as interfaces gráficas de sistemas operacionais e programas já existentes. O Dosvox possui grande relevância no contexto educacional brasileiro, pois foi um dos primeiros sistemas voltados à acessibilidade de pessoas cegas e ainda é amplamente utilizado em escolas e universidades. Sua popularidade se deve ao conjunto de ferramentas integradas: como editor de textos, navegador e ambiente de programação, projetadas para uso autônomo por pessoas com deficiência visual. Assim, a familiaridade de muitos estudantes com o Dosvox reforça sua importância como ferramenta de inclusão digital e como ponto de partida para o aprendizado de recursos de tecnologia assistiva mais complexas, como IDEs integradas a leitores de tela modernos.

Além dos leitores de tela e sintetizadores de voz, existem outros recursos de TA desenvolvidos para atender às múltiplas necessidades de pessoas com deficiência, sendo fundamentais para garantir o acesso pleno a conteúdos e funcionalidades no ambiente digital. Dentre eles: (1) o *Project CodeTalk*[5], implementado como um *plugin* para o *VisualStudio*, mas com técnicas amplamente aplicáveis a qualquer IDE visual, que aborda sistematicamente as barreiras enfrentadas por usuários com deficiência visual - DV; (2) o *Emacspeak*[6], que é uma interface de fala que permite que usuários com DV interajam de forma independente e eficiente com o computador, e o (3) *Copilot Voice*[7], que é um recurso útil do *Microsoft Copilot* que permite interagir com o assistente de IA usando comandos de voz em linguagem natural. A seguir, são exploradas algumas das soluções baseadas em IA e seus impactos na experiência de programadores cegos.

2.2.2 Soluções de IA em Ambientes de Desenvolvimento Integrado (IDEs)

As soluções de Inteligência Artificial (IA) em Ambientes de Desenvolvimento Integrado (IDEs) têm revolucionado a forma como os programadores interagem com o código, oferecendo uma gama de ferramentas que aumentam a produtividade e a acessibilidade (Alizadehsani et al., 2022). Ferramentas como o *GitHub Copilot* (Wermelinger, 2023) e *TabNine Ai Code Assistant* (Corso et al., 2024) utilizam algoritmos avançados de aprendizado de máquina para fornecer sugestões de código em tempo real, autocompletar trechos de programação e corrigir erros, o que pode ser especialmente útil para programadores cegos. Além disso, soluções de *debugging* baseadas em IA, como o *DeepCode AI* (Nadukuda, 2023) e o *Amazon CodeGuru* (Sikha e Others, 2024), identificam vulnerabilidades e problemas de desempenho no código, proporcionando *feedback* imediato que pode ser acessado auditivamente. Essas inovações não apenas agilizam o processo de desenvolvimento, mas também tornam o ambiente de codificação mais inclusivo, permitindo que programadores

[5]https://www.microsoft.com/en-us/research/project/codetalk/
[6]https://emacspeak.sourceforge.net
[7]https://www.microsoft.com/pt-br/microsoft-copilot/

com deficiência visual naveguem, escrevam e depurem código com maior eficiência e independência. A seguir, apresentaremos algumas soluções de IA integradas a IDEs.

O *GitHub Copilot* pode ser acessado por meio de um *plugin* integrado a editores como o *Visual Studio Code*, onde fornece sugestões de código de forma automática enquanto o usuário digita ou mediante comandos específicos, como pressionar `Alt-` ou `Enter` (Wermelinger, 2023). As sugestões aparecem em fonte cinza e itálica no local do cursor e são chamadas de *ghost text*, podendo ser aceitas com `Tab` ou exploradas em versões alternativas por meio do comando `Ctrl-Enter`, que exibe até dez variações em um painel separado. Essa funcionalidade visa acelerar a escrita de código ao completar trechos automaticamente com base no contexto do que está sendo digitado. Assim, neste trabalho, sempre que nos referirmos à IA integrada ao *Visual Studio Code* estamos nos referindo especificamente ao *GitHub Copilot*.

Já a IA integrada ao *Replit*[8], chamada *Replit Agent*, tem se mostrado uma ferramenta muito promissora para tornar a programação acessível a pessoas com deficiência visual. Ela atua como um assistente de programação inteligente, ajudando a simplificar tarefas complexas e facilitando o desenvolvimento de código por meio de comandos em linguagem natural, o que é especialmente valioso para programadores cegos ou com baixa visão.

No *Google Colab* [9] temos o novo assistente de codificação com foco em IA, integrado diretamente no ambiente do *Colab*. Esse assistente funciona como um colaborador agente projetado para entender profundamente o código, intenções e fluxos de trabalho em tempo real. Ele vai além da simples conclusão de código, ajudando ativamente em todo o seu notebook, incluindo múltiplas células, com tarefas como geração de código, depuração, limpeza de dados e engenharia de recursos (Llerena-Izquierdo et al., 2024).

Por fim, temos o *ChatGPT* que é definido por Mohamed et al. (2024) como um modelo de inteligência artificial generativa baseado em grandes modelos de linguagem (*Large Language Models* - LLMs), cuja aplicação tem se expandido significativamente em áreas como medicina, educação, comunicação e, especialmente, engenharia de software. Segundo os autores, o *ChatGPT* tem sido amplamente utilizado por desenvolvedores tanto para tarefas relacionadas à codificação (como escrita de trechos de código, depuração e análise) quanto para atividades não técnicas, como auxílio na escrita, preparação para entrevistas técnicas e realização de testes de Turing. Essa versatilidade evidencia o papel emergente do *ChatGPT* como uma ferramenta crítica no apoio à produtividade e à criatividade de desenvolvedores, destacando seu potencial na construção de um cenário cada vez mais orientado à colaboração entre humanos e sistemas de IA.

Embora ferramentas como o *Google Colab*, o *Replit* e o *ChatGPT* sejam amplamente utilizadas por programadores, inclusive aqueles com deficiência visual, é importante

[8] https://replit.com
[9] http://colab.research.google.com

destacar que essas plataformas não foram originalmente concebidas com o objetivo de atuar como assistentes de acessibilidade. Seu desenvolvimento esteve voltado para a facilitação do aprendizado, experimentação e produtividade em programação de forma geral, sem foco específico em recursos inclusivos.

2.2.3 Implementações de IA para Acessibilidade Digital

Com os avanços recentes, a IA passou a ser incorporada a muitas tecnologias, aprimorando significativamente sua funcionalidade e tornando as tarefas cotidianas mais acessíveis (Philbin, 2023). A IA não só potencializa as ferramentas existentes, mas tem se consolidado como um novo recurso de TA em si, oferecendo soluções inovadoras para atender às mais diversas demandas de acessibilidade (Pandey, 2023). Essa capacidade de adaptação proporciona experiências mais inclusivas, centradas nas necessidades específicas de cada usuário. A tecnologia assistiva é descrita por Galvão Filho (2022) como um conjunto de recursos e serviços que têm o objetivo de proporcionar ou ampliar habilidades funcionais de pessoas com deficiência, promovendo assim a inclusão e a autonomia. Este conceito abrange desde dispositivos simples, como lupas, até tecnologias mais complexas, como *softwares* de comunicação aumentativa. Apesar do aumento das opções de recursos de tecnologia assistiva e da acessibilidade ampliada às tecnologias convencionais, as pessoas com deficiências ainda encontram dificuldades para aprender a programar, dado que a escrita de código é uma tarefa complexa mesmo após a compreensão dos recursos e da semântica de uma linguagem de programação (Baker et al., 2019).

Para superar esses desafios, o *IntelliCode* foi desenvolvido como uma solução que elimina barreiras, permitindo que indivíduos com deficiências alcancem seus objetivos de programação (Kumar et al., 2023). Utilizando comandos de voz, este ambiente de programação torna a escrita de código mais acessível, promovendo a independência e a inclusão de programadores com diferentes necessidades. O *IntelliCode* (Kumar et al., 2023) é um ambiente de programação baseado em voz, desenvolvido para auxiliar programadores com deficiência visual ou cegueira, bem como aqueles com lesões motoras, como lesões por esforço repetitivo, que dificultam a programação. Kumar et al. (2023) indicam que cerca de 2 em cada 100 desenvolvedores de *software* possuem alguma deficiência visual, e muitos continuam a enfrentar desafios significativos na prática da programação.

Apesar de o IntelliCode apresentar-se como uma solução projetada especificamente para programadores cegos, ele não se configura como a melhor ou mais adotada alternativa por esse público. Isso ocorre por diversos motivos identificados tanto na literatura quanto na prática dos participantes desta pesquisa. Primeiro, trata-se de um ambiente restrito: embora baseado em voz, ele não possui integração plena com leitores de tela amplamente utilizados, como NVDA e JAWS, o que limita seu uso em fluxos de trabalho reais. Além disso, seu conjunto de funcionalidades ainda é reduzido quando comparado a IDEs modernas, carecendo de recursos avançados de depuração, navegação por código, personalização de atalhos e extensões. Soma-se a

isso o fato de que o IntelliCode não se integra a ferramentas contemporâneas de IA generativa, como ChatGPT, Copilot ou os assistentes inteligentes do Colab e Replit, que hoje compõem parte importante da prática de programação, inclusive para pessoas cegas. Os próprios autores (Kumar et al., 2023) reconhecem limitações na robustez do reconhecimento de voz e na ausência de suporte para tarefas complexas. Assim, embora importante como iniciativa, o IntelliCode não atende plenamente às necessidades reais de acessibilidade e produtividade dos desenvolvedores cegos, reforçando a necessidade de investigar como as ferramentas de IA generativa podem preencher lacunas deixadas por soluções anteriores.

Outro estudo de implementação com IA é o de Brotosaputro et al. (2024) que investiga o impacto dos recursos de TA baseados em IA na acessibilidade para pessoas com deficiência. A pesquisa combina abordagens qualitativas e quantitativas para avaliar a usabilidade, eficiência e satisfação do usuário ao utilizar soluções integradas de IA em comparação com tecnologias assistivas tradicionais. Os resultados indicam que ferramentas baseadas em IA proporcionam uma melhoria significativa na execução de tarefas, reduzindo o tempo de conclusão e aumentando a autonomia dos usuários. Entre as aplicações analisadas, destacam-se sistemas de reconhecimento de fala e de emoções, que demonstraram alto potencial para aprimorar a comunicação e a interação de pessoas com deficiência. No entanto, o estudo também aponta desafios, como a necessidade de adaptações para diferentes perfis de usuários e a importância da acessibilidade digital equitativa.

Complementando essa perspectiva sobre o potencial da IA na acessibilidade, outro estudo relevante é o de M S et al. (2024) apresenta o *WebSight*, uma extensão de navegador que utiliza um gerador de descrições de imagens baseado em IA integrando-se de forma eficiente com um servidor *Flask*, reconhecimento óptico de caracteres (OCR) e um modelo dedicado de gerador de descrições de imagens. Esta extensão melhora sistematicamente a acessibilidade *web* ao substituir os atributos 'alt' das imagens por descrições meticulosamente geradas. Essa abordagem contribui significativamente para a inclusão digital, fornecendo aos usuários com deficiência visual informações precisas e contextualmente ricas, já que as pessoas com deficiência visual ainda têm dificuldade para usar a maioria dos sites.

2.3 IA no Ensino Superior

Zawacki-Richter et al. (2019) realizaram uma revisão sistemática sobre o uso da Inteligência Artificial (IA) na educação superior, identificando suas principais aplicações, como sistemas tutores inteligentes, aprendizado personalizado e automação de tarefas administrativas. O foco da pesquisa foi mapear tendências e identificar lacunas no envolvimento dos educadores no desenvolvimento e uso dessas tecnologias. Os autores destacam que, embora a IA tenha avançado significativamente na área educacional, há uma lacuna na participação ativa dos educadores no desenvolvimento dessas tecnologias, o que pode comprometer sua efetividade pedagógica. Além disso,

o estudo aponta desafios como a falta de padronização das ferramentas de IA e a necessidade de pesquisas mais voltadas para acessibilidade e inclusão digital.

De maneira semelhante, Michel-Villarreal e Vilalta-Perdomo (2023) exploram os impactos da inteligência artificial generativa no ensino superior, destacando seus desafios e oportunidades. Discutem como ferramentas como o *ChatGPT* podem personalizar o ensino, apoiar professores, desenvolver habilidades críticas nos estudantes e prepará-los para o mercado de trabalho. Além disso, abordam preocupações éticas, como privacidade de dados e possíveis vieses nos algoritmos, bem como desafios técnicos que podem afetar a adoção da tecnologia em diferentes instituições.

Corroborando com essas discussões, Aler Tubella et al. (2024) exploram estratégias para incorporar a IA responsável no ensino superior. Através de uma revisão da literatura existente e entrevistas com 11 especialistas de cinco países, os autores identificam competências, recursos e desafios na implementação de uma IA confiável na educação superior. As conclusões são apresentadas na forma de recomendações tanto para educadores quanto para formuladores de políticas, visando traduzir diretrizes em práticas de ensino, capacitando a próxima geração a contribuir para uma IA ética e segura. Na recomendação para educadores, destaca-se a importância de incluir explicitamente, nos cursos, os requisitos da *High-Level Expert Group* (HLEG)[10], que são eles: agência e supervisão humana (garantir autonomia e controle humano sobre a IA); robustez técnica e segurança (assegurar que o sistema seja preciso, seguro e resiliente); privacidade e governança de dados (proteger dados pessoais e garantir seu uso responsável); transparência (tornar decisões e processos compreensíveis e rastreáveis); diversidade, não discriminação e equidade (evitar vieses e promover inclusão); bem-estar social e ambiental (contribuir para impactos positivos na sociedade e no meio ambiente); responsabilidade (estabelecer mecanismos claros de prestação de contas e auditoria), quando pertinentes, deixando claro como eles se conectam ao conteúdo ministrado. Recomenda-se também integrar metodologias de desenvolvimento de IA Confiável, como procedimentos de registro, técnicas de coleta de dados com preservação de privacidade e ferramentas de explicabilidade, além de definir resultados de aprendizagem claros, que contemplem três níveis: apreciação (identificar aplicabilidade e dimensões dos requisitos), análise (deliberar sobre implementações e implicações éticas) e aplicação (selecionar e implementar soluções técnicas).

Para os formuladores de políticas Aler Tubella et al. (2024), sugere-se coordenar a introdução da IA Confiável nos currículos por meio de estratégias nacionais, garantindo uniformidade, investir na formação e contratação de especialistas para fortalecer a expertise docente e incentivar a colaboração interdisciplinar, valorizando-a no currículo e atribuindo créditos a atividades que integrem diferentes áreas do conhecimento.

[10]https://digital-strategy.ec.europa.eu/en/policies/expert-group-ai

2.4 Trabalhos correlatos

Alguns estudos têm identificado problemas e limitações de acessibilidade enfrentados por estudantes com deficiência visual em cursos de Computação. Baker, Bennett e Ladner (Baker et al., 2019) realizaram uma pesquisa qualitativa e descobriram que os estudantes enfrentam desafios diários que vão desde o acesso aos materiais e a realização das atividades até o relacionamento com os professores. Estes desafios são amplamente discutidos em pesquisas recentes, como a de Zen et al. (2023), que identificou dificuldades específicas com materiais didáticos e a interação com IDEs, e a de Mountapmbeme et al. (2022), que examinou barreiras ao longo de todo o percurso educacional, desde o acesso a materiais até interações com professores. Adicionalmente, Chemnad e Othman (2024) realizaram uma revisão sistemática que foca nas aplicações de IA para acessibilidade digital, destacando a importância da IA na melhoria da acessibilidade para pessoas com deficiência. As subseções seguintes discutirão esses trabalhos em detalhes, proporcionando uma visão abrangente dos desafios e soluções propostas na literatura existente sobre acessibilidade na educação em programação para estudantes com deficiência visual.

2.4.1 Experiências Educacionais em Disciplinas de Programação de Computadores: uma Análise Qualitativa na Perspectiva dos Estudantes com Deficiência Visual

O trabalho de Zen et al. (2023) apresenta uma pesquisa qualitativa que identifica desafios e limitações enfrentados por estudantes com deficiência visual em disciplinas de Programação de Computadores, a partir de entrevistas com seis estudantes e egressos de cursos superiores de Computação. Os resultados mostram que muitos materiais utilizados nos cursos, como livros didáticos e apostilas, não são totalmente acessíveis aos recursos de tecnologia assistiva utilizados pelos entrevistados. A falta de um padrão estabelecido para a verbalização do código-fonte por leitores de tela resulta em omissão ou transmissão inadequada de informações importantes, dificultando a compreensão do código.

Os estudantes também relataram dificuldades de interação com os IDEs adotados nas disciplinas, sendo frequentemente necessário recorrer a ferramentas alternativas com funcionalidades reduzidas, o que demandava mais tempo e esforço cognitivo, causando frustração e impactando negativamente o engajamento e a dedicação. Além disso, a falta de treinamento e familiaridade com essas tecnologias ao longo do curso pode prejudicar a formação profissional dos estudantes.

Os resultados destacam a importância de conhecer as necessidades e preferências dos estudantes com deficiência visual para melhorar a acessibilidade das metodologias e ferramentas de ensino de programação. Professores de programação, professores de sala de recursos e educadores especiais precisam estar familiarizados com os recursos de tecnologia assistiva e suas configurações para auxiliar os estudantes. Promover

a conscientização e fornecer treinamento adequado aos educadores é essencial para criar um ambiente de aprendizagem inclusivo e eficaz.

2.4.2 Abordando as Barreiras de Acessibilidade na Programação para Pessoas com Deficiência Visual: Uma Revisão da Literatura

Addressing Accessibility Barriers in Programming for People with Visual Impairments: A Literature Review

Mountapmbeme et al. (2022) examinam as experiências educacionais de programadores cegos em cursos de ciência da computação no ensino superior. Através de uma pesquisa e entrevistas com 10 programadores cegos, os autores identificam e discutem os desafios enfrentados por esses estudantes, incluindo barreiras no acesso a materiais didáticos, interações com professores e a utilização de recursos de tecnologia assistiva.

Os resultados indicam que os estudantes cegos enfrentam dificuldades significativas em todas as etapas de seu percurso educacional, desde a acessibilidade dos materiais didáticos e realização de tarefas até o trabalho com professores. Essas barreiras aumentam a sensação de isolamento, diminuem a motivação, especialmente quando as tarefas são inacessíveis, e impedem os estudantes de aprender todos os conceitos abordados em seus programas de graduação. O estudo também aborda as implicações desses desafios para a criação de culturas mais acolhedoras para programadores com deficiência dentro da comunidade educacional de computação e sugere áreas para colaboração futura entre educadores e pesquisadores para remover essas barreiras.

2.4.3 Acessibilidade digital na era da inteligência artificial — Análise bibliométrica e revisão sistemática

Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review

A acessibilidade digital, conforme Chemnad e Othman (2024), envolve o design de sistemas e serviços digitais para permitir o acesso a indivíduos com deficiências, incluindo deficiências visuais, auditivas, motoras ou cognitivas. A IA tem o potencial de aprimorar a acessibilidade para pessoas com deficiências e melhorar sua qualidade de vida geral.

Esta revisão sistemática (Chemnad e Othman, 2024), abrangendo artigos acadêmicos de 2018 a 2023, foca nas aplicações de IA para acessibilidade digital. A pesquisa enfatiza o foco predominante na acessibilidade digital impulsionada por IA para deficiências visuais, revelando uma lacuna crítica no atendimento a deficiências de fala e audição, transtorno do espectro autista, distúrbios neurológicos e deficiências motoras. Isso destaca a necessidade de uma distribuição mais equilibrada das pesquisas

para garantir suporte equitativo para todas as comunidades com deficiências. O estudo também apontou a falta de adesão aos padrões de acessibilidade nos sistemas existentes, ressaltando a urgência de uma mudança fundamental no design de soluções para pessoas com alguma deficiência. No geral, esta pesquisa sublinha o papel vital da IA acessível na prevenção da exclusão e discriminação, instando uma abordagem abrangente para a acessibilidade digital que atenda às diversas necessidades de deficiência.

2.4.4 Avaliação de acessibilidade dos principais aplicativos móveis assistivos disponíveis para pessoas com deficiência visual

Accessibility evaluation of major assistive mobile applications available for the visually impaired

Bhagat et al. (2024) realizaram uma avaliação da acessibilidade de aplicativos móveis assistivos voltados para pessoas com deficiência visual, analisando quatro das principais ferramentas baseadas em IA e Visão Computacional (VC). O estudo considerou critérios como precisão, tempo de resposta, confiabilidade, acessibilidade, privacidade, eficiência energética e usabilidade, com dados diretos de usuários cegos e com baixa visão. Os resultados indicaram que, embora esses aplicativos forneçam suporte significativo para navegação e reconhecimento de objetos, ainda apresentam desafios na compatibilidade com leitores de tela e na interpretação de ambientes complexos. Além disso, questões como latência na resposta e falta de personalização foram apontadas como barreiras para uma experiência mais intuitiva e eficiente. Esse estudo reforça a necessidade de melhorias na integração entre IA e tecnologias assistivas, alinhando-se ao presente trabalho ao destacar a importância da acessibilidade digital no desenvolvimento de soluções inclusivas para programadores e usuários cegos.

2.4.5 Diretrizes de Acessibilidade em Ambientes de Desenvolvimento Integrado para Estudantes Cegos

Zen (2024) desenvolveu uma pesquisa focada na acessibilidade de Ambientes de Desenvolvimento Integrado (IDEs) para estudantes cegos, identificando barreiras enfrentadas no uso dessas ferramentas e propondo diretrizes para torná-las mais acessíveis. Analisou dificuldades como a compatibilidade limitada com leitores de tela, a ausência de *feedback* auditivo adequado e a complexidade na navegação por código, que impactam diretamente a autonomia e a produtividade dos programadores cegos. Com base nos achados, a pesquisa apresentou recomendações para o *design* de IDEs mais inclusivos, sugerindo melhorias na interface, integração com recursos de tecnologia assistiva e adaptação de funcionalidades de depuração e autocompletar. Este estudo dialoga diretamente com a presente pesquisa, pois reforça a necessidade de aprimoramento na acessibilidade das ferramentas de programação e a importância da IA como um meio para otimizar a experiência desses usuários, garantindo maior inclusão no ensino e na prática da computação.

2.4.6 Usuários de leitores de tela na era do Vibe Coding: adaptação, empoderamento e novo cenário de acessibilidade

Screen Reader Users in the Vibe Coding Era: Adaptation, Empowerment, and New Accessibility Landscape

O trabalho de Chen et al. (2025) investiga de forma aprofundada como programadores que utilizam leitores de tela interagem com assistentes de código baseados em inteligência artificial avançada, como o *GitHub Copilot*, no contexto do paradigma emergente de *vibe coding*. Trata-se de um estudo longitudinal de duas semanas com 16 participantes, envolvendo tutoriais, execução de tarefas de programação que simulavam cenários reais, uso livre do assistente no cotidiano e entrevistas de acompanhamento. O objetivo central foi compreender como essas ferramentas podem empoderar usuários com deficiência visual e quais novos desafios surgem com sua adoção. Os resultados mostram que assistentes de código avançados podem ampliar a autonomia e a produtividade, permitindo que desenvolvedores realizem tarefas antes consideradas de difícil execução, como o desenvolvimento de interfaces gráficas, além de reduzir barreiras históricas de acessibilidade. Por outro lado, o estudo identifica desafios relevantes, como dificuldades em expressar claramente à IA a intenção do código desejado, na revisão das respostas geradas, na alternância entre múltiplas visões no ambiente de desenvolvimento, na manutenção da consciência situacional e na aprendizagem de recursos avançados. Também se discute o equilíbrio entre automação e controle, evidenciando que, embora modos mais autônomos aumentem a eficiência, muitos usuários preferem interações que exijam confirmação explícita para garantir previsibilidade e segurança.

Observa-se que os estudos analisados oferecem contribuições relevantes sobre acessibilidade digital e o uso da Inteligência Artificial em contextos educacionais e profissionais, mas diferem do escopo e da abordagem adotados nesta dissertação. Trabalhos como os de Zen et al. (2023) e Mountapmbeme et al. (2022) concentram-se na identificação de barreiras enfrentadas por estudantes cegos no ensino de programação, sem avançar para a proposição e avaliação de soluções práticas. Já Chemnad e Othman (2024) realizam uma revisão sistemática sobre acessibilidade digital na era da IA, enfatizando a importância de estratégias inclusivas, mas sem aplicação direta ao contexto de ambientes de desenvolvimento de código.

O estudo de Bhagat et al. (2024) aprofunda limitações de aplicativos móveis voltados a pessoas cegas, porém restringe-se ao domínio das interfaces móveis, não abordando o processo de aprendizagem em programação. Adicionalmente, Zen (2024) propõe diretrizes de acessibilidade para IDEs voltadas a estudantes cegos, contribuindo com recomendações de design; contudo, não avalia empiricamente o impacto pedagógico dessas diretrizes no processo de ensino-aprendizagem. Por sua vez, Chen et al. (2025) evidencia avanços na autonomia e adaptação de programadores cegos em ambientes modernos de codificação, mas mantém o foco em experiências profissionais, sem discutir implicações pedagógicas. Diante disso, esta dissertação diferencia-se ao integrar as dimensões de acessibilidade e aprendizagem, propondo e avaliando ferramentas de

IA em IDEs para investigar empiricamente seus impactos no ensino de programação para estudantes cegos. Assim, o trabalho transcende as abordagens predominantemente descritivas, apresentando contribuições práticas e pedagógicas no ensino de computação.

Capítulo 3

Metodologia

Este estudo configura-se como uma pesquisa qualitativa, pois busca entender as experiências, percepções e desafios enfrentados por programadores cegos ao utilizar ferramentas de Inteligência Artificial (IA) e Ambientes de Desenvolvimento Integrado (IDEs). O estudo busca compreender o fenômeno em seu contexto real, sem a interferência de variáveis controladas, o que é característico da abordagem qualitativa. Especificamente, trata-se de um estudo de caso exploratório, que investiga um fenômeno em profundidade em um contexto específico (neste caso, a acessibilidade de ferramentas de programação para estudantes cegos). O estudo de caso permite uma análise detalhada e contextualizada das experiências dos participantes. A escolha pelo estudo de caso justifica-se pela necessidade de uma análise aprofundada das interações dos participantes com as ferramentas assistidas por IA, permitindo a identificação de desafios, vantagens e adaptações necessárias para uma maior acessibilidade.

A pesquisa também pode ser considerada descritiva, pois busca descrever as características e percepções dos participantes em relação à usabilidade e à integração das ferramentas de IA e IDEs com recursos de tecnologia assistiva. Em resumo, a pesquisa é uma combinação de qualitativa, estudo de caso exploratório e descritiva, focando na análise das experiências de programadores cegos em um contexto específico. A metodologia envolve as seguintes etapas, conforme Figura 3.1: levantamento bibliográfico, experimentos práticos, coleta de dados e análise englobando a análise qualitativa e a análise de conteúdo, que serão detalhadas nas subseções a seguir.

3.1 Levantamento Bibliográfico

A pesquisa iniciou-se com um levantamento bibliográfico, visando identificar as principais **barreiras enfrentadas por estudantes com deficiência visual no aprendizado de programação**, bem como **compreender as soluções e tecnologias existentes, incluindo aquelas baseadas em IA**, que têm sido propostas para abordar esses desafios.

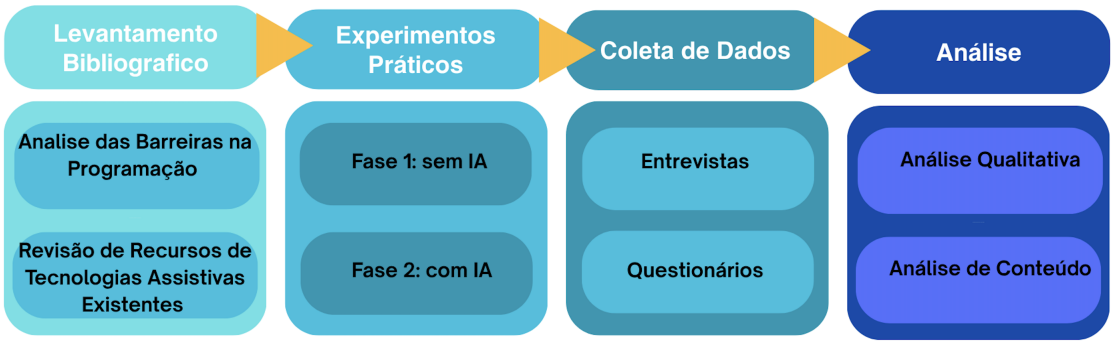

Figura 3.1: Esquema Representativo da Metodologia

Diferente de um mapeamento sistemático que segue protocolos rigidamente definidos para a seleção e categorização de estudos, este levantamento adotou uma abordagem sistematizada, reunindo procedimentos estruturados para identificar, selecionar e analisar criticamente a literatura existente. Essa estratégia permitiu uma busca ampla e orientada por critérios claros, sem, contudo, restringir a investigação às etapas formais de um protocolo rígido. A revisão bibliográfica foi conduzida entre setembro de 2024 e maio de 2025, considerando bases acadêmicas de referência na área de Computação e Educação, como ACM Digital Library, IEEE Xplore, Scopus, SpringerLink e Google Scholar. As buscas utilizaram combinações de palavras-chave em português e inglês, incluindo “artificial intelligence”, “accessibility”, “blind programmers”, “assistive technology”, “screen reader”, “programming education” e “integrated development environment”. Foram incluídos artigos publicados entre 2019 e 2025 que tratassem do uso de IA, recursos de tecnologia assistiva e/ou práticas educacionais voltadas à inclusão de pessoas cegas em contextos de programação. Após a remoção de duplicidades e a análise de títulos e resumos, dezesseis estudos foram selecionados, dos quais seis se destacaram pela relevância teórica e metodológica, sendo discutidos ao longo do Capítulo 2.

O levantamento teve como objetivo fornecer um panorama atualizado das pesquisas sobre acessibilidade na programação, guiando as próximas etapas do estudo. Os critérios de inclusão priorizaram artigos que abordavam barreiras enfrentadas por estudantes cegos, soluções educacionais, revisão de recursos de tecnologia assistiva e ferramentas de IA aplicadas ao ensino de programação. Trabalhos genéricos sobre acessibilidade digital, sem foco na programação, foram excluídos da análise.

A sistematização dos dados obtidos seguiu uma categorização temática, agrupando os estudos conforme (i) barreiras enfrentadas, (ii) soluções tecnológicas existentes, (iii) impacto da IA na acessibilidade e (iv) estratégias educacionais para ensino de programação. Essa abordagem possibilitou a síntese crítica dos achados, servindo como base para as etapas subsequentes de coleta de dados.

Diante das lacunas identificadas no levantamento bibliográfico, especialmente no que se refere à escassez de estudos empíricos que investiguem, na prática, a interação

de estudantes cegos com ambientes de programação com e sem o suporte de ferramentas de Inteligência Artificial, delineou-se um experimento dividido em fases, visando observar e comparar o desempenho e a experiência de programadores cegos em diferentes cenários.

3.2 Experimentos Práticos

Antes de apresentar os procedimentos adotados na realização dos experimentos práticos, faz-se necessário contextualizar o perfil dos participantes envolvidos na pesquisa.

3.2.1 Descrição dos Participantes

Critérios de Inclusão e Exclusão

A seleção dos participantes ocorreu de forma intencional, por meio de comunidades virtuais compostas por pessoas cegas que estudam na área de Computação. O primeiro contato foi estabelecido através do grupo de e-mails cegos_programadores@googlegroups.com, dedicado à troca de experiências e discussões sobre acessibilidade em tecnologias de programação. A partir desse grupo, foi possível obter acesso ao grupo de *WhatsApp* denominado "Pessoas Cegas Programadoras", que, à época, contava com aproximadamente 211 membros de diferentes regiões do Brasil. No grupo, foi enviada uma mensagem-convite apresentando os objetivos da pesquisa e convidando os interessados a participarem do estudo.

Os **critérios de inclusão** definidos foram:

- ser pessoa cega (cegueira total ou baixa visão grave);
- estudar na área de Computação;
- possuir experiência prévia em pelo menos uma linguagem de programação e uma IDE;
- apresentar familiaridade com o uso de leitores de tela (como NVDA, JAWS ou equivalentes);
- dispor de computador e acesso à internet para a realização dos experimentos de forma remota.

Os **critérios de exclusão** envolveram:

- ausência de familiaridade mínima com ambientes de desenvolvimento (IDEs) ou com o uso de leitores de tela;
- indisponibilidade de equipamentos ou conexão estável que permitissem a execução das tarefas.

Após a triagem das respostas obtidas por meio do formulário, dez participantes que atenderam aos critérios estabelecidos foram convidados a integrar o estudo. Buscou-se contemplar diferentes níveis de experiência, contextos acadêmicos e regiões geográficas, de modo a garantir uma amostra representativa e diversificada das experiências de pessoas cegas com a programação.

Sobre a Pesquisa

Os dez participantes com deficiência visual, Tabela 3.1, tinham idades entre 23 e 54 anos e eram oriundos de diferentes regiões do Brasil, incluindo Bahia, Pernambuco, Rio de Janeiro e São Paulo. Todos apresentavam cegueira total, sendo sete pessoas cegas de nascença e três que adquiriram a deficiência ao longo da vida. Observa-se uma predominância do gênero masculino no grupo, composto por oito homens e duas mulheres.

Em relação à formação acadêmica, os participantes apresentavam níveis educacionais diversos, variando desde cursos técnicos até pós-graduação em áreas relacionadas à tecnologia. Cinco deles estavam matriculados no curso de Sistemas de Informação, três cursavam Análise e Desenvolvimento de Sistemas, um era estudante de nível técnico em Desenvolvimento de Sistemas e outro já havia concluído a graduação, estando atualmente matriculado em uma pós-graduação em Arquitetura e Desenvolvimento Java.

Quanto à experiência em programação, o tempo de prática variou entre dois e vinte e quatro anos. Alguns participantes relataram ter iniciado os estudos de forma autodidata antes de ingressarem formalmente em cursos da área, conforme evidenciado na Tabela 3.2. Foi mencionada a familiaridade com diversas linguagens de programação, como Python, Java, C, C++, JavaScript e PHP. Além disso, os participantes relataram o uso de uma ampla gama de recursos de tecnologia assistiva, com destaque para leitores de tela como NVDA, JAWS, TalkBack, VoiceOver, Orca e WebVox, além de ferramentas como o Be My Eyes e o DOSVOX.

Esses dados revelam um grupo com trajetórias formativas heterogêneas, distintos níveis de experiência prática e diferentes graus de adaptação às tecnologias assistivas. Essa diversidade enriquece a análise dos dados, permitindo compreender de forma mais abrangente os desafios enfrentados por programadores cegos e as estratégias que empregam para superar as barreiras de acessibilidade no processo de aprendizagem e prática da programação.

Além disso, a composição desse grupo de participantes fornece uma base para investigar como fatores como tempo de experiência, formação acadêmica e tipo de deficiência visual influenciam na interação com ambientes de desenvolvimento e ferramentas de inteligência artificial. Essa perspectiva é essencial para identificar padrões de uso, necessidades específicas e potenciais caminhos para a construção de ambientes de aprendizagem mais inclusivos e eficazes.

Tabela 3.1: Perfil dos Participantes

Px	**Idade**	**Gênero**	**Origem**	**D.V.**	**Formação**
P1	27	Masc.	BA	Cego de nascença	Pós graduação em arquitetura e desenvolvimento java (online)
P2	54	Masc.	PE	Desenvolveu deficiência visual	Sistemas de Informação (online)
P3	33	Masc.	SP	Cego de nascença	Sistemas de Informação (presencial)
P4	25	Fem.	BA	Cega de nascença	Sistemas de Informação (presencial)
P5	23	Masc.	SP	Cego de nascença	Técnico Desenvolvimento de Sistemas (presencial)
P6	26	Masc.	RJ	Desenvolveu deficiência visual	Sistemas de Informação (presencial)
P7	34	Masc.	PE	Cego de nascença	Sistemas de Informação (presencial)
P8	23	Fem.	SP	Cega de nascença	Análise e Desenvolvimento de Sistemas (prresencial)
P9	35	Masc.	SP	Desenvolveu deficiência visual	Análise e Desenvolvimento de Sistemas (online)
P10	26	Masc.	SP	Cego de nascença	Análise e Desenvolvimento de Sistemas (presencial)

Nota: P_x representa o participante x, onde "x" indica o número do participante.

3.2.2 Descrição do Experimento

Os experimentos foram conduzidos de forma remota, utilizando a plataforma *Google Meet*[1] para facilitar a interação e gravação das atividades, com o consentimento livre e esclarecido dos participantes. Portanto, 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 (Apêndice A) foi obtido de todos os envolvidos antes do início das atividades, assegurando o respeito à autonomia e aos direitos dos participantes.

Assim, os experimentos foram realizados nos computadores pessoais dos participan-

[1] https://meet.google.com

Tabela 3.2: Linguagens, recursos assistivos e experiência dos participantes

Px	**Linguagens Utilizadas**	**Recursos de Tecnologias Assistivas**	**Experiência/Estudo com Programação**
P1	Java, C#, TypeScript	Leitores de tela NVDA (computador) e TalkBack (Android)	Trabalha há 5 anos, estuda há 12 anos
P2	Clipper, Assembler, Delphi, Pascal, Python	JAWS e DOSVOX	24 anos
P3	Python, JavaScript	JAWS, NVDA, TalkBack e VoiceOver	2 anos
P4	C, C++, Python, Java, JavaScript, PHP, BGT	NVDA e VoiceOver	Iniciou o curso em 2018, autodidata desde 2013
P5	C#, PHP, Python, C, JavaScript	NVDA, DOSVOX e Jishu (Android)	6 anos
P6	Python, Java, React, React Native	TalkBack, VoiceOver, JAWS, Narrator, NVDA	8 anos
P7	Python	Leitor de tela	6 anos
P8	C#, C++, C, Python, Java, HTML, CSS	NVDA e Be My Eyes (descritor de imagens)	3 anos e meio
P9	PHP, JavaScript, um pouco de Python	NVDA (PC) e Google TalkBack (smartphone)	Estuda programação há 15 anos; como deficiente visual desde 2018
P10	Python, Java, C, PHP, JavaScript	NVDA, TalkBack (celular), Orca, WebVox	3 anos

Nota: P_x representa o participante x, onde "x" indica o número do participante.

tes, logo já estavam configurados com ferramentas e leitores de tela que utilizavam rotineiramente, como NVDA e JAWS. A pesquisadora observou as sessões em tempo real e gravou a tela de cada transmissão para análise posterior. A pesquisa foi realizada individualmente com cada participante. 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 as gravações asseguraram a coleta de dados detalhada e completa para análise de dados, respeitando as condições éticas do estudo.

Os Experimentos Práticos foram estruturados, conforme a sequência didática (Apêndice B), em três etapas principais, descritas na Figura 3.2, sendo a **primeira** dedicada à apresentação e contextualização dos experimentos. Durante os 15 minutos iniciais, os participantes foram informados sobre os objetivos da pesquisa. 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 o mesmo fosse desenvolvido do zero. Além disso, foram anotados quais leitores de tela e ferramentas de IA seriam utilizados, como o NVDA, JAWS, ChatGPT 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.

3.2.3 Fase 1: sem IA

Chamamos a **segunda etapa** de Fase 1 – sem IA, conforme Figura 3.2. Nela, estudantes cegos realizaram tarefas de programação utilizando exclusivamente recursos tradicionais de tecnologia assistiva, como leitores de tela e softwares acessíveis (por exemplo, WebVox[2]). Essa configuração permitiu observar, de forma isolada, as estratégias cognitivas, operacionais e tecnológicas adotadas pelos participantes. A atividade prática consistiu na **implementação de um algoritmo para organizar uma sequência numérica em ordem crescente, sem o uso de funções prontas ou ferramentas de Inteligência Artificial**. Os participantes podiam optar por utilizar uma lista predefinida ou permitir a inserção manual dos valores. Foram autorizadas consultas técnicas por meio de pesquisas *online*, desde que não envolvessem soluções baseadas em IA. Durante a execução, analisaram-se os principais desafios enfrentados, o tempo necessário para a implementação e os tipos de erros cometidos.

O experimento foi realizado de forma remota, com os participantes utilizando seus próprios dispositivos, ambientes de desenvolvimento integrados (IDEs) e leitores de tela já previamente configurados e conhecidos, buscando preservar a naturalidade das condições em que costumam programar. Cada sessão foi acompanhada em tempo real pela pesquisadora, por meio de videochamadas no *Google Meet*. Para fins de registro e análise posterior, as sessões foram gravadas com consentimento dos participantes, e utilizou-se a extensão IA TacTiq[3] para realizar a transcrição automática das falas, facilitando o levantamento dos dados qualitativos.

Durante a atividade, registraram-se o tempo de conclusão, as dificuldades encontradas e observações feitas pelos participantes, seja por anotações ou relatos verbais. As coletas ocorreram de forma remota e síncrona, com uso de compartilhamento de tela para acompanhamento integral da tarefa.

3.2.4 Fase 2: com IA

Na **terceira etapa** ou fase 2-com IA, 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,

[2]https://webvox.en.softonic.com/chrome/extension
[3]http://tactiq.io

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.

3.3 Coleta de Dados

Entrevista semi-estruturada e Questionário Online

Após as atividades práticas, foi conduzida uma **quarta etapa** (Figura 3.2) de coleta de informações e discussão, com duração de 30 minutos. Foi realizada uma entrevista semi-estruturada (Apêndice C) e aplicado um questionário acessível (Apêndice D) para registrar as percepções dos participantes sobre as atividades realizadas. Essas etapas permitiram 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.

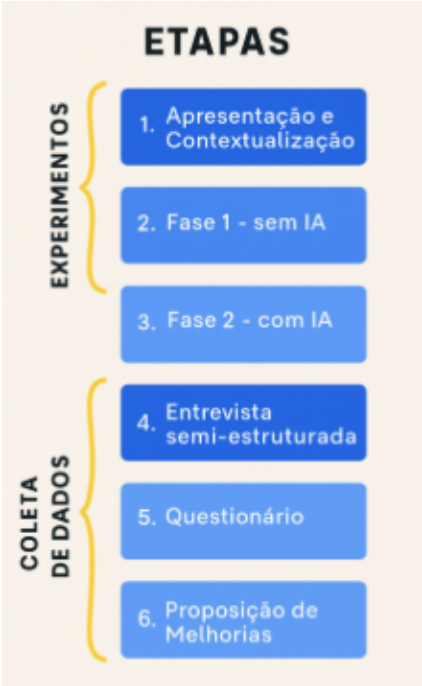

Figura 3.2: Etapas dos Experimentos Práticos e Coleta de Dados

A **última etapa** (Figura 3.2) 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 acessibilidade 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 tempo para correção das atividades com e sem IA, que serão discutidos no Capítulo 4. 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.

3.4 Análise

Para a análise dos dados qualitativos coletados durante o experimento, adotou-se a técnica de Análise de Conteúdo, conforme proposta por Bardin (2011), com foco na identificação de categorias temáticas emergentes a partir das entrevistas, das observações e dos formulários aplicados aos participantes. Essa abordagem foi escolhida por sua capacidade de revelar padrões de sentido, percepções e estratégias subjetivas em contextos educacionais, especialmente em estudos que envolvem sujeitos com deficiência.

3.4.1 Análise Qualitativa

A análise qualitativa dos dados teve como objetivo interpretar as percepções, experiências e significados atribuídos pelos participantes às atividades de programação realizadas com e sem o uso de Inteligência Artificial (IA). Essa abordagem buscou compreender de que maneira as ferramentas baseadas em IA influenciaram aspectos como acessibilidade, autonomia, produtividade e satisfação dos estudantes cegos ao interagir com diferentes ambientes de desenvolvimento integrados (IDEs).

Os dados qualitativos foram obtidos a partir de três fontes principais: (i) entrevistas semiestruturadas realizadas após as etapas experimentais; (ii) questionários abertos aplicados aos participantes para coleta de percepções individuais; e (iii) observações diretas feitas durante as sessões experimentais, registradas em notas de campo e gravações de tela. A triangulação dessas fontes possibilitou uma visão abrangente das experiências relatadas e observadas.

A análise qualitativa foi integrada à análise quantitativa, de modo que ambas se complementassem. Enquanto a análise quantitativa mediu o tempo de execução e o número de erros nas atividades com e sem IA, a análise qualitativa interpretou como e por que essas diferenças ocorreram, destacando as percepções dos participantes sobre os benefícios e limitações das ferramentas de IA. Essa integração possi-

bilitou uma compreensão mais profunda do impacto educacional e da acessibilidade proporcionada pelas soluções de IA em ambientes de programação.

Por fim, os resultados da análise qualitativa subsidiaram a discussão apresentada no Capítulo 5, na qual são detalhados os aspectos pedagógicos e tecnológicos identificados ao longo da pesquisa, fortalecendo a interpretação das evidências empíricas e orientando recomendações para o desenvolvimento de práticas e ferramentas mais acessíveis.

3.4.2 Análise de Conteúdo

As principais categorias de análise emergiram de forma indutiva, contemplando os seguintes eixos: *barreiras de acessibilidade*, *impacto da IA na autonomia e produtividade*, *limitações na integração entre leitores de tela e IDEs*, e *sugestões de melhoria das ferramentas assistivas*. Essa categorização permitiu compreender não apenas os resultados objetivos do experimento, mas também os aspectos subjetivos envolvidos na interação dos participantes com as tecnologias utilizadas.

3.4.3 Etapas da Análise de Conteúdo

- **Pré-análise:** Nesta fase, foi realizada a leitura flutuante das entrevistas transcritas e das respostas aos questionários (corpus da análise). O objetivo foi apreender o sentido geral das falas e identificar unidades de registro relevantes.
- **Codificação:** Foram destacados trechos significativos dos relatos dos participantes, que expressavam opiniões, dificuldades, experiências e sugestões sobre acessibilidade na programação.
- **Categorizacão:** A partir da codificação, emergiram categorias temáticas e subcategorias que representam os núcleos de sentido mais recorrentes nos dados.
- **Inferência e Interpretação:** Com base nas categorias, foram realizadas inferências que relacionam os dados empíricos com os objetivos da pesquisa e com a literatura sobre acessibilidade, educação e inteligência artificial.

3.4.4 Categorias Temáticas Emergentes

Os dados foram organizados em seis categorias principais, com suas respectivas subcategorias. A Tabela 3.3 apresenta esse agrupamento.

A partir destas categorias, a análise de conteúdo permitiu interpretar as experiências e percepções dos participantes, revelando padrões de sentido sobre as barreiras de acessibilidade, o impacto da IA na autonomia e produtividade, e as limitações na integração entre leitores de tela e ambientes de desenvolvimento. Essa abordagem

Tabela 3.3: Categorias temáticas e subcategorias identificadas

Categoria	**Definição**	**Subcategorias (Exemplos)**
Acessibilidade em Ambientes de Programação	Refere-se à facilidade de uso de IDEs, leitores de tela e infraestrutura por pessoas cegas.	- Leitor de tela ineficiente - IDEs pouco acessíveis (Eclipse, Colab) - VS Code preferido pela compatibilidade
Desafios Técnicos na Programação	Dificuldades práticas enfrentadas ao codificar, depurar e compreender sintaxe/lógica.	- Depuração de erros - Interpretação de mensagens - Estruturação do código - Pouco feedback do leitor
Recursos Educacionais e Inclusão	Refere-se à adequação de materiais didáticos, avaliações e apoio institucional.	- Falta de material adaptado - Professores despreparados - Aulas em vídeo inacessíveis - Falta de apoio dos núcleos de inclusão
Uso e Limites da Inteligência Artificial	Experiências e percepções sobre IA como ferramenta de apoio para programação e acessibilidade.	- ChatGPT, Cursor, Replit usados - Problemas com contexto em códigos longos - IA sem integração com leitores - Desejo por IA específicas para cegos
Estratégias Individuais de Superação	Estratégias autônomas adotadas para lidar com as barreiras.	- Pedido de ajuda a colegas - Busca em tutoriais - Consulta a fóruns - Tentativa e erro

qualitativa complementou os resultados quantitativos, possibilitando uma compreensão mais ampla dos efeitos pedagógicos e tecnológicos das ferramentas de IA no processo de aprendizagem de programação por estudantes cegos.

Capítulo 4

Resultados

Este capítulo apresenta os resultados obtidos a partir do experimento realizado com estudantes cegos, cujo objetivo foi avaliar a acessibilidade e a usabilidade de ferramentas de Inteligência Artificial (IA) e ambientes de desenvolvimento integrados (IDEs) no ensino de programação. As análises contemplam dados quantitativos e qualitativos, permitindo compreender de forma abrangente como o uso de IA influencia a produtividade, a autonomia e a experiência desses estudantes no processo de codificação.

Apresentamos, a seguir, o experimento conduzido em duas condições distintas: sem o uso de IA e com o uso de IA. A comparação entre esses cenários possibilita identificar barreiras técnicas e de acessibilidade, assim como benefícios e limitações na integração entre leitores de tela e ferramentas de IA. Na sequência, discutimos os achados obtidos por meio da análise de conteúdo das entrevistas e dos questionários, estruturados em categorias temáticas que abrangem acessibilidade nos ambientes de programação, desafios técnicos na Programação, recursos educacionais e inclusão, uso e limitações da IA e estratégias individuais de superação. Por fim, os resultados são articulados com a literatura, evidenciando as contribuições deste estudo para o avanço de práticas e tecnologias mais inclusivas no ensino de programação para pessoas cegas.

4.1 O Experimento

O experimento realizado neste estudo foi estruturado a partir da sequência didática descrita no Apêndice B, composta por quatro etapas: a) contextualização, b) atividade sem uso de IA, c) atividade com uso de IA e d) discussão final. Inicialmente, os participantes foram apresentados ao problema proposto e orientados quanto ao uso das ferramentas permitidas em cada fase.

Na primeira atividade prática, os estudantes cegos deveriam resolver um algoritmo de ordenação sem utilizar recursos de Inteligência Artificial, apenas com o suporte

de leitores de tela e os ambientes de desenvolvimento de sua escolha. Em seguida, na segunda etapa, o mesmo problema foi reapresentado, permitindo-se agora o uso de ferramentas baseadas em IA, como o *ChatGPT*, *GitHub Copilot* ou *Google Colab*, para obter sugestões, correções ou mesmo gerar código.

Ao final, foi realizada uma entrevista semiestruturada para discutir percepções, dificuldades e comparações entre as duas abordagens. Essa sequência possibilitou analisar o impacto da IA na produtividade, usabilidade e acessibilidade do processo de programação, permitindo uma avaliação qualitativa da experiência dos participantes com e sem o apoio dessas tecnologias.

4.1.1 Contextualização

Durante a contextualização, 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 (IDEs ou Editor de Texto) e os leitores de tela foram testados pelos próprios participantes, com a orientação da pesquisadora, para garantir pleno funcionamento, conforme as preferências individuais, nos computadores pessoais dos participantes. Decisões importantes foram tomadas, como permitir o uso de ferramentas com as quais os participantes já estavam familiarizados, evitando uma curva de aprendizado desnecessária para novas tecnologias e o uso de consultas a *sites* para relembrar o funcionamento do algoritmo.

4.1.2 Atividade sem uso de IA

O objetivo dessa etapa, conforme Seção 1.3, é compreender as principais barreiras de acessibilidade enfrentadas por estudantes cegos em ambientes de programação. 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 crescente para uma lista de números de forma manual, sem o uso de funções prontas**, o que revelou barreiras técnicas e cognitivas específicas para cada linguagem escolhida. O leitor de tela adotado pela maioria dos participantes foi o NVDA, escolhido por ser gratuito, leve e compatível com os principais ambientes de desenvolvimento. A exceção foi o participante 2, que além do NVDA utilizou o WebVox[1] (um leitor web gratuito que permite aos cegos navegar na Internet). As linguagens escolhidas refletiram o repertório e a familiaridade prévia de cada participante: os Participantes 2, 3, 5, 6, 7, 8 e 10 optaram por Python; o Participante 1 utilizou Java; o Participante 4 escolheu C++; e o Participante 9 utilizou PHP.

Iremos adotar que os Participantes serão chamados de Px, onde x varia de 1 a 10, assim P1 se refere ao Participante 1 e assim, sucessivamente. A Tabela 4.1 apresenta

[1] https://chromewebstore.google.com/detail/webvox

um quadro comparativo dos dez participantes na etapa sem o uso de IA, sintetizando dados como linguagem de programação utilizada, tipo de algoritmo implementado, tempo despendido para escrever e corrigir o código, consulta a recursos externos e IDEs empregadas. Cada participante pôde escolher a linguagem e o ambiente de desenvolvimento com os quais já possuía familiaridade, o que permitiu focar especificamente nas dificuldades relacionadas à acessibilidade, compreensão de mensagens de erro, organização do código e interação com leitores de tela.

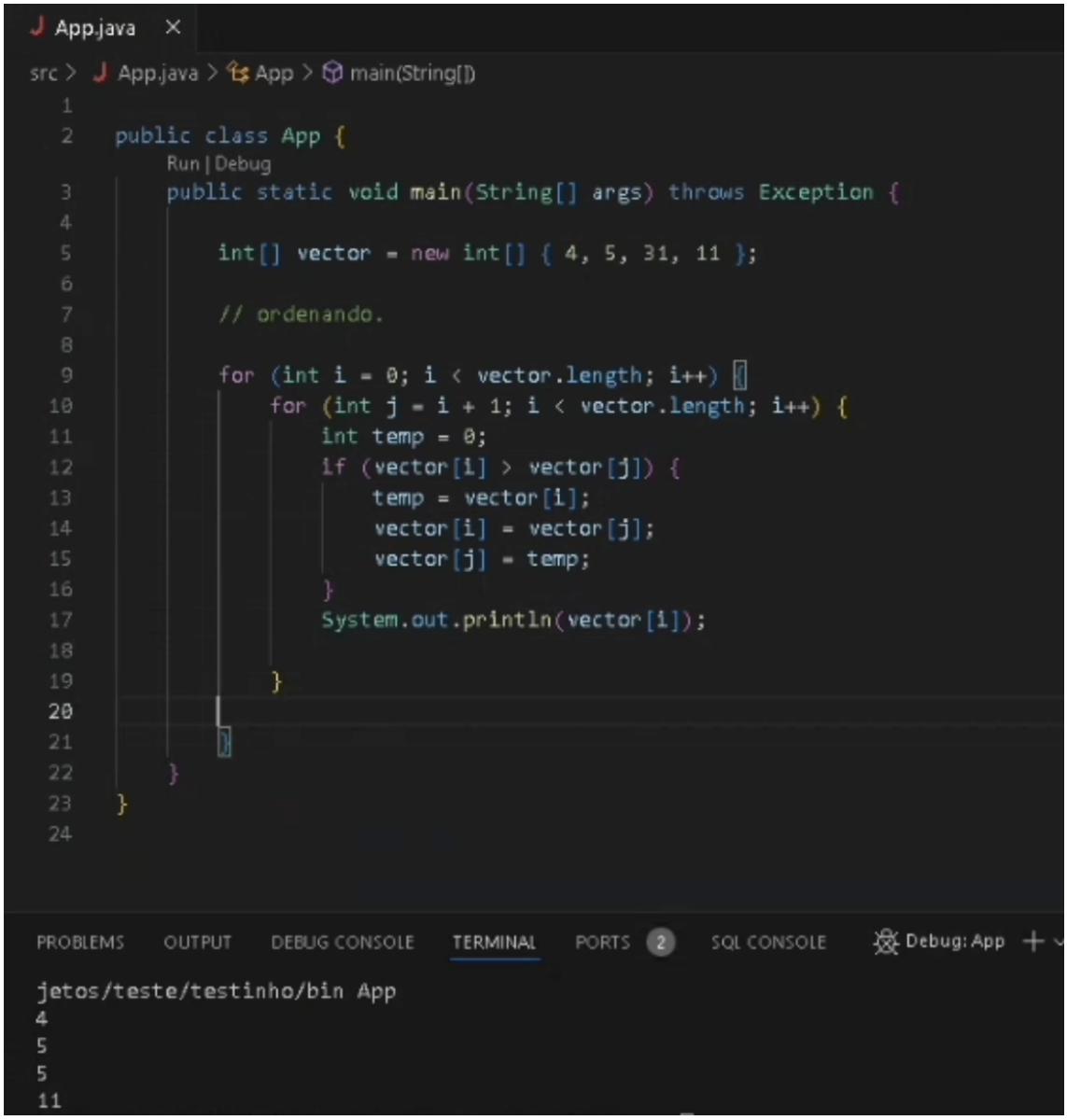

Figura 4.1: Tela de Implementação do P1

P1 tentou inicialmente construir o código do zero, mas enfrentou obstáculos tanto na definição do vetor quanto na estruturação correta dos laços de repetição. Apesar de buscar referências externas, sua implementação continha diversos equívocos lógicos:

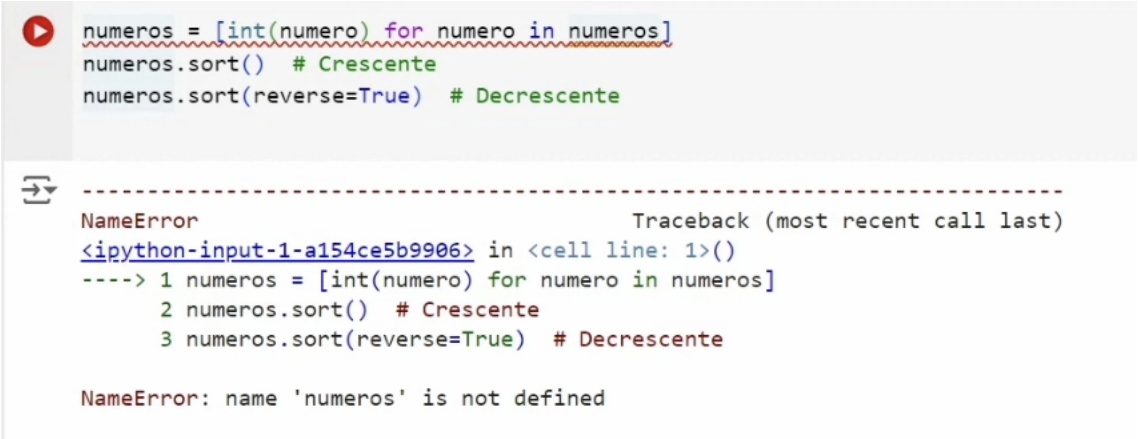

Figura 4.2: Tela de Implementação do P2

condições de parada incorretas nos laços `for`, incrementos inadequados, além da impressão do vetor ser realizada dentro do *loop*, resultando em saídas parciais e incorretas, como podemos observar na Figura 4.1. O tempo total gasto sem IA foi de **128 minutos (61 para a implementação inicial e 67 para a tentativa de correção)**, evidenciando não apenas a complexidade da tarefa para um estudante cego, mas também as limitações do ambiente sem suporte inteligente. Embora o código executasse sem erros de compilação, os problemas lógicos persistiam, e o participante considerou a tarefa finalizada, mesmo sem ter atingido a saída correta. Esses achados reforçam a importância de ambientes mais acessíveis, que ofereçam recursos de apoio à lógica e à estruturação do código.

Apesar de demonstrar familiaridade com o Colab e buscar soluções online, P2 enfrentou **dificuldades significativas na compreensão e adaptação de códigos disponíveis em fóruns como o *Stack Overflow***. O leitor de tela não conseguiu interpretar adequadamente a estrutura do código que ele copiou do fórum, o que dificultou a depuração e a personalização para atender à proposta da tarefa. O código utilizado incluía elementos desnecessários, como ordenações mistas (crescente e decrescente), e não apresentava de forma clara a definição da lista nem a estrutura de repetição esperada, como podemos ver na Figura 4.2. Após cerca de 60 minutos de tentativas, o participante não conseguiu finalizar o programa e optou por encerrar a atividade. Esse resultado evidencia como a acessibilidade limitada na leitura e compreensão de código compromete a autonomia do estudante e reforça a importância de ferramentas que ofereçam suporte contextual e adaptado às demandas da pessoa cega.

Durante a atividade, foram identificados diversos erros que comprometeram a execução do programa do P3, **como um espaço indevido no nome da função (`bubble _ sort`), o uso de uma variável (`n`) não declarada, e erros de indentação que prejudicaram a estrutura lógica do algoritmo**, Figura 4.3. Esses problemas, ainda que sutis visualmente, foram de difícil percepção com o uso do leitor de tela. A leitura genérica das mensagens de erro no terminal, como `syntax error near unexpected`

`token '('`, não fornecia pistas claras sobre a origem das falhas, tornando o processo de depuração especialmente desafiador. O participante concluiu a implementação inicial em 17 minutos, mas levou mais 70 minutos para tentar corrigir os erros, sem recorrer a buscas externas, baseando-se exclusivamente em seu conhecimento prévio. O caso de P3 reforça a importância de ambientes de desenvolvimento que ofereçam *feedbacks* mais claros e acessíveis para apoiar a autonomia de programadores com deficiência visual.

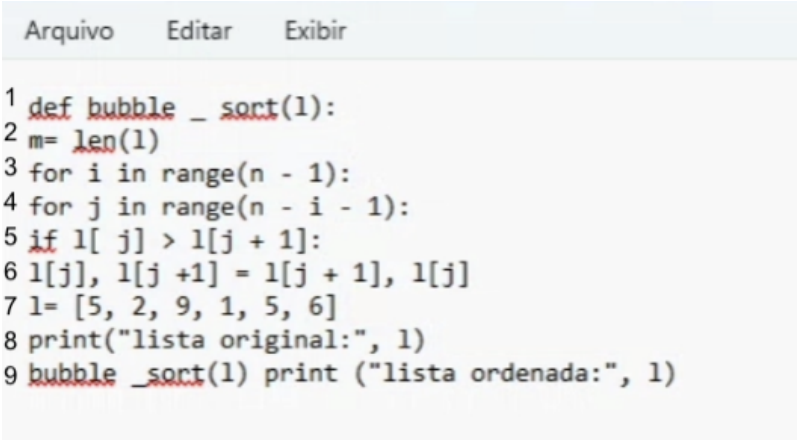

Figura 4.3: Tela de Implementação do P3

P4, que utilizou a linguagem C++, enfrentou dificuldades para instalar as extensões necessárias e para navegar na interface do VS Code, especialmente ao interagir com caixas de diálogo e notificações, o que atrasou o início da codificação. Durante a escrita do código, surgiram problemas relacionados à **declaração e manipulação de vetores, erros de sintaxe em estruturas de repetição e uso inadequado de bibliotecas** - Figura 4.4. A implementação apresentou falhas como a ausência da biblioteca `<utility>` para o uso da função `swap`, erros de pontuação como o uso incorreto de `std>:swap` em vez de `std::swap`, omissão de chaves em blocos `for`, e vírgulas e colchetes mal posicionados na declaração do vetor. Mesmo após consultar materiais *online*, o participante levou 85 minutos para concluir a versão inicial, e outros 90 minutos para tentar corrigir os erros, demonstrando dificuldades em adaptar as soluções encontradas à sua estrutura. A saída do programa também não incluía espaçamentos adequados, comprometendo a clareza dos resultados apresentados. Esses achados evidenciam que, mesmo com conhecimento técnico, a falta de acessibilidade e apoio contextualizado no ambiente de programação pode impactar fortemente a autonomia e o desempenho de programadores cegos.

P5 enfrentou diversas **dificuldades estruturais e lógicas, especialmente na manipulação de listas e na implementação do laço de repetição para ordenação**, utilizando a linguagem de programação Phyton. Um problema significativo ocorreu durante a exibição dos dados: mesmo que os números estivessem presentes no terminal, o leitor de tela não os lia adequadamente ao pressionar a tecla `<Enter>`, o que dificultou a verificação da saída e prolongou o processo de depuração. **A lista foi inicializada**

Figura 4.4: Tela de Implementação do P4

de forma incorreta, armazenando um valor único em vez de permitir a inserção progressiva dos dados. Outros erros envolveram o uso desnecessário de variáveis auxiliares, tentativa de acesso a índices inválidos e redundância no uso da função `sort()` (Figura 4.5). O participante concluiu a implementação inicial em 97 minutos e dedicou mais 56 minutos à busca por soluções *online* para resolver os erros encontrados. Durante essa busca, localizou exemplos de código que se mostraram relevantes e conseguiu adaptá-los para corrigir os problemas existentes. O caso de P5 demonstra como falhas na comunicação entre o ambiente de desenvolvimento e o leitor de tela comprometem a autonomia do programador e tornam o processo significativamente mais demorado e complexo.

Apesar de recorrer a uma pesquisa na internet para obter o código do `Bubble Sort`, P6 enfrentou uma série de dificuldades ao longo da tarefa, implementando também em Phyton. As principais barreiras envolveram **aspectos básicos da digitação e estruturação do código, como ausência de dois-pontos (:), após comandos condicionais, uso inadequado de vírgulas e problemas frequentes de indentação** (Figura 4.6), que afetaram diretamente a execução do programa. Essas falhas, embora sim-

```
1 #encoding: utf-8
2 tamanho = input("Por favor, Digite o tamanho da lista:")
3 quantidade_numero = [tamanho]
4 for obter in range(0, int(tamanho)):
5     obter = obter + 1
6     print("Digite", format(obter), "número:")
7     quantidade_numero.append(int(input()))
8 quantidade_numero.sort(key=int)
9 for exibir in range(0, int(tamanho)):
10    print(quantidade_numero[exibir])
```

Figura 4.5: Tela de Implementação do P5

ples para usuários videntes, tornaram-se críticas em razão das **limitações do leitor de tela, que não sinalizava com precisão a localização dos erros, muitas vezes indicados visualmente por sublinhados ou serrilhados no VS Code**. P6 também teve dificuldades para compreender as mensagens de erro exibidas no terminal e interpretar corretamente os retornos do NVDA, como quando o leitor indicava apenas números genéricos relacionados à posição do cursor ou à linha. Do ponto de vista conceitual, P6 demonstrou fragilidade na compreensão da lógica de programação, especialmente no que diz respeito à estrutura dos laços `for` utilizados no algoritmo `Bubble Sort`. As dúvidas surgiram em relação aos limites do laço, à lógica de ordenação e ao funcionamento interno do algoritmo. Ao longo da atividade, foi necessário intervir com explicações adicionais para que ele conseguisse compreender os fundamentos da ordenação e corrigir seu código. P6 levou 44 minutos para realizar a implementação inicial e mais 16 minutos para ajustes e depuração, totalizando uma hora de atividade. Sua experiência evidencia como a junção de barreiras técnicas, acessibilidade limitada e lacunas conceituais pode comprometer significativamente a autonomia e a produtividade de estudantes cegos em ambientes de programação tradicionais.

Inicialmente, o P7 tentou construir um código manualmente, mas enfrentou dificuldades com a manipulação de listas e com a estruturação da função de ordenação, o que o levou a buscar uma solução na internet. Ao encontrar e copiar um código baseado no algoritmo `Insertion Sort`, Figura 4.7, P7 tentou adaptá-lo à lista que havia criado, substituindo variáveis e ajustando o conteúdo. No entanto, **erros de indentação surgiram durante a edição, dificultando a execução correta do código**. Como o VS Code sinalizava erros apenas por meio de um som (“bip”) e sem indicar a natureza específica do problema, o participante teve que navegar linha por linha para identificar a falha. Após a correção da indentação, o código foi executado, mas sem gerar saída visível, já que não havia nenhum comando de impressão incluído. Essa ausência gerou confusão, prolongando ainda mais a depuração. Ao identificar o problema e inserir a chamada da função, o participante acabou posicionando-a incorretamente dentro da própria definição, resultando em uma chamada recursiva

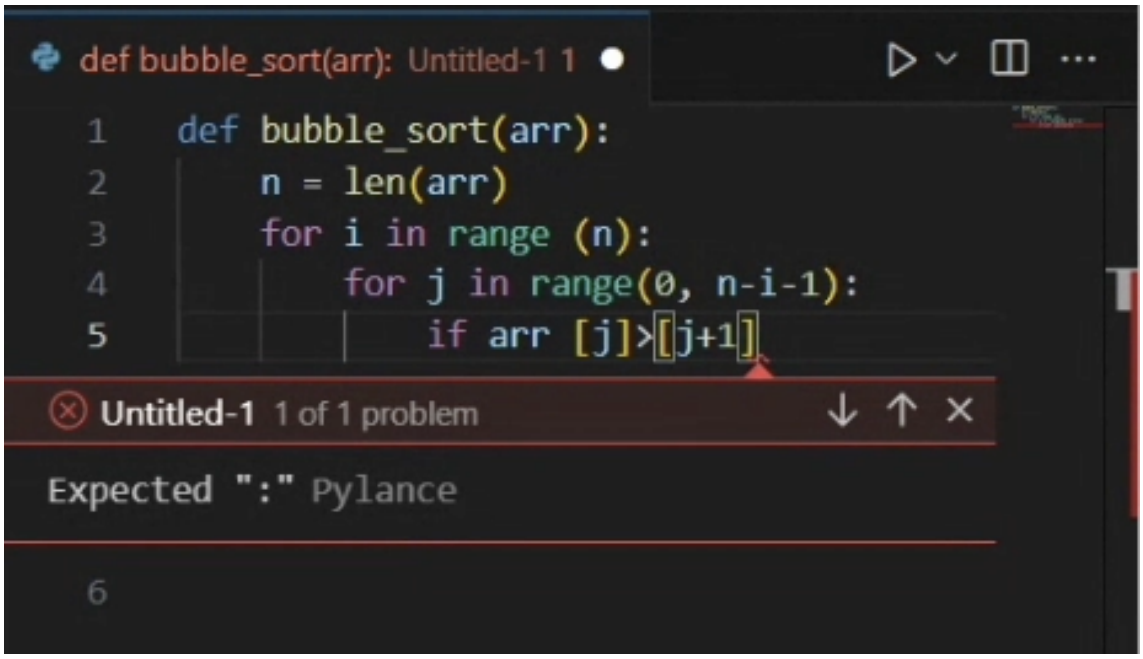

Figura 4.6: Tela de Implementação do P6

e em um loop infinito. Somente após diversos ajustes, o código foi executado corretamente. No total, P7 levou cerca de 127 minutos para concluir a atividade, demonstrando como aspectos sutis, como indentação automática e retorno genérico de erros, impactam significativamente o processo de programação para pessoas cegas.

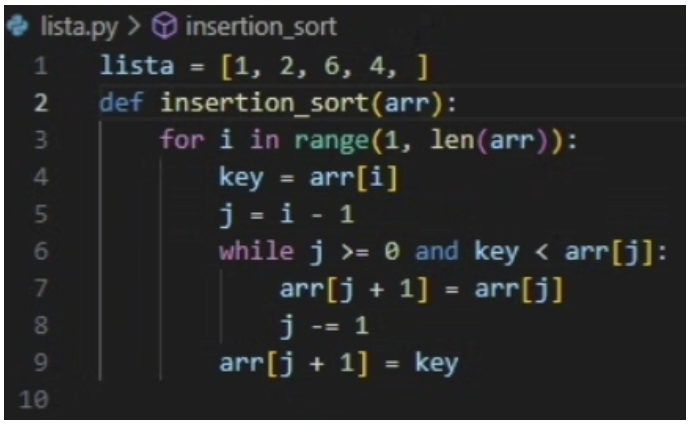

Figura 4.7: Tela de Implementação do P7

Embora tenha demonstrado domínio na construção do algoritmo, P8 enfrentou um contratempo técnico: um conflito entre extensões do VS Code, configuradas para Java, que dificultou momentaneamente a execução do código em Python. Ainda assim, encontrou alternativas para testar o funcionamento do programa. P8 também relatou questões relacionadas à **acessibilidade do ambiente, como a exibição de informações irrelevantes no terminal após a execução do código, o que compromete**

a leitura para usuários de leitores de tela. Para lidar com aspectos de usabilidade, P8 mencionou utilizar configurações específicas no NVDA que produzem sinais sonoros conforme a profundidade da indentação e notificações sonoras para a presença de erros, facilitando a navegação e a correção do código. Apesar desses desafios pontuais, P8 conseguiu realizar a atividade com autonomia e eficiência (Figura 4.8). O tempo total para concluir essa primeira etapa foi de aproximadamente 20 minutos, sendo 17 minutos para a construção do código com o auxílio da internet e 3 minutos para ajustes.

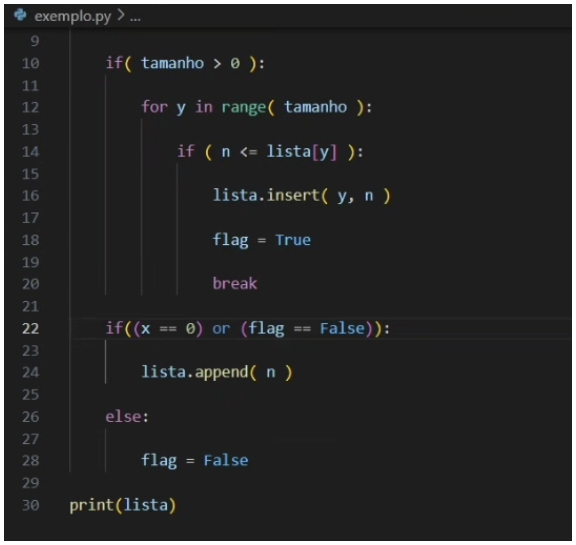

Figura 4.8: Tela de Implementação do P8

P9 enfrentou diversos problemas de acessibilidade que dificultaram sua autonomia e fluidez na realização da tarefa. Um dos principais obstáculos foi a **dificuldade em identificar erros no código**, já que o leitor de tela NVDA não sinalizava de forma clara a localização das falhas apontadas pela IDE, comprometendo o processo de depuração. Além disso, **o ambiente do Visual Studio Code apresentava excesso de informações visuais que, ao serem lidas automaticamente pelo leitor, causavam sobrecarga auditiva e dificultavam a concentração**. P9 também não utilizava recursos sonoros específicos para indicar níveis de indentação, o que pode ter prejudicado a percepção da estrutura do código. Somado a isso, enfrentou dificuldades na navegação pelo editor, especialmente para manter o foco alinhado com o cursor de escrita (a posição do cursor nem sempre era clara, e o foco do leitor de tela nem sempre acompanhava o ponto de edição corretamente, o que gerava insegurança na manipulação do código). Esses fatores evidenciam barreiras importantes que ainda persistem nos ambientes de programação e que impactam diretamente a experiência de usuários cegos. P9 iniciou a resolução da tarefa utilizando a linguagem PHP, tentando implementar um algoritmo de ordenação com base em um exemplo previamente encontrado, semelhante ao `Bubble Sort`. Ele levou aproximadamente 21 minutos para copiar, adaptar e testar esse código, porém, devido a um erro na lógica de atualização dos elementos, causado pelo uso incorreto de um *array* auxiliar,

o algoritmo não produziu o resultado esperado. Diante da falha e da dificuldade em identificar o problema, o participante optou por apagar completamente o código e recomeçar a tarefa. Em sua segunda tentativa, decidiu utilizar o algoritmo `Selection Sort`, que também encontrou na internet, adaptando-o ao seu contexto. Essa nova abordagem exigiu maior esforço de compreensão e ajustes, levando cerca de 43 minutos até que o código estivesse funcional. Assim, o tempo total investido por ele para concluir a primeira etapa foi de aproximadamente 64 minutos.

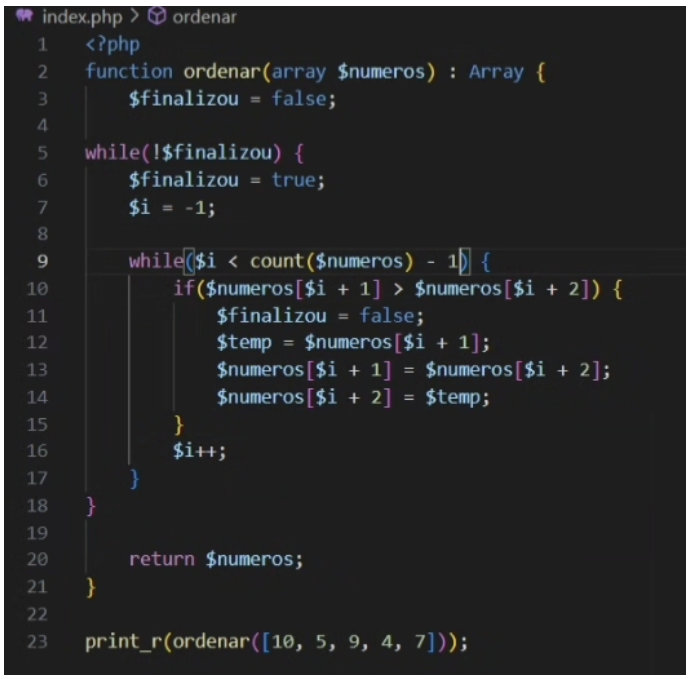

Figura 4.9: Tela de Implementação do P9

Durante o experimento, P10 enfrentou **diversas dificuldades, como as limitações técnicas do computador utilizado — um modelo antigo e de baixo desempenho, que provocava lentidão, travamentos e atrasos no compartilhamento de tela — e problemas recorrentes com o leitor de tela, que parava de responder após abrir pastas ou projetos, exigindo reinicialização**. Também relatou a falta de referências e suporte para discutir programação acessível, o que gera isolamento e insegurança na resolução de problemas. Teve dificuldade em lembrar a lógica completa de algoritmos de ordenação sem uso de funções prontas, com erros de formatação e indentação que afetaram o resultado do programa, além de dúvidas iniciais sobre como estruturar a entrada de dados. O P10 precisou de 82 minutos para desenvolver o algoritmo, com auxílio de pesquisas na internet e de mais 10 minutos para correção de erros.

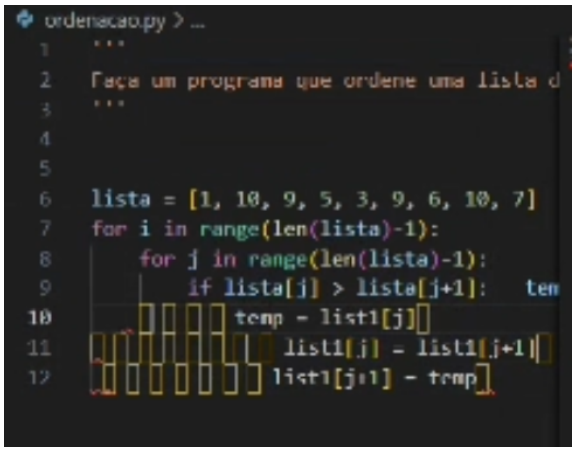

Figura 4.10: Tela de Implementação do P10

Síntese:
A análise qualitativa dos relatos e do desempenho dos participantes evidenciou limitações tanto técnicas quanto cognitivas, agravadas por interfaces pouco acessíveis e ausência de recursos de apoio automatizado.
Os relatos apresentados nesta seção evidenciam, de forma concreta, os obstáculos enfrentados por estudantes cegos ao programarem sem o apoio de ferramentas baseadas em Inteligência Artificial. As dificuldades recorrentes como: problemas de indentação, interpretação de mensagens de erro, ausência de sugestões contextuais e limitações na navegação pelo código, revelam um cenário de acessibilidade ainda distante do ideal. Mesmo com domínio básico das linguagens e o uso de leitores de tela como NVDA, os participantes demonstraram que a falta de integração entre recursos assistivos e ambientes de desenvolvimento compromete diretamente a autonomia, a usabilidade e a produtividade. Esses achados reforçam a importância de se repensar a acessibilidade digital no ensino de programação, especialmente quando se busca garantir uma experiência de aprendizagem equitativa e inclusiva.

A partir da Tabela 4.1, podemos concluir que, na etapa sem uso de IA, houve grande variação no desempenho dos participantes, tanto no tempo de implementação quanto de correção, evidenciando diferentes níveis de domínio em programação e familiaridade com os algoritmos escolhidos. A maioria recorreu a pesquisas na *web*, o que sugere necessidade de apoio externo para solucionar dúvidas, e predominou o uso do VS Code como IDE, ainda que alguns tenham optado por ferramentas como Colab, Repl.it ou até mesmo o Bloco de Notas. Observa-se também a preferência por algoritmos simples e amplamente conhecidos, como Bubble Sort, indicando possível limitação de repertório ou escolha por métodos de implementação mais acessíveis para a tarefa.

Tabela 4.1: Quadro comparativo dos participantes na etapa sem uso de IA

Px	**Linguagem de Programação**	**Algoritmo**	**Tempo (fazer / corrigir)**	**Pesquisou na Web?**	**IDE**
P1	Java	Consulta sobre ordenação	61 min / 67 min	Sim	VS Code
P2	Python	Funções prontas	60 min / desistiu	Sim	Colab
P3	Python	Bubble Sort	17 min / 70 min	Não	Bloco de Notas / VS Code
P4	C++	Bubble Sort	85 min / 90 min	Sim	Repl.it / VS Code
P5	Python	Consulta sobre ordenação	97 min / 56 min	Sim	Bloco de Notas / VS Code
P6	Python	Bubble Sort	44 min / 16 min	Sim	VS Code
P7	Python	Insertion Sort	65 min / 62 min	Sim	VS Code
P8	Python	Ordenação com Flag	17 min / 3 min	Sim	VS Code
P9	PHP	Selection Sort	21 min / 43 min	Sim	VS Code
P10	Python	Bubble Sort	82 min / 10 min	Sim	VS Code

4.1.3 Atividade com uso de IA

Os objetivos dessa etapa, conforme Seção 1.3, são: investigar a integração entre leitores de tela e ferramentas de IA, avaliando seus impactos na compreensão do código e analisar as percepções dos participantes sobre o uso dessas ferramentas, considerando aspectos como confiabilidade, acessibilidade, clareza das sugestões e dependência de ajuda externa.

Cada participante teve a liberdade de escolher a ferramenta de IA, permitindo que trabalhassem com tecnologias com as quais já estavam familiarizados. A tarefa proposta envolvia a **implementação de um algoritmo de ordenação crescente para uma lista de números de forma manual, sem o uso de funções prontas**, ou seja, o mesmo problema, mas com o auxílio de ferramentas de IA. A orientação foi que era possível ajustar o código da primeira etapa ou construir um código novo para ser comparado com o que havia sido feito (na etapa anterior - sem o uso de IA).

A Tabela 4.2 sintetiza as principais informações da etapa com o uso de IA pelos participantes do estudo. A maioria utilizou o ChatGPT como ferramenta de apoio à programação, exceto P5, que recorreu ao Google Colab, e P4, que contou com a IA do

Repl.it. O domínio prévio da ferramenta era comum entre os participantes, ou seja, todos, com exceção de P4, já haviam utilizado alguma forma de IA anteriormente. No que diz respeito ao tipo de interação com o código, nove participantes revisaram códigos gerados pela IA, enquanto apenas P2 optou por criar um novo código com o suporte do ChatGPT. Os tempos de análise ou correção foram relativamente curtos, variando entre cinco e treze minutos na maioria dos casos, com exceção de P4 e P5, que demandaram 30 e 37 minutos, respectivamente, para acessar e compreender o código gerado. Em termos de acessibilidade, apenas dois participantes (P4 e P5) não conseguiram acessar o código gerado de forma autônoma, enquanto os demais concluíram a tarefa sem necessidade de ajuda externa, o que evidencia o potencial da IA para promover maior autonomia entre estudantes cegos quando integrada a ferramentas adequadas e previamente conhecidas.

Tabela 4.2: Resumo da etapa com uso de IA pelos participantes

Px	**Linguagem de Prog.**	**IA utilizada**	**Já usava IA?**	**Código**	**Tempo (gerar/analisar)**	**Acessou código gerado sem ajuda?**
P1	Java	ChatGPT	Sim	Revisou	Seg /13 min	Sim
P2	Python	ChatGPT	Sim	Criou	Seg / 6 min	Sim
P3	Python	ChatGPT, VS Code	Sim	Revisou	Seg / 5 min	Sim
P4	C++	IA do Repl.it	Não	Revisou	Seg /30 min	Não
P5	Python	Google Colab	Sim	Revisou	Seg /37 min	Não
P6	Python	ChatGPT	Sim	Revisou	Seg /12 min	Sim
P7	Python	ChatGPT	Sim	Revisou	Seg /12 min	Sim
P8	Python	VS Code/ ChatGPT	Sim	Revisou	Seg / 5 min	Sim
P9	PHP	ChatGPT	Sim	Revisou	Seg / 7 min	Sim
P10	Python	ChatGPT	Sim	Revisou	Seg / 5 min	Sim

Durante a resolução dessa etapa, P1 demonstrou autonomia e familiaridade com o uso da IA (ChatGPT), utilizando-a como apoio para revisar, comparar e aprimorar seu próprio código. Ele destacou que **prefere usar a IA em situações em que se sente travado**, mas que, no geral, gosta de tentar resolver os problemas por conta própria antes de recorrer ao recurso. Em relação à acessibilidade, P1 relatou uma experiência positiva. A interação com a IA foi considerada **acessível, especialmente porque os conteúdos gerados eram baseados em texto**, facilmente interpretados pelos leitores de tela. Ele mencionou o uso do NVDA e compartilhou estratégias que emprega quando as sugestões automáticas de código não são lidas corretamente: como alternar do modo de edição (no qual apenas o texto digitado é lido) para o

modo de navegação do leitor de tela, que permite percorrer elementos da interface e identificar as sugestões apresentadas pelo sistema. Também comentou sobre ajustes de configuração que otimizam a leitura dessas sugestões em algumas extensões. P1 ressaltou que, **embora alguns recursos de autocompletar (como os *"ghost texts"* no VS Code) possam apresentar obstáculos, essas barreiras não são da ferramenta de IA em si, mas sim de configurações ou da forma como as extensões estão implementadas**. Ele reforçou que, com os devidos ajustes no leitor de tela, consegue usar a IA com tranquilidade e eficácia. Essa etapa demandou segundos para exibição do código aprimorado; P1 precisou de 13 minutos para analisar o código gerado e comparar com o que havia feito. Para este participante, a segunda etapa do experimento **evidenciou não apenas a redução do tempo necessário para concluir a tarefa com o apoio da IA (em relação à etapa sem IA, que durou cerca de 70 minutos), mas também a viabilidade do uso dessas ferramentas por programadores cegos**, desde que observadas as questões de compatibilidade e configuração dos leitores de tela.

P2 buscou interagir com a ferramenta de IA mesmo com certas limitações de familiaridade técnica no momento, uma vez que não programava há algum tempo. Em termos de acessibilidade, P2 conseguiu navegar pelo navegador (Edge) e utilizar o ambiente necessário para acessar o ChatGPT. Portanto, também com o tempo de execução reduzido e a resolução assistida, o caso de P2 **mostra como ferramentas de IA podem oferecer suporte acessível mesmo a pessoas que estão retornando ou readaptando-se à prática da programação**, desde que estejam inseridas em um ambiente digital acessível e estruturado para o uso com leitores de tela. **O P2 não implementou o código manualmente na primeira etapa. Portanto, nesta etapa com IA, ele se concentrou na interação com a ferramenta para compreender o funcionamento do algoritmo proposto e explorar sua aplicabilidade com apoio da IA**. Foram necessários segundos para a exibição do código e 6 minutos para leitura e interpretação do código gerado pela IA.

P3, usando o ChatGPT e a IA integrada ao Visual Studio Code, **conseguiu detectar e corrigir os erros no código de maneira significativamente mais rápida**. A ferramenta ofereceu sugestões automáticas para corrigir problemas de indentação e remover espaços indevidos no nome da função, além de destacar com precisão a linha onde a variável `n` deveria ser definida. Isso tornou o processo de depuração mais eficiente e direto. Um benefício adicional foi a clareza das mensagens de erro apresentadas pela IA, que, ao contrário do terminal tradicional, onde o participante enfrentava dificuldades para interpretar os erros, forneceu explicações mais detalhadas e acessíveis. **A IA não apenas apontou os problemas de sintaxe e estrutura, mas também sugeriu possíveis soluções, diminuindo consideravelmente a necessidade de tentativa e erro**. Como resultado, o tempo total de depuração foi reduzido drasticamente, passando de 70 minutos, na etapa sem IA, para apenas alguns segundos com o uso da tecnologia, evidenciando o potencial das ferramentas baseadas em IA para apoiar programadores cegos na identificação e correção de erros de forma mais acessível e intuitiva.

Com o uso da IA integrada ao ambiente Repl.it, o P4 conseguiu otimizar significa-

tivamente sua produtividade na resolução da tarefa de programação. A ferramenta **sugeriu automaticamente a inclusão da biblioteca necessária, corrigiu erros de sintaxe e contribuiu para a organização da estrutura do código**. No entanto, apesar desses avanços, **a ausência de recursos de acessibilidade adequados no ambiente de desenvolvimento comprometeu a fluidez do processo**. O P4 enfrentou dificuldades para localizar botões, menus e pop-ups, sendo necessário o uso de instruções detalhadas para navegar utilizando atalhos de teclado e leitores de tela. Além disso, **parte do código gerado pela IA não foi lida pelo leitor de tela**, impedindo a compreensão completa da saída apresentada. O código foi gerado em segundos, mas, para acesso e análise, foram necessários 30 minutos. Esses desafios evidenciam que, **embora as ferramentas de IA representem um avanço significativo para o desenvolvimento de software, sua eficácia junto a programadores cegos ainda depende de melhorias na acessibilidade e usabilidade dos ambientes de programação**, a fim de assegurar uma experiência mais fluida, autônoma e verdadeiramente inclusiva.

P5 utilizou o Google Colab, ambiente que oferece funcionalidades integradas de sugestão e geração automática de código, além da possibilidade de integração com o ChatGPT. Durante a execução, P5 inicialmente colou o código desenvolvido na etapa anterior e solicitou à IA que realizasse uma análise e sugerisse melhorias. **O algoritmo gerado pela IA foi considerado mais eficiente e conciso do que a versão manual**. P5 reconheceu que a IA otimizou o tempo de desenvolvimento e aprimorou a estrutura do código, indicando que, com o uso da ferramenta, a atividade foi resolvida de forma mais rápida e com melhor qualidade. No entanto, **foram identificados problemas sérios de acessibilidade no ambiente**. Apesar de o leitor de tela NVDA conseguir ler os botões de execução e o nome das células no Colab, ele não conseguiu interpretar o conteúdo de código gerado automaticamente pela IA, o que foi descrito pelo participante como um trecho de código "invisível", similar a uma imagem. **Isso impossibilitou que P5 acessasse as sugestões da IA por conta própria, comprometendo a autonomia na leitura e análise das respostas geradas**. O tempo total da atividade com IA, para acesso e análise do código gerado, foi de aproximadamente 37 minutos. Embora mais produtiva em termos de lógica computacional, a etapa com IA **revelou que a ausência de compatibilidade entre o leitor de tela e a interface do Google Colab representa um entrave relevante para a plena inclusão de pessoas cegas**. Em síntese, a IA demonstrou ser uma aliada na construção e otimização de soluções em programação; porém, a eficácia de sua aplicação está diretamente condicionada à acessibilidade do ambiente onde é empregada. O caso de P5 evidencia que o avanço em inteligência artificial deve ser acompanhado de **políticas de design inclusivo**, para garantir que os benefícios da tecnologia sejam efetivamente acessíveis a todos.

P6 utilizou o ChatGPT, ferramenta com a qual já demonstrava familiaridade e que faz parte de sua rotina profissional. **A IA sugeriu um algoritmo com uma estrutura eficiente**, utilizando a função sort() da linguagem Python, e o participante P6 rapidamente reconheceu sua funcionalidade e validou a proposta. Durante a atividade, o participante P6 destacou que **costuma utilizar a IA como ferramenta**

de reforço em situações de bloqueio ou quando deseja confirmar o que já pensou. Ele também mencionou que **costuma adaptar o código sugerido conforme a sua necessidade**, o que demonstra senso crítico e domínio da prática com o apoio da IA. No tocante à **acessibilidade, P6 relatou que não encontra grandes barreiras no uso do ChatGPT**, especialmente por se tratar de uma interface textual compatível com leitores de tela. Ele mencionou que consegue utilizar a ferramenta tanto no navegador quanto via aplicativo móvel, adaptando-se bem aos dois formatos. No entanto, observou que algumas interfaces, como editores de código com sugestões automáticas *("ghost texts")*, ainda apresentam obstáculos, pois os leitores de tela nem sempre interpretam adequadamente esses elementos visuais. Em síntese, a etapa com IA foi concluída em tempo reduzido, cerca de 12 minutos para a análise do código gerado, demonstrando a eficácia da tecnologia no apoio à programação. Além disso, a experiência de P6 evidencia que, **quando acessíveis, as ferramentas baseadas em IA são capazes de ampliar significativamente a produtividade e a autonomia de programadores cegos**.

P7 utilizou o ChatGPT como principal ferramenta de apoio para programar em Python. A principal contribuição da IA foi a **possibilidade de fornecer sugestões de código e identificar prováveis causas de erros**, o que resultou em maior fluidez no processo de desenvolvimento. P7 relatou que, ao encontrar dificuldades para entender os erros apresentados pelo terminal, copiou o código e o submeteu ao ChatGPT, que lhe devolveu uma nova versão funcional. P7 afirmou que esse **apoio o ajudou a compreender que o erro estava relacionado à indentação, algo que antes ele não conseguiria identificar de forma autônoma**. Apesar dos avanços, persistiram limitações importantes. O terminal do VS Code continuou a representar uma barreira significativa de acessibilidade e, mesmo com a intervenção da IA, o participante P7 precisou recorrer a estratégias alternativas, como salvar o código, utilizar o Prompt do Windows e navegar manualmente na interface. Além disso, ele destacou que a IA, embora útil, exige que o usuário saiba "como perguntar" (engenharia de prompt) para obter respostas relevantes, o que impõe um desafio cognitivo adicional. Outro ponto crítico mencionado foi a redução da autonomia lógica. **P7 reconheceu que, se dependesse apenas da IA, tenderia a aceitar soluções prontas, sem refletir profundamente sobre a estrutura do algoritmo**.Embora a IA tenha proporcionado ganhos de produtividade, também suscitou preocupações quanto à superficialidade na aprendizagem e à dependência excessiva do recurso. Essas observações reforçam que, **embora a IA represente um avanço em termos de acessibilidade para programadores cegos, sua adoção requer equilíbrio e criticidade**, devendo ser acompanhada por práticas que incentivem a autonomia e o raciocínio lógico dos usuários.

P8 utilizou recursos de autocompletar habilitados no VS Code, especialmente com atalhos como Tab e Ctrl + ↓ (seta para baixo), indicando familiaridade com assistentes de código integrados ao ambiente de desenvolvimento. P8 reconheceu que **esses recursos auxiliavam na digitação de comandos e estruturas, contribuindo para uma maior agilidade na escrita do código**. Durante a execução da tarefa com suporte de IA (ChatGPT), P8 demonstrou interesse em soluções integradas que eli-

minassem informações visuais irrelevantes, como o caminho do arquivo e o tempo de execução, os quais eram lidos pelo leitor de tela e comprometiam sua experiência. P8 sugeriu que a integração ideal com o VS Code deveria exibir apenas a saída relevante, facilitando a leitura para usuários cegos. Também comentou sobre a configuração do NVDA para emitir sons que indicam a indentação, o que considera essencial para entender a estrutura do código. Esses dados revelam que, embora P8 já utilizasse sugestões automáticas de código antes da atividade, o uso intencional da IA como estratégia de apoio à programação potencializou sua percepção sobre eficiência, acessibilidade e usabilidade. Além disso, a etapa permitiu que P8 **identificasse as melhorias necessárias no ambiente de desenvolvimento para torná-lo mais acessível e funcional**. O código gerado pela IA foi muito semelhante ao que P8 criou, logo, foram necessários menos de 5 minutos para a análise do código.

P9, para aprimorar um código em PHP, utilizou o ChatGPT para revisar e melhorar seu código. A IA sugeriu mudanças que incluíram: **correção de erros sintáticos, como a troca de letras maiúsculas/minúsculas em nomes de variáveis; substituição de trechos de código por alternativas mais eficientes, como a troca direta de elementos em listas sem criar variáveis temporárias e sugestão de uso de função pronta de ordenação** (porém não adotada no experimento para evitar atalhos). Além disso, P9 mencionou que também utiliza eventualmente o Gemini[2], IA da Google, mas com menos frequência, pois identificava mais "alucinações" (respostas imprecisas ou inventadas) do que no ChatGPT. A interação com a IA foi descrita como positiva, **especialmente por sua capacidade de sugerir melhorias e apontar erros de forma acessível**. Ainda assim, P9 reforçou que prefere utilizar a IA apenas como último recurso, **para evitar dependência e desenvolver sua própria lógica de programação**. A leitura dos códigos gerados foi feita sem problemas com o leitor de tela NVDA, e os principais desafios relatados envolveram a lógica do problema, e não a interação com a IA em si. O tempo gasto, sobretudo para a análise do código gerado, foi decerca de 7 minutos.

P10 utilizou o ChatGPT para revisar e aprimorar o código escrito anteriormente em Python. Após copiar e colar seu código na interface da IA, o participante recebeu como resposta uma versão otimizada do algoritmo de ordenação, **com melhorias que incluíam a verificação das trocas dos valores, estratégia típica do `Bubble Sort` eficiente. Inicialmente, a IA sugeriu o uso da função `sort()`, no entanto, a pedido da pesquisadora, foi gerado um novo código sem o uso de funções prontas**. P10 destacou que a explicação da IA foi clara e que conseguiu compreender a lógica por trás da proposta, especialmente os ganhos de desempenho ao evitar comparações desnecessárias em listas já ordenadas. **A tarefa foi concluída com maior fluidez e autonomia, demonstrando o potencial da IA como recurso de apoio acessível no ensino de programação** para estudantes cegos. Essa etapa durou apenas 5 minutos.

[2]https://gemini.google.com/

Síntese
Na atividade com uso de IA, os participantes resolveram o mesmo problema de ordenação crescente da etapa anterior, agora com apoio de ferramentas de IA escolhidas livremente (majoritariamente ChatGPT, além da IA do Repl.it e do Google Colab). O objetivo foi investigar a integração entre leitores de tela e essas ferramentas, considerando compreensão de código, confiabilidade, clareza das sugestões, acessibilidade e necessidade de ajuda externa. Em geral, os participantes já tinham experiência prévia com IA e, na etapa com suporte, ou revisaram o código manual da primeira fase ou geraram um código novo para comparar. Os tempos para análise/correção do código gerado pela IA foram, em sua maioria, baixos (entre 5 e 13 minutos), com exceção de P4 e P5, que precisaram de 30 e 37 minutos devido a dificuldades de acessibilidade nas interfaces Repl.it e Google Colab. Em termos de acesso autônomo ao código, apenas esses dois participantes não conseguiram ler as respostas sem ajuda externa, enquanto os demais relataram boa integração entre IA e leitor de tela.
Os relatos individuais mostram um padrão consistente de ganho de produtividade e acessibilidade, mas com nuances importantes. Participantes como P1, P3, P6, P9 e P10 usaram a IA sobretudo para revisar, explicar erros e sugerir melhorias, reduzindo drasticamente o tempo de depuração (por exemplo, de cerca de 70 minutos para poucos minutos ou segundos) e valorizando a clareza textual das explicações. P2, que estava há algum tempo sem programar, ilustra como a IA pode apoiar quem está se readaptando, desde que o ambiente digital (navegador, leitor de tela) seja acessível. Já P4 e P5 evidenciam que, mesmo com códigos melhores e mais eficientes gerados em segundos, a falta de compatibilidade plena entre IA e interface (código "invisível" para o leitor de tela, elementos gráficos não lidos) compromete a autonomia. Por fim, o conjunto dos resultados indica que a IA funciona tanto como mediadora explicativa (quando ajuda a compreender e corrigir o próprio código) quanto como possível substituta da prática de programação (quando gera o código completo), o que exige cautela para que o ganho de tempo não venha à custa da aprendizagem e da autonomia dos estudantes cegos.

Comparações sobre as Etapas Sem e Com IA

A Figura 4.11 apresenta um gráfico de barras empilhadas que compara o tempo gasto pelos participantes nas duas etapas do experimento: sem o uso de IA e com o uso de IA. Para cada participante, foram considerados dois componentes de tempo em cada etapa: o tempo para elaborar o código e o tempo para corrigir os erros encontrados.

Na etapa sem IA (representada pelas barras em tons de azul), observa-se que o tempo total de execução variou amplamente entre os participantes, com destaque para o participante P4, que levou 85 minutos para escrever e 90 minutos para corrigir o

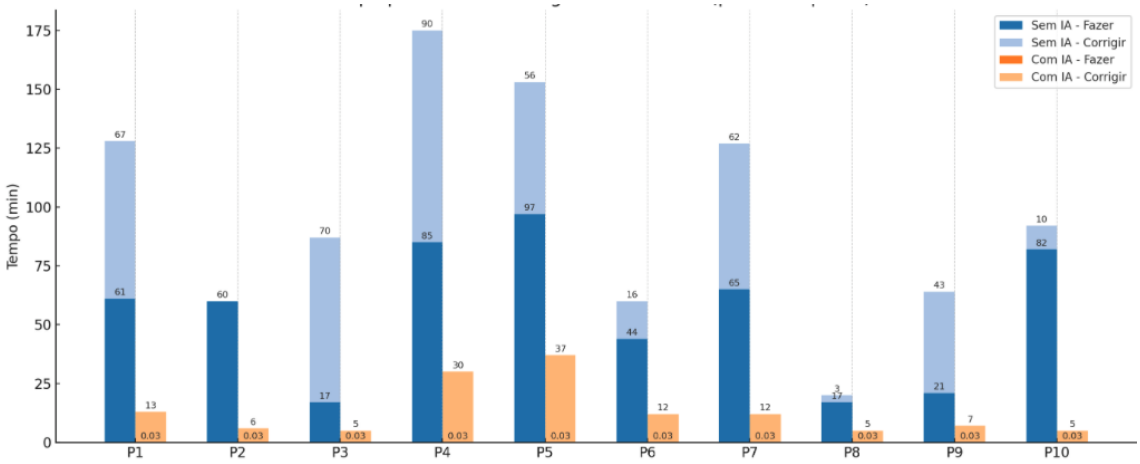

Figura 4.11: Comparação de tempo de execução com e sem IA

código, totalizando 175 minutos, e P5, com 153 minutos. Já o participante P2 abandonou a etapa de correção após os 60 minutos iniciais de desenvolvimento. Em média, a etapa sem IA foi marcada por longos períodos de depuração e dificuldades técnicas, como erros de lógica, sintaxe e uso de estruturas de repetição, conforme descrito anteriormente.

Na etapa com IA (barras em tons de laranja), o tempo de desenvolvimento inicial foi significativamente reduzido, com todos os participantes levando apenas cerca de 2 segundos (aproximadamente 0,03 minutos) para exibir o código (gerado ou revisado) - tempo de implementação, sugerido automaticamente pelas ferramentas de IA, como o ChatGPT. O tempo de análise da resposta da IA, que chamamos de tempo para corrigir, foi consideravelmente menor para a maioria dos participantes, com destaque para os ganhos de produtividade obtidos pelos participantes P1, P3, P6 e P10. Contudo, os participantes P4 e P5 destoaram desse padrão, apresentando 30 e 37 minutos, respectivamente, apenas na fase de análise do código. Essa diferença se deve a problemas de acessibilidade: ambos enfrentaram dificuldades para acessar as respostas fornecidas pela IA por meio dos leitores de tela e precisaram de ajuda externa para interagir com os conteúdos sugeridos, o que ampliou significativamente o tempo total de execução.

Esse gráfico evidencia de forma clara o impacto positivo das ferramentas de IA na redução do tempo total de execução, principalmente na etapa de correção, mantendo a tarefa acessível e mais eficiente para programadores cegos.

Observou-se uma redução expressiva no tempo de depuração com o uso da IA, sobretudo porque as ferramentas foram capazes de traduzir e explicar mensagens de erro que, sem o apoio tecnológico, não eram compreendidas pelos participantes. No entanto, esse ganho temporal não implica, necessariamente, em um ganho de aprendizagem ou de autonomia. Embora a IA tenha auxiliado na interpretação dos erros, em alguns casos ela chegou a gerar o código completo, o que desloca o papel do par-

ticipante de construtor para verificador do código. Assim, o processo tornou-se mais eficiente, mas menos ativo do ponto de vista cognitivo. Para parte dos participantes, a IA funcionou como mediadora explicativa, promovendo maior compreensão e acessibilidade, enquanto, para outros, representou um agente substitutivo da prática de programação. Portanto, o ganho de tempo deve ser interpretado como um avanço em acessibilidade e eficiência, mas não necessariamente como um indicador de aprendizagem significativa ou de desenvolvimento de autonomia.

4.1.4 Análise de Conteúdo

A partir da análise de conteúdo, realizada segundo os procedimentos de Bardin (2011), emergiram categorias que refletem as percepções e experiências dos participantes em relação ao uso da Inteligência Artificial em ambientes de desenvolvimento integrados. Essas categorias sintetizam as principais barreiras, potencialidades e implicações educacionais identificadas durante o experimento e nas entrevistas realizadas após as atividades práticas.

De modo geral, os resultados qualitativos revelaram que a IA contribuiu para a ampliação da autonomia dos participantes e para a redução do tempo de depuração, favorecendo uma aprendizagem mais fluida e produtiva. No entanto, persistem desafios relacionados à acessibilidade plena das ferramentas utilizadas, principalmente devido à limitação de integração entre leitores de tela e os recursos interativos das IDEs, o que impactou a navegação e a interpretação das sugestões de código.

A interpretação das categorias indica que a adoção de ferramentas de IA pode representar um avanço significativo na inclusão digital de estudantes cegos, desde que acompanhada de ajustes de acessibilidade e estratégias educacionais que favoreçam o uso efetivo dessas tecnologias. Os resultados obtidos fundamentam a discussão apresentada no Capítulo 5, na qual são articulados os aspectos pedagógicos, tecnológicos e éticos decorrentes do uso da IA no ensino de programação.

Análise por Categoria e Unidades de Registro (UR)

Categoria 1: Acessibilidade em Ambientes de Programação

As dificuldades relacionadas à acessibilidade de IDEs e editores de código foram mencionadas por todos os 10 participantes. Os relatos destacam incompatibilidades com leitores de tela, falta de retorno sonoro adequado, menus não acessíveis e limitações na navegação por teclado. O VS Code foi frequentemente citado como a ferramenta mais compatível, enquanto outras, como Eclipse e Colab, foram apontadas como problemáticas. As falas a seguir retratam a percepção dos estudantes cegos em relação à esta categoria.

- *"Uso o VS Code porque consigo navegar com atalhos e ele respeita as estruturas do código. Outros ambientes não me deixam nem saber onde estou." (Participante 1)*

- *"Muitas vezes, é impossível ajustar o zoom ou o contraste da interface. Além disso, a navegação por teclado falha em algumas janelas." (Participante 2)*
- *"O Colab é péssimo com leitores. Tive que mudar de ambiente porque ele travava tudo." (Participante 3)*
- *"O VS Code, acho completo e acessível. Ele funciona muito bem com o NVDA." (Participante 4)*
- *"Tentei usar IDEs com Inteligência Artificial integrada, mas as caixas de diálogo não eram lidas pelo NVDA." (Participante 5)*
- *"No ambiente acadêmico, o laboratório usa ferramentas que não têm acessibilidade. Precisei levar meu notebook com o VS Code instalado." (Participante 6)*
- *"Têm muitos menus com pop-ups escondidos, e o leitor não consegue identificar o que está sendo exibido. Isso atrapalha demais." (Participante 7)*
- *"O leitor de tela não interage com os avisos de erro ou com a estrutura visual das IDEs, então é como andar no escuro." (Participante 8)*
- *"A acessibilidade depende muito do ambiente. Se a ferramenta não tem suporte a atalhos ou à leitura das barras laterais, não consigo programar." (Participante 9)*
- *"Já tentei usar o Eclipse, mas é bem confuso; ele tem muitos botões que o leitor não lê, e as opções não são muito intuitivas." (Participante 10)"*

Categoria 2: Desafios Técnicos na Programação

Os desafios técnicos relacionados à prática da programação, como erros de sintaxe, estruturação lógica, indentação e compreensão de mensagens de erro, foram mencionados por 9 dos 10 participantes. A ausência de retorno visual torna esses desafios ainda mais complexos, especialmente quando o leitor de tela falha em sinalizar corretamente os problemas ou quando os erros são sutis e dependem de detalhes visuais, como pontuação ou espaçamento. As falas abaixo trazem a percepção dos estudantes cegos em relação a esta categoria.

- *"Uso muito de tentativa e erro. Faço o código, executo, e se não rodar, vou alterando até funcionar." (Participante 1)*
- *"A sintaxe, alguns erros em linguagem case sensitive, erros de formatação... como duas vírgulas no lugar de uma." (Participante 2)*
- *"Os erros de indentação em Python são os que mais me atrapalham. Um espaço a mais e tudo para de funcionar." (Participante 3)*
- *"Minha maior dificuldade é aprender com conteúdos em vídeo sem que o código seja disponibilizado." (Participante 4)*

- *"Tem código que só consigo resolver se alguém me disser o que está aparecendo visualmente. O leitor não ajuda nesses casos." (Participante 5)*
- *"Tenho dificuldade em saber por onde começar... O maior desafio é saber quando aparece um erro no VS Code... para o leitor de tela, é como se não tivesse nada." (Participante 6)*
- *"Erro de digitação ou comando mal escrito derruba tudo, e o NVDA não mostra isso de forma clara." (Participante 7)*
- *"A depuração é o ponto mais crítico. Sem saber o que está errado, você fica perdido." (Participante 8)*
- *"Depurar sem feedback visual é muito difícil. Às vezes, sei que tem erro porque o programa não roda, mas não sei onde está o problema." (Participante 9)*

Categoria 3: Recursos Educacionais e Inclusão

As dificuldades relacionadas aos materiais didáticos, avaliações e práticas pedagógicas inclusivas foram mencionadas por 8 dos 10 participantes. Os relatos revelam que, embora alguns professores se esforcem para adaptar os conteúdos, a maioria dos materiais ainda é distribuída em formatos visuais inacessíveis (e.g. PDFs com imagens sem descrição, vídeos sem transcrição, ausência de versões em texto). Além disso, foi apontada a falta de preparo pedagógico e técnico por parte de muitos docentes e a ausência de suporte efetivo por núcleos de acessibilidade nas instituições. As falas seguintes relatam as perspectivas desta categoria.

- *"Em geral, é o mesmo material dos colegas, só que sem imagens ou gráficos. E isso prejudica muito o meu aprendizado." (Participante 1)*
- *"Hoje está melhor por conta dos documentos em PDF e HTML com texto, mas antes recebia arquivos cheios de imagens sem descrição." (Participante 2)*
- *"Falta material acessível, adequado e de instrução mesmo. O professor precisa entender como adaptar e como ensinar." (Participante 3)*
- *"As avaliações raramente são adaptadas. Muitas vezes, recebo as mesmas questões que os demais, mas em um formato que não consigo ler." (Participante 4)*
- *"Nem sempre os materiais são adaptados. Às vezes, tenho que recorrer a gambiarras para conseguir entender." (Participante 5)*
- *"Falta preparo por parte dos professores e também sensibilidade para entender que a acessibilidade não é favor, é direito." (Participante 6)*
- *"A universidade não está preparada para nos manter aqui. Não há estrutura nem acompanhamento especializado para nos ajudar." (Participante 7)*
- *"Recebo materiais com imagens e gráficos que não têm descrição. Às vezes, perco completamente o conteúdo." (Participante 8)*

Categoria 4: Uso e Limites da Inteligência Artificial

O uso de ferramentas de Inteligência Artificial (IA) no apoio à programação foi mencionado por 8 dos 10 participantes. Muitos relataram experiências positivas ao utilizar IAs como ChatGPT, Replit AI e Cursor para sugerir ou corrigir códigos. No entanto, também apontaram limitações relacionadas à falta de integração com leitores de tela, dificuldade em interpretar sugestões de código geradas automaticamente e ausência de explicações detalhadas. A ausência de acessibilidade nas interfaces e na documentação das IAs utilizadas também foi recorrente nas falas dos participantes, como pode ser observado abaixo.

- *"Gostaria de uma IA pensada para cegos, que descrevesse o que está fazendo no código e avisasse quando algo não foi lido." (Participante 1)*
- *"Usei IA só por curiosidade. O problema é que as respostas são genéricas demais. Não ajuda muito sem contexto." (Participante 2)*
- *"A IA precisa pensar como a gente. Não adianta só completar código, tem que explicar e ser acessível com leitor de tela." (Participante 3)*
- *"Uso o ChatGPT para resolver alguns problemas de código, mas ele nem sempre entende o que eu quero. Às vezes, preciso reescrever várias vezes." (Participante 4)*
- *"Já testei IA no Replit, ela completou um código para mim, mas não consegui entender o que ela fez porque o leitor não lia a resposta completa." (Participante 5)*
- *"Quando a IA sugere algo e você não sabe se está certo ou errado, fica dependente. Precisamos de algo mais interativo." (Participante 6)*
- *"O que falta é uma IA que se integre com os leitores, que funcione como um suporte real para navegar e entender o código." (Participante 7)*
- *"O Cursor é útil, mas a janela de sugestões nem sempre é lida pelo NVDA." (Participante 9)*

Categoria 5: Estratégias Individuais de Superação

Apesar das diversas barreiras enfrentadas, 7 dos 10 participantes relataram adotar estratégias pessoais para superar os desafios na programação. As soluções envolvem desde a busca autônoma por tutoriais e fóruns até o uso de tentativa e erro, apoio de colegas, adaptação do ambiente de estudo e dependência de ferramentas familiares como o VS Code. Essas estratégias revelam um alto grau de resiliência e autodidatismo por parte dos participantes, como relatado abaixo, mesmo diante de estruturas educacionais e tecnológicas frequentemente excludentes.

- *"Tento bastante antes de pedir ajuda. Faço o código várias vezes até funcionar." (Participante 1)*

- *"Busco muito no YouTube, especialmente canais que explicam passo a passo." (Participante 3)*
- *"Utilizo apenas meu conhecimento prévio e tento lembrar das estruturas que já vi antes." (Participante 4)*
- *"Uso fóruns como Stack Overflow, mesmo que nem sempre consiga entender tudo." (Participante 5)*
- *"Uso métodos alternativos, como copiar exemplos e adaptar, porque nem sempre consigo construir tudo do zero." (Participante 6)*
- *"Peço ajuda aos colegas quando não consigo entender um código ou depurar." (Participante 7)*
- *"Me adaptei ao VS Code porque já sei os atalhos e como ele funciona. Uso sempre a mesma configuração." (Participante 9)*

As categorias revelam que, embora haja avanços na acessibilidade de ferramentas de programação, persistem desafios significativos tanto no âmbito técnico quanto educacional. A inteligência artificial aparece como uma aliada promissora, desde que suas soluções sejam desenvolvidas com base em princípios de acessibilidade digital e sob orientação pedagógica adequada.

4.1.5 Frequência das Categorias e Subcategorias

A Figura 4.12 apresenta um gráfico de barras que ilustra a frequência de menções às subcategorias temáticas identificadas durante a análise de conteúdo, organizadas por categoria principal e diferenciadas por cores para facilitar a visualização. Os dados foram extraídos de **dois tipos de coleta qualitativa**:

- **entrevistas semiestruturadas**, (Apêndice C), realizadas com os participantes;
- **respostas às questões fechadas e abertas do formulário**, (Apêndice D), respondidas pelos 10 participantes cegos que estudam e atuam na área de programação.

Cada barra representa o número de participantes que mencionaram uma determinada subcategoria em seus relatos.

Vale destacar que o total de participantes é $n = 10$. As barras do gráfico não representam indivíduos distintos, mas sim a frequência de menções. Por isso, há sobreposição entre as barras: um mesmo participante pode ter citado mais de uma subcategoria. Portanto, os valores expressos não somam 10, mas representam a quantidade de menções individuais por subtema, reforçando a diversidade e a riqueza das experiências relatadas. O uso de cores permite observar, de forma comparativa, a distribuição das subcategorias entre as grandes áreas de análise, como Tecnologia Assistiva, Barreiras Educacionais, Ambientes de Desenvolvimento e Inteligência Artificial.

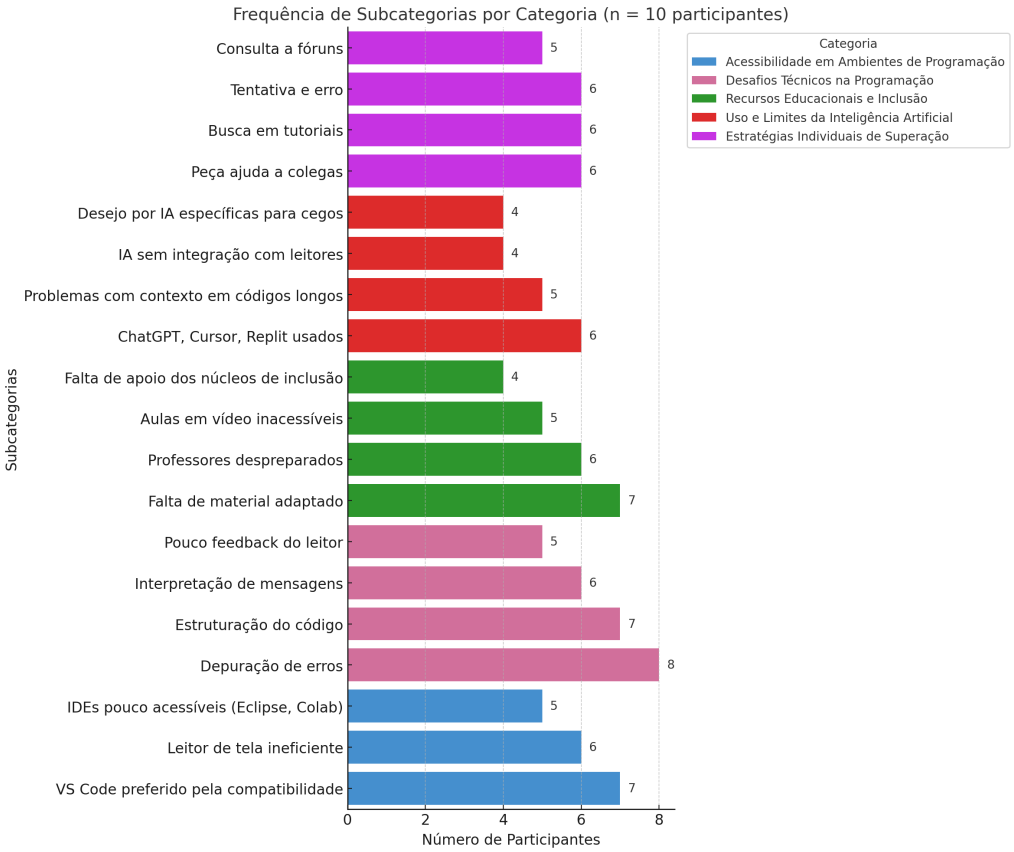

Figura 4.12: Frequência de Subcategorias por Categoria

Dentre as subcategorias com maior número de menções, destacam-se “Depuração de erros” (8 participantes) e “VS Code preferido pela compatibilidade” (7 participantes), pertencentes, respectivamente, às categorias Desafios Técnicos na Programação e Acessibilidade em Ambientes de Programação. Tais achados evidenciam que, apesar das limitações enfrentadas, os participantes buscam soluções práticas e se adaptam a ferramentas mais acessíveis para realizarem suas atividades de programação.

Outras subcategorias com alta frequência incluem “Falta de material adaptado” (7), “Estruturação do código” (7), e “ChatGPT, Cursor, Replit usados” (6), indicando que tanto as barreiras educacionais quanto o uso emergente de ferramentas de Inteligência Artificial ocupam papel relevante na experiência dos participantes. Já subcategorias como “Desejo por IA específicas para cegos” e “Falta de apoio dos núcleos de inclusão”, mencionadas por quatro participantes, embora menos frequentes, revelam lacunas importantes relacionadas à personalização tecnológica e ao suporte institucional.

O gráfico também mostra que diversas estratégias de superação individuais, como “Tentativa e erro”, “Busca em tutoriais”, e “Pedido de ajuda a colegas”, foram relatadas por mais da metade dos participantes, evidenciando um esforço contínuo por autonomia frente aos desafios enfrentados.

Portanto, a visualização da frequência das subcategorias reforça a presença de padrões recorrentes nas falas dos participantes, ao mesmo tempo em que revela pontos críticos de atenção para práticas pedagógicas inclusivas e o desenvolvimento de recursos de tecnologia assistiva mais eficazes.

4.2 Recomendações Técnicas e Pedagógicas

Os dados coletados por meio das entrevistas e questionários evidenciaram diversos desafios enfrentados por pessoas com deficiência visual no processo de aprendizagem em programação. Com base nessas evidências empíricas, é possível extrair recomendações alinhadas ao objetivo de favorecer ambientes mais inclusivos e eficazes.

As recomendações técnicas derivam diretamente das barreiras enfrentadas pelos participantes ao interagir com IDEs e ferramentas de IA, especialmente no que se refere à compatibilidade com leitores de tela, à organização da interface e à clareza das mensagens de erro. Já as recomendações pedagógicas emergiram das experiências relatadas no contexto educacional, incluindo dificuldades com materiais inacessíveis, ausência de mediação especializada e inadequações metodológicas.

Recomendações técnicas:

Do ponto de vista técnico, os participantes destacaram a importância da adoção de ferramentas que sejam compatíveis com leitores de tela, como o VS Code, frequentemente citado como o editor mais acessível. Por outro lado, IDEs como o Eclipse e plataformas como o Google Colab (cujas áreas de resposta não são lidas por leitores de tela) foram apontadas como pouco acessíveis, sugerindo a necessidade de critérios de acessibilidade digital na seleção e uso de ambientes de desenvolvimento.

Além disso, os relatos sobre dificuldades na leitura de mensagens de erro indicam a urgência de ferramentas de depuração acessíveis, com saídas textuais claras e suporte a atalhos de teclado. Oferecer *feedbacks* sonoro e textual personalizáveis, pois a inclusão de sinais sonoros ou resumos textuais sobre erros e sugestões de código pode auxiliar usuários cegos a compreenderem rapidamente o que precisa ser ajustado no código. Outro ponto é permitir a reativação das sugestões de código, no caso do VS Code, visto que os participantes relataram que as sugestões automáticas de código (*ghost text*) eram lidas apenas uma vez, exigindo ações artificiais para reaparecer, como deletar parte do texto e voltar a escrevê-lo. Portanto, as ferramentas devem oferecer formas acessíveis de revisão dessas sugestões.

No campo das tecnologias emergentes, o uso de ferramentas de Inteligência Artificial, como ChatGPT e Cursor, mostrou-se promissor, mas ainda limitado em termos de acessibilidade. Participantes relataram falhas de integração com leitores de tela e dificuldades em interpretar respostas automatizadas. Assim, recomenda-se o desenvolvimento de IAs assistivas que não apenas forneçam código, mas que sejam capazes de explicar, estruturar e interagir com o usuário por meio de *feedbacks* auditivos claros e navegáveis.

Síntese:

- Utilizar editores e ambientes de desenvolvimento compatíveis com leitores de tela, especialmente o VS Code.
- Evitar ou adaptar ferramentas pouco acessíveis, como Eclipse e Google Colab.
- Adotar critérios de acessibilidade digital na seleção de IDEs.
- Implementar depuradores acessíveis, com mensagens de erro textuais e claras.
- Garantir suporte a atalhos de teclado na navegação e depuração.
- Oferecer feedback sonoro e textual personalizável para identificação de erros e sugestões.
- Disponibilizar formas acessíveis de revisar e reativar sugestões automáticas de código.
- Melhorar a integração de ferramentas de IA com leitores de tela.
- Desenvolver IAs capazes de explicar e estruturar código de forma clara e navegável.
- Assegurar que o código gerado por IAs seja totalmente interpretável por leitores de tela.

Recomendações pedagógicas:

Do ponto de vista pedagógico, os resultados apontam para uma carência de materiais didáticos adaptados e de estratégias inclusivas por parte dos docentes. A presença de aulas com vídeos, materiais com imagens sem descrição textual e avaliações não adaptadas revela a necessidade urgente de formação docente em acessibilidade educacional digital. Recomenda-se a adoção de recursos multimodais (texto alternativo, transcrição de vídeos, documentos acessíveis) e práticas avaliativas flexíveis, que respeitem as especificidades sensoriais dos estudantes.

As avaliações e exercícios práticos devem estar disponíveis em formatos compatíveis com recursos de tecnologia assistiva (texto, HTML acessível, LaTeX com estrutura semântica, etc.), evitando o uso exclusivo de imagens para apresentar trechos de código. Relatos apontaram ausência de apoio especializado em cursos EAD, o que dificultou o aprendizado autônomo.

Outro ponto é a utilização orientada de IA como ferramenta didática. O uso de IA como o ChatGPT mostrou-se eficaz na identificação e correção de erros, mas precisa ser inserido pedagogicamente de forma crítica, para evitar dependência excessiva e estimular a compreensão dos conceitos de programação.

Por fim, destaca-se que a maioria dos participantes recorre a estratégias individuais de superação, como tentativa e erro, tutoriais e ajuda de colegas, o que evidencia uma lacuna no suporte institucional. Isso reforça a importância de estruturas de apoio permanentes, como núcleos de acessibilidade atuantes, tutoriais acessíveis, monitorias especializadas e espaços de escuta ativa nas instituições de ensino.

Essas recomendações foram construídas a partir das vivências de dez participan-

tes com deficiência visual, com perfis variados, e refletem tanto suas dificuldades quanto as estratégias utilizadas para superá-las. Assim, espera-se que tais orientações possam auxiliar na reformulação de práticas educacionais e no desenvolvimento de tecnologias mais acessíveis, contribuindo para um ensino de programação verdadeiramente inclusivo.

Síntese:

- Produzir materiais didáticos acessíveis e adaptados, incluindo descrição textual de imagens e vídeos com transcrição.
- Adotar recursos multimodais, como texto alternativo, documentos acessíveis e vídeos com legendas e audiodescrição.
- Implementar práticas avaliativas flexíveis, respeitando as necessidades sensoriais dos estudantes.
- Disponibilizar exercícios e avaliações em formatos compatíveis com recursos de tecnologia assistiva (texto, HTML acessível, LaTeX semântico).
- Evitar o uso exclusivo de imagens para apresentar códigos ou conteúdos essenciais.
- Ampliar a formação docente em acessibilidade educacional digital.
- Oferecer apoio especializado e contínuo aos estudantes, especialmente em cursos EAD.
- Utilizar IA como ferramenta didática de forma orientada e crítica, evitando dependência excessiva.
- Estimular o desenvolvimento da compreensão conceitual de programação, mesmo com uso de IA.
- Criar estruturas institucionais de suporte, como núcleos de acessibilidade, monitorias especializadas e tutoriais acessíveis.
- Fomentar espaços de escuta ativa para acompanhar dificuldades permanentes dos estudantes cegos.

Capítulo 5

Discussão

A presente discussão é conduzida à luz do objetivo central da pesquisa: avaliar a acessibilidade e a usabilidade de ferramentas de Inteligência Artificial e Ambientes de Desenvolvimento Integrado no ensino de programação para estudantes cegos, identificando desafios, estratégias e percepções relacionadas à autonomia e à integração com recursos de tecnologia assistiva. Neste capítulo, os dados empíricos oriundos de entrevista, questionário e experimentos práticos são interpretados criticamente, em diálogo com a literatura científica revisada ao longo do trabalho. A investigação foi orientada por um conjunto de hipóteses, ver Capítulo 1, que abordam, por exemplo, o potencial da IA para melhorar a navegabilidade e a produtividade desses estudantes (H1, H2), bem como as limitações persistentes relacionadas à usabilidade das ferramentas e à integração com leitores de tela (H3). Além disso, foram consideradas variáveis como o nível de familiaridade dos estudantes com programação e a comparação entre as etapas com e sem uso de IA (H4 e H5), compondo um arcabouço interpretativo para a análise.

A escolha metodológica de um estudo de caso com múltiplas unidades de análise permitiu examinar em profundidade a experiência de estudantes com deficiência visual em sua formação acadêmica em Computação, respondendo a uma lacuna identificada na literatura, que frequentemente se concentra nas barreiras enfrentadas no mercado de trabalho. Ao voltar-se para a trajetória formativa, esta pesquisa busca compreender como se manifestam as barreiras e os apoios durante o processo de aprendizagem.

A análise qualitativa dos dados coletados foi organizada com base na técnica de Análise de Conteúdo (Bardin, 2011), resultando em categorias temáticas que estruturam a presente discussão:

(i) Acessibilidade em ambientes de programação;

(ii) Desafios técnicos na Programação;

(iii) Recursos Educacionais e Inclusão;

(iv) Uso e Limites da Inteligência Artificial;

(v) Estratégias Individuais de Superação.

A adoção dessas categorias permitiu uma leitura sistemática dos dados e favoreceu o cruzamento entre os achados empíricos e os principais conceitos trabalhados na fundamentação teórica, como acessibilidade digital, recursos de tecnologia assistiva e inclusão no ensino de programação.

Ao articular os resultados com os referenciais teóricos e as hipóteses formuladas, esta seção visa aprofundar a compreensão sobre os caminhos e limites da acessibilidade no ensino de programação para pessoas com deficiência visual, destacando possibilidades de intervenção pedagógica e tecnológica com vistas a uma formação mais equitativa, autônoma e eficiente.

5.1 Acessibilidade em Ambientes de Programação

A análise dos relatos dos participantes revelou uma disparidade significativa na acessibilidade dos IDEs utilizados no processo de aprendizagem em programação. A maioria dos estudantes demonstrou preferência pelo **Visual Studio Code** (VS Code), apontado como mais responsivo aos leitores de tela, com melhor estrutura de navegação e recursos de autocompletar acessíveis. Em contraste, ferramentas como o **Eclipse** foram frequentemente descritas como confusas, com estruturas de menus e interfaces que dificultam a interação por meio de recursos de tecnologia assistiva, como o NVDA ou o JAWS. Essas percepções se alinham às discussões teóricas sobre os desafios da acessibilidade digital em contextos técnicos. Eckhardt et al. (2019), ao analisarem recursos de tecnologia assistiva em ambientes computacionais, destacam que a ausência de estrutura semântica adequada e o uso de elementos gráficos não rotulados são barreiras frequentes enfrentadas por programadores cegos. Da mesma forma, Wilkens et al. (2021) alertam para a importância do design inclusivo nas IDEs, enfatizando que muitos desses ambientes foram originalmente concebidos para usuários com visão, resultando em interfaces pouco adaptadas ao uso com leitores de tela.

Apesar dos avanços em ferramentas de tecnologia assistiva, como leitores de tela e sintetizadores de voz, **persistem limitações importantes**. Muitos leitores não conseguem interpretar corretamente os elementos visuais complexos das IDEs, como árvores de diretórios, *tooltips*[1] (moldura flutuante quando se passa o mouse), janelas de depuração ou mensagens de erro destacadas por cores. Essas falhas afetam diretamente a autonomia dos estudantes e a fluidez do processo de codificação, exigindo, muitas vezes, estratégias compensatórias, como copiar o código para editores de texto mais simples ou recorrer ao auxílio externo.

Além disso, a falta de padronização na acessibilidade entre os ambientes de desenvolvimento gera insegurança no aprendizado. O estudante precisa aprender não

[1] https://getbootstrap.com.br/docs/4.1/components/tooltips/

apenas a lógica da programação, mas também desenvolver competências de navegação adaptativa diante de ferramentas inconsistentes. Nesse contexto, observa-se que **a acessibilidade não pode ser tratada como um complemento**, mas como uma característica fundamental na concepção de qualquer recurso educacional digital voltado à programação.

À luz desses resultados, reforça-se a necessidade de ampliar o compromisso das comunidades de desenvolvimento de software com os princípios de acessibilidade desde o design (*"born accessible"*)[2] e de criar padrões técnicos que promovam a compatibilidade plena com recursos de tecnologia assistiva. Ao negligenciar esse aspecto, ferramentas tradicionais de TA digitais tornam-se insuficientes para garantir equidade no ensino de programação, limitando as oportunidades de formação plena e autônoma para estudantes cegos.

Síntese da Subseção

- Há grande disparidade na acessibilidade das IDEs utilizadas por estudantes cegos.
- O VS Code foi apontado como o ambiente mais acessível, responsivo ao leitor de tela e com melhor estrutura de navegação.
- Eclipse e outras IDEs foram percebidas como confusas e pouco acessíveis, com menus complexos e elementos não rotulados.
- Leitores de tela têm dificuldades para interpretar elementos visuais complexos, como árvores de diretórios, *tooltips*, janelas de depuração e mensagens de erro coloridas.
- Essas limitações comprometem a autonomia, exigindo estratégias compensatórias, como usar editores mais simples ou recorrer a ajuda externa.
- A falta de padronização entre ambientes de desenvolvimento gera insegurança e demanda habilidades adicionais de navegação adaptativa.
- A acessibilidade deve ser tratada como característica central, e não opcional, no design de ferramentas educacionais para programação.
- É necessária a adoção de princípios de design inclusivo (*"born accessible"*) e padrões técnicos que garantam compatibilidade com tecnologias assistivas.

5.2 Desafios Técnicos e Estratégias de Programação

A experiência dos participantes ao lidarem com ambientes de programação revelou uma série de desafios técnicos recorrentes, especialmente relacionados à sintaxe, lógica de programação, depuração (*debug*) e limitações das IDEs utilizadas. Muitos relataram dificuldade em interpretar mensagens de erro que, por vezes, não são lidas corretamente pelos leitores de tela. Problemas de sintaxe simples, como o uso incorreto de delimitadores, letras maiúsculas e minúsculas ou o fechamento de blocos

[2] https://bornaccessible.benetech.org

de código, foram apontados como causas frequentes de frustração e tempo excessivo em tarefas que, para videntes, seriam mais simples.

O processo de depuração mostrou-se particularmente complexo. Por não contar com recursos visuais como destaques de linha ou marcações de erro em tempo real, os participantes relataram a necessidade de reexecutar manualmente trechos do código ou utilizar saídas (*prints*) para tentar localizar os problemas. Como apontado por alguns, esse processo exigia uma compreensão detalhada da estrutura do código e uma memória de trabalho apurada, o que demandava um maior esforço cognitivo.

Diante desses desafios, os participantes desenvolveram estratégias adaptativas variadas. Uma das mais recorrentes foi o uso de recursos externos, como fóruns de discussão, *blogs* e consultas à documentação oficial das linguagens. Outros mencionaram a estratégia de tentativa e erro, reescrevendo partes do código ou testando diferentes abordagens até encontrar uma solução funcional. A ajuda de colegas, professores ou grupos de apoio também se destacou como uma prática comum, especialmente em momentos em que as barreiras tecnológicas se tornavam intransponíveis de forma individual.

Essas práticas estão alinhadas com os pressupostos da aprendizagem construcionista, que enfatiza a participação ativa do estudante na construção do conhecimento, mesmo diante de dificuldades. Como discute Papert (1986), o "aprender fazendo" é essencial na formação em computação, especialmente quando os alunos têm liberdade para experimentar, errar e corrigir seus erros com base em *feedbacks* e observações. No caso dos estudantes cegos, essa abordagem exige também um ambiente acessível e um suporte pedagógico que reconheça suas especificidades.

Por fim, essas estratégias também refletem um importante grau de autonomia e protagonismo na resolução de problemas. Apesar das barreiras estruturais, os participantes demonstraram iniciativa para contornar os desafios impostos pelos recursos técnicos limitados. Isso reforça a importância de criar ambientes de aprendizagem que não apenas removam barreiras, mas também promovam condições para o desenvolvimento da autonomia na programação.

Síntese da Subseção

- Participantes relataram desafios recorrentes relacionados à sintaxe, lógica de programação, depuração e limitações das IDEs.
- Mensagens de erro muitas vezes não eram lidas corretamente pelos leitores de tela, dificultando a interpretação e correção.
- Problemas simples, como erros de delimitadores, uso incorreto de maiúsculas e minúsculas ou blocos não fechados, geraram frustração e consumo excessivo de tempo.
- A depuração foi apontada como uma das tarefas mais complexas, devido à ausência de recursos visuais como destaque de linha ou marcações em tempo real.
- Estratégias compensatórias incluíram: reexecução manual de trechos, uso de *prints*, consulta a fóruns, blogs, documentação oficial e tentativa e erro.
- Ajuda de colegas, professores e grupos de apoio foi frequentemente necessária diante de barreiras tecnológicas mais severas.
- Essas práticas dialogam com princípios construcionistas, valorizando o aprender fazendo, a experimentação e o ajuste gradual baseado em *feedback*.
- A superação das dificuldades evidencia autonomia, protagonismo e iniciativa dos participantes mesmo em contextos pouco acessíveis.

5.3 Recursos Educacionais e Inclusão

Os relatos dos participantes revelaram diversos desafios enfrentados em ambientes educacionais no ensino superior, especialmente no que se refere à falta de materiais didáticos adaptados, à inexperiência dos professores com recursos de acessibilidade e à estrutura pouco responsiva das atividades acadêmicas. Muitos estudantes citaram a ausência de alternativas acessíveis a conteúdos visuais, como gráficos e diagramas, a dificuldade de acompanhar videoaulas sem audiodescrição e o recebimento de provas e exercícios em formatos não compatíveis com leitores de tela.

Essas experiências evidenciam uma lacuna significativa entre o discurso institucional sobre inclusão e a prática pedagógica cotidiana. A falta de preparação docente para lidar com a diversidade funcional compromete a participação plena dos estudantes cegos e reforça a urgência de se adotar os princípios do Design Universal para a Aprendizagem - *Universal Design for Learning* (DUA/UDL). Conforme Meyer et al. (2014), o UDL propõe a criação de currículos e práticas que ofereçam múltiplas formas de representação, expressão e engajamento, permitindo que todos os alunos, independentemente de suas habilidades, possam aprender de forma significativa.

A ausência de materiais acessíveis e de formação pedagógica adequada cria barreiras adicionais para estudantes com deficiência visual, que precisam constantemente compensar a negligência institucional com esforços extras. Esses achados dialogam diretamente com estudos que evidenciam o mesmo cenário: Zen et al. (2023) iden-

tificam que materiais didáticos frequentemente não são compatíveis com leitores de tela, incluindo apostilas, slides e trechos de código apresentados como imagem, dificultando o acompanhamento das aulas e a realização de atividades básicas. De modo semelhante, Mountapmbeme et al. (2022) destacam que essa falta de acessibilidade, aliada à pouca preparação docente, impacta negativamente o engajamento e a autonomia dos estudantes cegos ao longo de toda a formação acadêmica. Baker et al. (2019) também reforçam que a precariedade dos materiais e a ausência de adaptações sistemáticas intensificam a dependência de apoio externo, contribuindo para a sensação de isolamento acadêmico. Assim, os relatos deste estudo convergem com a literatura ao evidenciar que a produção de recursos educacionais acessíveis não é apenas uma questão técnica, mas um requisito pedagógico fundamental para a equidade no ensino de programação.

Síntese da Subseção

- Participantes relataram falta de materiais didáticos adaptados no ensino superior.
- Professores demonstraram pouca experiência com acessibilidade e recursos inclusivos.
- Estudantes enfrentaram ausência de alternativas acessíveis para conteúdos visuais, como gráficos e diagramas.
- Videoaulas frequentemente não possuíam audiodescrição, dificultando o acompanhamento.
- Provas e exercícios eram disponibilizados em formatos incompatíveis com leitores de tela.
- Há uma distância significativa entre o discurso institucional de inclusão e a prática pedagógica real.
- O Design Universal para a Aprendizagem (DUA/UDL) é recomendado para promover múltiplas formas de representação, expressão e engajamento.
- Ambientes inclusivos requerem políticas pedagógicas estruturadas e proativas.

5.4 Uso e Limites da Inteligência Artificial

Durante a Fase 2 dos experimentos, que contou com a utilização de ferramentas de IA, os participantes relataram percepções amplamente positivas sobre a experiência. A maioria destacou que a IA proporcionou maior agilidade na execução das tarefas, principalmente por meio de **sugestões de código, correções automáticas e explicações contextuais em linguagem natural**. Esse suporte contribuiu para a redução do tempo de execução das atividades e para a diminuição dos bloqueios cognitivos, favorecendo uma experiência de programação mais fluida e menos frustrante.

A acessibilidade também foi beneficiada, ainda que parcialmente. Os participantes relataram maior compreensão do código e das mensagens de erro com o suporte da IA, que atuava como uma ”ponte” entre a complexidade sintática e a compreensão

semântica do problema. No entanto, também foram observadas limitações importantes. Algumas respostas geradas pelas ferramentas não eram totalmente compatíveis com os leitores de tela, ou não eram verbalizadas de forma clara, exigindo adaptações ou ajuda externa. Apesar disso, muitos relataram aumento de autonomia, por conseguirem concluir tarefas sem depender continuamente de colegas ou docentes.

Além disso, emergiram questões relacionadas às **alucinações da IA**. Em um caso específico, a ferramenta gerou duas versões distintas de um mesmo algoritmo, sem sinalização clara para o leitor de tela, e adicionou a palavra "metanfetamina", o que dificultou a escolha da opção correta. Wermelinger (2023) e Chen et al. (2025) apontam sobre a fragilidade das ferramentas de IA em tarefas de programação: embora úteis, elas podem produzir respostas factualmente imprecisas ou estruturalmente incorretas, exigindo do usuário senso crítico e conhecimento prévio para interpretar e validar as sugestões. Outro ponto relevante diz respeito aos **scripts e prompts utilizados pelos participantes**. Observou-se grande variação na forma como os estudantes solicitavam ajuda à IA, o que impactou diretamente a qualidade das respostas. Participantes que forneceram prompts mais detalhados e contextualizados obtiveram retornos mais úteis, enquanto aqueles que usaram descrições vagas ou excessivamente amplas receberam respostas superficiais. Isso está alinhado com os achados de Philbin (2023) e Zawacki-Richter et al. (2019), que destacam a importância do letramento em IA, isto é, a capacidade de formular instruções claras e interpretar criticamente o resultado gerado, como competência essencial para o uso pedagógico eficaz dessas tecnologias.

Esses achados também dialogam com Pandey et al. (2022), que exploram o papel da IA na promoção da aprendizagem personalizada e inclusiva, e com autores como Brotosaputro et al. (2024), que discutem a IA como ferramenta de apoio à equidade digital. No entanto, as limitações observadas, especialmente alucinações, falta de compatibilidade plena com leitores de tela e inconsistências nas respostas, corroboram as advertências de Wilkens et al. (2021) sobre os riscos de se depender de sistemas de IA não totalmente alinhados a requisitos de acessibilidade e verificabilidade. Assim, a IA demonstrou potencial significativo como aliada na inclusão digital, ampliando a autonomia e a compreensão de estudantes cegos. Contudo, sua utilização também evidenciou fragilidades técnicas e pedagógicas, reforçando a necessidade de modelos mais robustos, maior integração com recursos de tecnologia assistiva e formação específica sobre como criar prompts eficazes e validar criticamente as respostas geradas.

Síntese da Subseção

- A maioria dos participantes relatou percepções positivas durante a Fase 2, destacando maior agilidade nas tarefas graças às sugestões de código e correções automáticas.
- O suporte da IA reduziu o tempo de execução das atividades e diminuiu bloqueios cognitivos, tornando a programação mais fluida e menos frustrante.
- A acessibilidade foi parcialmente beneficiada: houve melhor compreensão de código e mensagens de erro com apoio da IA.
- Algumas respostas geradas não eram totalmente compatíveis com leitores de tela, o que exigiu adaptações ou ajuda externa.
- Muitos participantes relataram maior autonomia, conseguindo concluir atividades sem depender de colegas ou docentes.
- Foram identificadas limitações importantes, como alucinações da IA, incluindo a inserção de termos aleatórios.
- A clareza e qualidade das respostas variaram conforme o prompt enviado: instruções detalhadas produziram respostas melhores; prompts vagos geraram retornos superficiais.
- A IA demonstrou potencial para apoiar aprendizagem personalizada e equidade digital, segundo a literatura.
- Persistem fragilidades, como inconsistências, problemas de compatibilidade com leitores de tela e riscos associados à dependência excessiva da tecnologia.
- Conclui-se que a IA é uma aliada promissora, mas exige modelos mais robustos, integração aprimorada com recursos de tecnologia assistiva e formação específica para uso crítico e eficaz.

5.5 Estratégias Individuais de Superação

Os relatos dos participantes evidenciaram um conjunto expressivo de estratégias individuais utilizadas para superar as barreiras encontradas durante as atividades de programação. Diante das limitações de acessibilidade das ferramentas, muitos desenvolveram táticas próprias para lidar com interfaces complexas, mensagens de erro pouco claras e dificuldades na interpretação do código. Entre as estratégias mais comuns, destacou-se o uso de leitores de tela combinados com editores auxiliares, como o bloco de notas, permitindo reorganizar ou isolar trechos de código que eram lidos de forma fragmentada nas IDEs. Essa prática facilitou a revisão manual do conteúdo e tornou a depuração mais eficiente. Outra estratégia relatada consistiu na revisão linha a linha do código, mesmo quando gerado automaticamente pela IA, como forma de compreender sua estrutura e garantir a corretude lógica. Participantes também recorreram à navegação por atalhos e ao mapeamento mental dos ambientes de desenvolvimento, criando rotinas personalizadas para localizar áreas específicas da interface que não eram anunciadas pelo leitor de tela. Além disso, a colaboração eventual com familiares, amigos ou colegas videntes surgiu como recurso

pontual para acessar trechos de código inacessíveis em plataformas como Google Colab.

Essas estratégias individuais revelam não apenas a criatividade dos participantes, mas também a necessidade constante de compensar falhas estruturais nas ferramentas de programação. Ao relatar as táticas desenvolvidas, vários participantes reforçaram a importância de melhorias nas interfaces de IA, como descrições verbais mais detalhadas, organização das sugestões de código e acessibilidade plena nos menus e caixas de diálogo. Embora tais sugestões apontem caminhos para aprimoramentos tecnológicos, elas emergem diretamente de desafios enfrentados cotidianamente e se configuram como respostas adaptativas a barreiras persistentes. Conforme argumentam Ferrari e Hurst (2021), o desenvolvimento de soluções digitais inclusivas demanda a participação ativa dos próprios usuários com deficiência, que são capazes de identificar nuances de uso muitas vezes invisíveis a projetistas e desenvolvedores. As experiências relatadas nesta pesquisa reforçam que tais estratégias individuais, embora eficazes no curto prazo, não devem substituir a responsabilidade coletiva de promover tecnologias realmente acessíveis.

Um aspecto adicional observado nas estratégias de superação refere-se ao uso intensivo da Inteligência Artificial como apoio imediato para a resolução de problemas. Porém, esse recurso, quando utilizado de forma contínua e automática, pode favorecer uma dependência excessiva, reduzindo o desenvolvimento da autonomia cognitiva. Autores como Zawacki-Richter et al. (2019) e Michel-Villarreal e Vilalta-Perdomo (2023) alertam para esse risco, destacando que a IA deve servir como suporte complementar, e não como substituta do processo mental envolvido na programação. Assim, é essencial que as estratégias individuais incluam também práticas de verificação crítica, compreensão conceitual e depuração manual, garantindo que o estudante permaneça protagonista do próprio aprendizado.

Síntese da Subseção

- Participantes desenvolveram diversas estratégias individuais para superar barreiras de acessibilidade durante as atividades de programação.
- Muitos utilizaram leitores de tela em conjunto com editores auxiliares, como o bloco de notas, para reorganizar ou isolar trechos de código que não eram lidos corretamente nas IDEs.
- A revisão linha a linha do código (inclusive do código gerado pela IA) foi uma prática comum para garantir compreensão da estrutura e da lógica.
- Estratégias de navegação por atalhos e mapeamento mental das interfaces ajudaram a localizar elementos da IDE que não eram anunciados pelo leitor de tela.
- Alguns participantes recorreram pontualmente a familiares, amigos ou colegas videntes para acessar trechos de código inacessíveis em plataformas como o Google Colab.
- As estratégias revelam criatividade e esforço constante para compensar falhas estruturais das ferramentas de programação.
- Os participantes destacaram a necessidade de melhorias nas interfaces de IA, como descrições verbais mais detalhadas e sugestões organizadas de forma acessível.
- As táticas desenvolvidas reforçam que soluções tecnológicas realmente acessíveis devem envolver a participação ativa de pessoas com deficiência no processo de design.
- Estratégias individuais, embora úteis, não substituem a responsabilidade institucional e tecnológica de promover acessibilidade plena.
- O uso intensivo da IA como apoio imediato foi observado, mas pode gerar dependência excessiva se não for acompanhado de verificação crítica e compreensão conceitual.
- A IA deve funcionar como suporte complementar, e não como substituta do pensamento lógico e da prática de programação.

5.6 Síntese Representativa

A análise dos dados revelou um panorama multifacetado sobre a experiência de estudantes cegos na aprendizagem de programação, permitindo identificar complementaridades, tensões e contradições entre as categorias temáticas discutidas. As barreiras de acessibilidade em ambientes de programação, como a incompatibilidade com leitores de tela e a baixa responsividade de interfaces, emergem como um obstáculo central, afetando diretamente a autonomia e a produtividade dos estudantes. Ao mesmo tempo, as estratégias de superação adotadas pelos participantes, como o uso de tentativas sucessivas, apoio de colegas e consulta a tutoriais online, demonstram um alto nível de iniciativa e resiliência, ainda que essas ações sejam frequentemente realizadas em condições desfavoráveis.

A discussão apresentada integra os achados empíricos das entrevistas, questionários e experimentos práticos, revelando um panorama complexo sobre a acessibilidade, usabilidade e autonomia de estudantes cegos no ensino de programação mediado por ferramentas digitais. Os resultados evidenciam que os ambientes de desenvolvimento permanecem marcados por barreiras estruturais, como interfaces não padronizadas, elementos visuais não rotulados e inconsistências no suporte a leitores de tela, que exigem estratégias compensatórias e elevam o esforço cognitivo dos estudantes. Ao mesmo tempo, os desafios técnicos relacionados à sintaxe, depuração e reconhecimento de erros tornam o processo de aprendizagem mais lento e dependente de múltiplos recursos externos, reforçando a importância de práticas pedagógicas que promovam autonomia e compreensão conceitual. A análise também mostra que lacunas institucionais, especialmente no que se refere à disponibilidade de materiais acessíveis e à formação docente, impactam diretamente a inclusão, obrigando os estudantes a desenvolver táticas próprias para suprir a ausência de suporte pedagógico adequado. Quando inserida nesse cenário, a Inteligência Artificial emerge como um recurso ambíguo: por um lado, amplia a compreensão do código, acelera a depuração e oferece apoio imediato; por outro, apresenta limitações como alucinações, inconsistências, falta de clareza nas respostas e integração incompleta com tecnologias assistivas. Por fim, as estratégias individuais de superação demonstram a iniciativa dos participantes, mas também evidenciam que soluções acessíveis não podem depender exclusivamente da resiliência dos usuários, e sim de um ecossistema tecnológico e pedagógico comprometido com o "born accessible".

Na Fase 1 do experimento, realizada sem o uso de Inteligência Artificial, observou-se um cenário marcado por dificuldades recorrentes relacionadas à sintaxe, depuração manual, interpretação de erros e navegação em IDEs pouco acessíveis. O processo de codificação mostrou-se mais lento, dependente de múltiplas tentativas, pesquisas externas e da exploração linha a linha do código, ampliando o esforço cognitivo e evidenciando limitações estruturais nas ferramentas utilizadas.

A introdução da Inteligência Artificial na Fase 2 mostrou-se um divisor de águas. As ferramentas de IA, quando acessíveis, contribuíram significativamente para a compreensão de códigos, redução do tempo de execução das tarefas e maior confiança dos participantes na execução de soluções algorítmicas. Contudo, essas melhorias estão condicionadas à integração eficiente entre a IA e os leitores de tela, o que nem sempre foi garantido. Essa contradição evidencia que o potencial das ferramentas de IA é promissor, mas ainda limitado por questões de usabilidade e design acessível.

Do ponto de vista educacional, os relatos revelaram um descompasso entre a demanda por formação inclusiva e a realidade institucional. A escassez de materiais adaptados, a falta de preparação docente e as barreiras curriculares reafirmam a urgência de ações estruturantes para garantir equidade no acesso ao conhecimento. Respondendo às perguntas de pesquisa, pode-se afirmar que:

- (Q1) **Como o uso de ferramentas de Inteligência Artificial (IA) em Ambientes de Desenvolvimento Integrado (IDEs) impacta a acessibilidade de estudan-**

tes programadores cegos? O uso de IA impacta a acessibilidade de forma ambivalente, pois por um lado, melhora a compreensão do código e das mensagens de erro ao oferecer explicações em linguagem natural, funcionar como ponte entre sintaxe e semântica e apoiar a navegação conceitual; por outro, ainda apresenta limitações importantes, como incompatibilidade parcial com leitores de tela, dificuldade de leitura das sugestões geradas (especialmente em plataformas como Google Colab) e problemas de formatação que dificultam a percepção de múltiplas respostas, mantendo barreiras que restringem a autonomia plena dos estudantes cegos.

- (Q2) **De que forma essas ferramentas influenciam a produtividade e o tempo de execução das tarefas de programação realizadas por esses estudantes?** As ferramentas de IA influenciam positivamente a produtividade ao reduzir o tempo de execução das tarefas, acelerar a correção de erros, sugerir melhorias estruturais no código e oferecer apoio imediato durante a depuração. Os estudantes conseguem concluir as atividades com mais rapidez e fluidez quando a IA é acessível. Porém, em casos em que o leitor de tela não consegue acessar o conteúdo gerado, parte do benefício é reduzido pelo tempo gasto tentando localizar ou compreender as respostas.
- (Q3) **Quais são as percepções e desafios relatados pelos participantes em relação ao uso de IA em comparação ao desenvolvimento sem suporte automatizado?** Os participantes relataram percepções predominantemente positivas sobre o uso da IA, destacando maior autonomia, menor frustração, mais clareza no processo de programação e sensação de apoio pedagógico contínuo. No entanto, também foram identificados desafios relevantes, como alucinações (códigos incorretos ou termos aleatórios), respostas incompletas ou confusas, dificuldade de navegação e leitura pelo leitor de tela, múltiplas versões de código sem sinalização adequada e risco de dependência excessiva da IA no processo formativo. Portanto, o uso com IA é visto como vantajoso, mas requer cautela, verificação crítica e melhorias na integração com recursos de tecnologia assistiva.

As percepções dos participantes destacam tanto avanços quanto lacunas na usabilidade das ferramentas, apontando caminhos para melhorias técnicas e pedagógicas.

Capítulo 6

Diretrizes Técnicas e Pedagógicas para a Inclusão de Estudantes Cegos na Programação

Com base nos achados da pesquisa e nas experiências relatadas pelos participantes, este capítulo propõe diretrizes didático-metodológicas que visam promover a inclusão efetiva de estudantes cegos no ensino de programação. As proposições foram elaboradas à luz das dificuldades identificadas, das estratégias de superação utilizadas pelos participantes e da análise das ferramentas acessíveis e do uso da Inteligência Artificial (IA) como recurso de apoio pedagógico.

6.1 Domínio do Leitor de Tela como Pré-requisito Pedagógico

A primeira proposição diz respeito à formação prévia no uso do leitor de tela, especialmente o NVDA, ferramenta gratuita e amplamente utilizada entre os participantes. O domínio eficiente do leitor de tela deve ser considerado um pré-requisito fundamental para qualquer atividade computacional, inclusive as voltadas à aprendizagem da programação. Sem esse domínio, o estudante enfrentará barreiras não apenas na escrita de código, mas também na interação com ambientes de desenvolvimento, sites e demais aplicações educacionais. A formação inicial deve incluir:

Navegação por objeto (usando teclas específicas como 7 e 9, ou atalhos do NVDA). Leitura de mensagens de erro e alertas visuais. Identificação de trechos destacados (como linhas serrilhadas em IDEs, que indicam erros).

6.2 Conhecimento do Sistema Operacional

Paralelamente ao leitor de tela, é imprescindível que o estudante compreenda os recursos básicos do sistema operacional (Windows, Linux ou macOS). Essa familiaridade permitirá a realização de tarefas como: (a) Instalação e configuração de ambientes de desenvolvimento; (b) Edição de variáveis de ambiente (PATH); e (c) Execução de comandos no terminal. Sugerimos que os cursos de programação para estudantes cegos incluam módulos introdutórios de familiarização com o SO em conjunto com o leitor de tela.

6.3 Minimização da Concorrência Cognitiva no Início da Aprendizagem

Diante das dificuldades cognitivas relatadas no uso simultâneo de IDEs e aprendizado da linguagem, recomenda-se que os estudantes iniciem a prática de programação por meio de editores simples, como o Bloco de Notas, e utilizem o terminal como ambiente de execução. Isso permitiria: (a) Foco na lógica e sintaxe da linguagem; (b) Redução da sobrecarga de comandos e atalhos que as IDEs exigem; e (c) Treinamento da leitura e interpretação de erros no terminal. Essa abordagem se mostrou eficaz especialmente entre participantes iniciantes, sendo indicada até que se atinja um nível razoável de familiaridade com a linguagem e com a navegação por tela.

6.4 Introdução Gradual ao Uso de IDEs

Após o domínio dos fundamentos, é possível migrar para IDEs mais robustas, como VS Code, IntelliJ ou Replit, desde que acompanhadas de materiais instrucionais acessíveis. A pesquisa demonstrou que: (a) VS Code é acessível, mas a leitura de sugestões (texto fantasma) é limitada; (b) Ferramentas como Replit e Google Colab apresentam trechos não lidos pelo leitor de tela, exigindo ajuda externa. Sugere-se que tutores e professores forneçam tutoriais com foco na acessibilidade das IDEs, indicando comandos úteis, atalhos e como configurar recursos como leitura de indentação e mensagens de erro.

6.5 Incentivo ao Hábito de Soletrar e Verificar Sintaxe

A soletração de comandos (Leitor de tela), nomes de variáveis e palavras-chave foi citada como prática essencial, especialmente para linguagens como Python e Java, nas quais a diferenciação entre maiúsculas e minúsculas e a indentação são cruciais. Recomenda-se que os docentes incentivem: (a) Revisão linha a linha com o leitor de tela; (b) Criação de hábitos de soletrar e verificar cada símbolo de pontuação; e (c) Utilização de recursos do NVDA para detecção de indentação.

6.6 Formação na Pesquisa Autônoma e no Uso Ético da IA

Uma das principais vantagens observadas no uso da IA foi a capacidade de fornecer explicações e correções rápidas, inclusive com comentários linha a linha. Contudo, a pesquisa também mostrou a necessidade de: (a) Orientar os estudantes quanto ao uso crítico das ferramentas; (b) Explicar como formular boas perguntas (prompts) para IA; e (c) Ensinar a distinguir boas fontes de pesquisa, tanto em buscadores quanto em plataformas como GitHub, YouTube e repositórios de livros gratuitos. A formação para uso da IA deve envolver aspectos técnicos e éticos, destacando a importância da autonomia e do aprendizado consciente, evitando a dependência exclusiva da ferramenta.

6.7 Comandos Essenciais para Programadores Cegos

O domínio dos comandos de acessibilidade é um pré-requisito fundamental para a autonomia de estudantes cegos no processo de aprendizagem da programação. Os participantes da pesquisa relataram, de forma recorrente, que muitas das dificuldades enfrentadas na utilização de ambientes de desenvolvimento não estavam diretamente relacionadas à lógica de programação, mas sim à falta de domínio sobre o leitor de tela e sobre os comandos do sistema operacional e das IDEs. Dessa forma, propõe-se a inclusão de uma etapa introdutória em cursos de programação acessível voltada exclusivamente à familiarização com esses comandos. Essa formação deve contemplar quatro pilares principais: (1) comandos do leitor de tela (NVDA), (2) navegação no sistema operacional, (3) uso de terminal e linha de comando, e (4) comandos básicos de uma IDE acessível, como o Visual Studio Code.

6.7.1 Comandos do NVDA

O NVDA é o leitor de tela mais utilizado entre os participantes da pesquisa, por ser gratuito, compatível com o Windows e altamente personalizável. A tecla NVDA é a tecla modificadora usada para acionar comandos no leitor de tela NVDA. Ela não é uma tecla física específica no teclado, mas pode ser configurada para ser uma das seguintes:
Insert (Ins) → padrão no layout desktop
Caps Lock → padrão no layout laptop
Os comandos abaixo são essenciais para navegação, leitura de código e identificação de erros:

A familiaridade com esses comandos possibilita ao estudante identificar mensagens de erro, navegar entre elementos da interface e compreender estruturas de código com maior autonomia.

Tabela 6.1: Comandos principais do NVDA para programadores cegos

Ação	**Comando**
Pausar/continuar fala	Ctrl
Interromper completamente	Ctrl + Shift
Ler linha atual	NVDA + L
Soletar palavra	NVDA + K, K
Ler tudo	NVDA + ↓
Navegar por objeto (anterior/próximo/pai/filho)	NVDA + Shift + ←/→/↑/↓
Ativar objeto	NVDA + Enter

6.7.2 Navegação no Sistema Operacional

Muitas tarefas exigem que o estudante manipule configurações, instale programas ou navegue por diretórios. Abaixo, alguns comandos fundamentais no Windows; esses comandos favorecem a autonomia digital, especialmente no contexto de ativi-

Tabela 6.2: Comandos úteis do sistema operacional Windows

Ação	**Atalho**
Abrir menu iniciar	Tecla Windows
Executar comando	Windows + R
Alternar entre janelas	Alt + Tab
Abrir terminal	Windows + R → “cmd”
Minimizar todas as janelas	Windows + D

dades que exigem manipulação de arquivos, variáveis de ambiente e instalação de dependências.

6.7.3 Uso de Terminal

Aprender a utilizar o terminal é essencial para atividades como compilação, execução de scripts e navegação em projetos. Recomenda-se que o estudante pratique comandos básicos de terminal, como:

Tabela 6.3: Comandos básicos de terminal (CMD ou PowerShell)

Ação	**Comando**
Ver conteúdo de uma pasta	`dir`
Acessar uma pasta	`cd nome_da_pasta`
Voltar uma pasta	`cd ..`
Executar arquivo Python	`python arquivo.py`
Verificar variáveis de ambiente	`echo %PATH%`

Essa familiarização reduz a dependência de interfaces visuais e permite que o estudante tenha mais controle sobre o ambiente de desenvolvimento.

6.7.4 Comandos do Visual Studio Code (VS Code)

O VS Code foi a IDE mais utilizada na pesquisa, destacando-se por sua acessibilidade relativa. No entanto, exige conhecimento de comandos específicos para uso fluido com leitores de tela. Um aspecto importante observado é que o texto de sugestão automática no VS Code (texto fantasma) é lido apenas uma vez pelo leitor de tela. Caso o estudante deseje ouvir novamente, precisa apagar e digitar novamente o trecho, o que pode gerar frustração caso o aluno não saiba disso. Os principais incluem:

Tabela 6.4: Comandos úteis do Visual Studio Code com leitor de tela

Ação	**Atalho**
Abrir paleta de comandos	Ctrl + Shift + P
Ir para arquivo	Ctrl + P
Abrir terminal embutido	Ctrl + `
Auto completar	Ctrl + Espaço
Ir para próximo erro (Problemas)	Ctrl + Shift + M
Cancelar sugestão	Esc

6.7.5 Configurações do Leitor de Tela

Para que estudantes cegos possam programar com maior autonomia e eficiência, é essencial que o leitor de tela esteja configurado adequadamente para as demandas do ambiente de desenvolvimento. Em especial, é necessário que o NVDA consiga anunciar a quantidade de espaços (fundamental para linguagens como Python), realizar a soletração de palavras e auxiliar na diferenciação entre erros e *warnings* ao utilizar IDEs como o Visual Studio Code. A seguir, apresentam-se orientações práticas para configurar o NVDA de forma a atender a essas necessidades específicas.

1. Anunciar Espaços e Tabulações
Linguagens como Python são altamente sensíveis à indentação. Por isso, é imprescindível que o leitor de tela informe com clareza a quantidade de espaços ou tabulações presentes em uma linha de código. Para habilitar essa função no NVDA:

- Pressione NVDA + N para abrir o menu do NVDA.
- Acesse ”Preferências”> ”Configurações...”.
- No painel lateral esquerdo, selecione a opção ”Revisão de fala”.
- Marque a caixa ”Anunciar espaços”, permitindo que o NVDA verbalize cada espaço em branco.

- Para que tabulações também sejam anunciadas, vá até a seção ”Pontuação/símbolos”e ajuste o nível para ”Algumas”ou ”Todas”.
- Além disso, durante a revisão de código, recomenda-se o uso do modo de leitura caractere por caractere com o comando NVDA + „ o que facilita a identificação precisa da formatação de cada linha.

2. Soletração de Palavras

Durante a programação, é comum que erros estejam relacionados à digitação incorreta de nomes de variáveis, comandos ou palavras-chave. Para auxiliar nesse aspecto, o NVDA oferece comandos de soletração a seguir:

- NVDA + K: soletra a palavra atual.
- NVDA + K (duas vezes) ou NVDA + ;: soletra com descrição fonética, o que pode ser especialmente útil para diferenciar letras similares (como ”d”e ”b”).

Embora o NVDA não soletre automaticamente em todos os contextos, o uso consciente desses atalhos pode reduzir significativamente o tempo gasto na identificação de erros de digitação.

3. Identificação Sonora de Erros e *Warnings* no VS Code

O NVDA não interpreta diretamente os níveis de severidade das mensagens no Visual Studio Code (por exemplo, distinguir se uma mensagem é um erro ou apenas um *warning*). No entanto, algumas estratégias podem ser adotadas para contornar essa limitação: Opção 1: Utilização de Complementos com *Feedback* Sonoro.

É possível instalar extensões como Speech History ou Tone Indicators. Também há a opção de configurar gestos personalizados com sons, usando o addon NVDA Remote Sound Extension, que permite associar alertas sonoros específicos a tipos de mensagens. Opção 2: Configurações no Visual Studio Code Instale a extensão Error Lens (autor: Alexander), que destaca visualmente erros e warnings diretamente no código. Com o foco na linha de código destacada, utilize o comando NVDA + ↑ para que o leitor de tela leia a linha inteira, incluindo mensagens como: “linha 10, erro: variável não definida” “linha 10, aviso: tipo implícito” Dica adicional: pode-se personalizar os sons do próprio sistema operacional Windows para indicar eventos como erros ou alertas, criando uma resposta auditiva mais eficaz quando eventos são disparados pela IDE.

4. Salvando as Configurações no Perfil do NVDA

Após realizar todas as personalizações, recomenda-se salvar as configurações para que elas sejam mantidas após reinicializações:

- Abra novamente o menu do NVDA (NVDA + N).
- Acesse ”Ferramentas”> ”Salvar configuração”.

Esse procedimento garante que os ajustes realizados se tornem parte do perfil padrão de uso do NVDA.

6.8 Considerações Finais

As proposições aqui apresentadas fornecem um referencial prático para educadores, instituições de ensino e desenvolvedores de ambientes de aprendizagem acessíveis, contribuindo para uma educação mais inclusiva, equitativa e tecnicamente eficiente no ensino de programação para pessoas com deficiência visual.

Capítulo 7

Conclusões

O presente estudo investigou o impacto de ferramentas de Inteligência Artificial (IA) na acessibilidade e produtividade de estudantes programadores cegos em Ambientes de Desenvolvimento Integrado (IDEs). A partir de uma intervenção estruturada com múltiplos participantes, combinando atividades práticas, entrevistas e questionários, foi possível compreender de forma aprofundada não apenas as barreiras enfrentadas por esses estudantes, mas também o potencial e as limitações reais da IA quando integrada às práticas de programação acessível.

De modo geral, os achados revelam que a **IA desempenha um papel efetivo como mediadora cognitiva** para estudantes cegos, especialmente quando utilizada para **depuração, reorganização estrutural do código e explicação contextualizada de erros**. As ferramentas geraram reduções significativas no tempo dedicado à resolução de problemas, minimizaram bloqueios cognitivos recorrentes e atuaram como uma camada interpretativa entre o estudante e a estrutura sintática das linguagens de programação. Esse efeito foi particularmente evidente quando a IA ofereceu **feedback textual detalhado**, estruturado e explicativo, um formato altamente compatível com leitores de tela.

Ao mesmo tempo, a intervenção revelou um contraponto marcante entre a experiência sem IA (Fase 1) e com IA (Fase 2). Sem o suporte automatizado, os participantes enfrentaram longos períodos de tentativa e erro, navegação exaustiva linha a linha e dependência de múltiplas buscas externas para decifrar problemas sintáticos e lógicos. A introdução da IA reduziu drasticamente essas dificuldades, reforçando seu potencial como tecnologia assistiva.

Entretanto, a intervenção também expôs um conjunto consistente de limitações, que transcendem aspectos técnicos e se relacionam diretamente ao design das ferramentas. **Não é a IA que se mostra inacessível, mas sim as plataformas que hospedam e exibem as respostas da IA**, responsáveis por mediar a interação entre o estudante e o conteúdo gerado. A acessibilidade de sistemas como Google Colab, Replit ou mesmo componentes do VS Code mostrou-se insuficiente para leitores de tela, gerando lacunas importantes, como: sugestões de código exibidas em ghost text não

lidas pelo leitor de tela; painéis de saída inacessíveis; botões e elementos dinâmicos sem rótulos; dificuldade em acessar o código gerado ou corrigido pela IA. Essas barreiras não impedem a IA de funcionar, mas interrompem o acesso do usuário ao resultado gerado, configurando um problema de acessibilidade das plataformas, e não, a priori, da tecnologia de IA em si.

Esses achados reforçam um ponto central: Quando integrada a interfaces inadequadas, a IA deixa de ser um apoio e passa a ser mais um elemento com o qual o estudante precisa lutar para conseguir aprender, reduzindo o potencial inclusivo da tecnologia. A pesquisa, portanto, demonstra que **não existe aumento efetivo da autonomia estudantil sem melhorias estruturais nos ambientes de desenvolvimento e nas plataformas que hospedam modelos de IA**.

Além disso, os resultados sugerem que a IA não substitui a necessidade de formação docente em acessibilidade digital. A experiência dos participantes mostrou que barreiras educacionais persistem mesmo quando a tecnologia funciona adequadamente: materiais inacessíveis, ausência de padronização na descrição de interfaces, lacunas na mediação pedagógica e desconhecimento de ferramentas assistivas se mantêm como impeditivos significativos. A IA, nesse cenário, atua como apoio, mas não resolve problemas que são essencialmente instrucionais, curriculares e institucionais.

Outro aspecto relevante diz respeito ao protagonismo dos participantes: mesmo em ambientes pouco acessíveis, todos demonstraram estratégias próprias de navegação, correção e adaptação. A IA, quando acessível, potencializou esse protagonismo, ampliando a capacidade de resolver problemas de forma autônoma e fortalecendo a confiança no próprio processo de aprendizagem. Isso indica que a IA pode contribuir não apenas tecnicamente, mas também afetivamente, reduzindo frustrações históricas associadas a IDEs e ferramentas pouco inclusivas.

Em síntese, os resultados permitem afirmar que:

- A IA tem impacto positivo robusto na produtividade e na depuração de código, especialmente pela clareza das explicações.
- A IA melhora a acessibilidade cognitiva, mas não resolve, por si só, a falta de acessibilidade estrutural das IDEs.
- Persistem desafios críticos de navegação, leitura de sugestões e interpretação de código gerado por IA, devido a falhas de compatibilidade com leitores de tela.
- A autonomia dos estudantes aumenta quando IA e leitor de tela operam de forma integrada, mas diminui quando a interface apresenta ruído ou sobrecarga cognitiva.
- A IA é mais eficiente quando usada como instrumento complementar, e não como substituto de práticas pedagógicas acessíveis.

Portanto, a principal conclusão desta pesquisa é que a Inteligência Artificial representa uma oportunidade concreta e promissora para tornar o ensino de programação

mais acessível, porém ainda insuficiente enquanto estiver acoplada a interfaces não projetadas sob a lógica do *"born accessible"*. Torna-se, portanto, necessário articular avanços tecnológicos com diretrizes de design universal, formação docente e políticas institucionais de inclusão para que o potencial da IA seja plenamente concretizado no ensino de programação para estudantes cegos.

7.1 Trabalhos Futuros

Com base nos resultados obtidos, diversos caminhos promissores podem ser explorados em trabalhos futuros. Uma possibilidade é ampliar a amostra e replicar o estudo com um número maior e mais diverso de participantes cegos, incluindo estudantes em diferentes níveis de formação (ensino médio técnico, graduação, cursos livres e pós-graduação) e com variados graus de experiência em programação. Essa expansão permitiria compreender de forma mais profunda como perfis distintos interagem com a IA, quais barreiras são comuns e quais são específicas, além de favorecer a construção de intervenções mais generalizáveis.

Também se recomenda investigar o desempenho e a acessibilidade de ferramentas de IA em diferentes linguagens de programação e em outros ambientes de desenvolvimento. Como os participantes deste estudo utilizaram majoritariamente Python, Java e C, um desdobramento natural envolve testar a IA em linguagens menos exploradas por pessoas cegas, como JavaScript, C, Go e Rust, além daquelas amplamente utilizadas em cursos introdutórios, como Scratch e blocos visuais acessíveis. Outro eixo de análise futura inclui examinar ajustes, personalizações e configurações que possam melhorar a compatibilidade entre leitores de tela e sistemas de IA, otimizando a experiência dos usuários.

Um eixo de investigação particularmente relevante e alinhado às lacunas identificadas consiste no **desenvolvimento de uma IDE acessível com suporte de IA**, concebida desde o início sob a perspectiva *born accessible*. Tal ambiente poderia integrar, de forma nativa e sem remendos posteriores, recursos como:

- suporte pleno a leitores de tela e braile dinâmico;
- navegação exclusivamente via teclado;
- reconhecimento inteligente de erros com explicações sonoras;
- feedback auditivo estrutural (avisos, estados, alertas);
- sugestões de código lidas de forma clara, segmentada e repetível;
- organização semântica de painéis, menus e saídas;
- modos alternativos de visualização de estruturas algorítmicas;
- comandos de voz e prompts adaptados ao fluxo de leitura linear.

A criação de uma IDE acessível com IA integrada representa um avanço significativo para a área, pois eliminaria ou reduziria muitas das barreiras estruturais observadas neste estudo, sobretudo aquelas relacionadas à falta de padronização semântica e à inacessibilidade de painéis de saída, caixas de sugestão e elementos dinâmicos.

Outro campo de investigação importante envolve a **criação de diretrizes pedagógicas e técnicas para o uso da IA na aprendizagem de programação**, com foco tanto em estudantes cegos quanto videntes. Uma contribuição inovadora seria o desenvolvimento de um conjunto estruturado de orientações para a construção de prompts eficientes, éticos e didáticos, contemplando:

- como solicitar explicações de conceitos básicos e intermediários;
- como pedir depuração passo a passo;
- como pedir reescrita acessível de trechos de código;
- como formular perguntas que ajudem na compreensão lógica e algorítmica;
- como identificar e evitar dependência excessiva da IA;
- como utilizar a IA para estudar, revisar e consolidar conteúdos.

Essas diretrizes poderiam se tornar um recurso importante para disciplinas introdutórias, promovendo equidade entre estudantes com diferentes perfis sensoriais e vivências em tecnologia.

Além disso, estudos longitudinais podem acompanhar, ao longo de semestres ou anos, a evolução do aprendizado de programação com e sem o suporte da IA. Investigações dessa natureza permitiriam observar impactos duradouros sobre a autonomia, a autoconfiança, a retenção de conhecimento e a capacidade de resolução de problemas, dimensões que transcendem os efeitos imediatos encontrados neste estudo.

Finalmente, é pertinente investigar abordagens híbridas que combinem práticas pedagógicas tradicionais, uso crítico da IA e integração com recursos de acessibilidade. Essas abordagens poderiam resultar em metodologias didáticas escaláveis, sustentáveis e replicáveis, adaptadas tanto para o ensino presencial quanto remoto, contribuindo para práticas educacionais inclusivas e para a redução das desigualdades de acesso ao aprendizado de programação.

Referências

Abhishek, S., Sathish, H., Kumar, A., e Anjali, T. (2022). Aiding the visually impaired using artificial intelligence and speech recognition technology. In *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*, páginas 1356–1362. IEEE.

Albusays, K., Ludi, S., e 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*, páginas 91–100.

Aler Tubella, A., Mora-Cantallops, M., e Nieves, J. C. (2024). How to teach responsible ai in higher education: Challenges and opportunities. *Ethics and Information Technology*, 26(1):3.

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

Alves, E. G. C., Santos, Q. P., de Jesus, A. F., Azevedo, R. L., e et al. (2025). Inclusão no processo de ensino-aprendizagem e diversidade educacional. *Revista ft*, 29(142):35–36.

Amin, N., Saeed, A., Khalid, A., Usman, M., e 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.

Baker, C. M., Bennett, C. L., e Ladner, R. E. (2019). Educational experiences of blind programmers. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, páginas 759–765.

Bardin, L. (2011). *Análise de conteúdo*. Edições 70, São Paulo.

Bastos, P. A. L. S., Silva, M. S., Ribeiro, N. M., Mota, R. S., e Galvão Filho, T. (2023). Tecnologia assistiva e políticas públicas no brasil. *Cadernos Brasileiros de Terapia Ocupacional*, 31:e3401.

Bhagat, S., Joshi, P., Agarwal, A., e Gupta, S. (2024). Accessibility evaluation of major assistive mobile applications available for the visually impaired. *arXiv preprint*, 2407.17496.

Bonito, M. A. (2015). *Processos da comunicação digital deficiente e invisível: mediações, usos e apropriações dos conteúdos digitais pelas pessoas com deficiência visual no Brasil*. Tese de Doutorado, Universidade do Vale do Rio dos Sinos, São Leopoldo, Brasil. Tese de doutorado.

Bray, A., Devitt, A., Banks, J., Sanchez Fuentes, S., Sandoval, M., Riviou, K., Byrne, D., Flood, M., Reale, J., e Terrenzio, S. (2024). What next for universal design for learning? a systematic literature review of technology in udl implementations at second level. *British Journal of Educational Technology*, 55(1):113–138. Received: 14 July 2022; Accepted: 5 April 2023.

Brotosaputro, G., Supriyadi, A., e Jones, M. (2024). Ai-powered assistive technologies for improved accessibility. *International Transactions on Artificial Intelligence (ITALIC)*, 3(1):76–84.

Caldeira, V. M. M., Lima, A. F., de Souza, N. E. F., Nicolau, P. C., e de Sousa, A. M. (2025). Codificando o futuro: A programação na formação de jovens mentes. *Revista Científica Arbitrada de Estudos em Engenharia*, 7(2). Data de submissão: 17/01/2025. Data de publicação: 17/02/2025.

Cavalcante, R. A. d. S. (2022). Acessibilidade digital enquanto direito humano: diálogos sobre a usabilidade de dispositivos e de estratégias acessíveis com pessoas com deficiência. Dissertação de mestrado, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro. Área de concentração: Educação Inclusiva e Processos Educacionais.

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

Chen, N., Qiu, L. K., Wang, A. Z., Wang, Z., e Yang, Y. (2025). Screen reader users in the vibe coding era: Adaptation, empowerment, and new accessibility landscape.

Corso, V., Mariani, L., Micucci, D., e Riganelli, O. (2024). Assessing ai-based code assistants in method generation tasks. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, páginas 380–381.

dos Santos Borges, J. A. (2000). O sistema dosvox: um novo conceito de acesso de deficientes visuais à informática. Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro (NCE/UFRJ). Disponível em: `http://intervox.nce.ufrj.br/dosvox/`. Acesso em: 16 julho. 2025.

Eckhardt, K., Schiering, I., Gabel, A., Ertas, F., e Müller, S. V. (2019). Visual programming for assistive technologies in rehabilitation and social inclusion of people with intellectual disabilities. In *Proceedings of Mensch und Computer 2019*, páginas 731–735. Association for Computing Machinery.

Espada-Chavarria, R., González-Montesino, R. H., López-Bastías, J. L., e Díaz-Vega, M. (2023). Universal design for learning and instruction: Effective strategies for inclusive higher education. *Education Sciences*, 13(6).

Ferrari, C. e Hurst, A. (2021). Accessible web development: Opportunities to improve the education and practice of web development with a screen reader. *ACM Trans. Access. Comput.*, 14(2).

Galvão Filho, T. (2022). *Tecnologia Assistiva: um itinerário da construção da área no Brasil*. Editora CRV, Curitiba.

Galvão Filho, T. A. (2009). A tecnologia assistiva: de que se trata? In Machado, G. J. C. e Sobral, M. N., editores, *Conexões: educação, comunicação, inclusão e interculturalidade*, páginas 207–235. Redes Editora, Porto Alegre.

Huff, E. W., Boateng, K., Moster, M., Rodeghero, P., e Brinkley, J. (2020). Examining the work experience of programmers with visual impairments. In *2020 ieee international conference on software maintenance and evolution (icsme)*, páginas 707–711. IEEE.

Khan, M. A., Paul, P., Rashid, M., Hossain, M., e Ahad, M. A. R. (2020). An ai-based visual aid with integrated reading assistant for the completely blind. *IEEE Transactions on Human-Machine Systems*, 50(6):507–517.

Kumar, M., Yadav, H., Yadav, J., e Jha, M. (2023). Intellicode: A speech-based programming environment. In *Journal of Xi'an Shiyou University, Natural Science Edition*.

Llerena-Izquierdo, J., Mendez-Reyes, J., Ayala-Carabajo, R., e Andrade-Martinez, C. (2024). Innovations in introductory programming education: The role of ai with google colab and gemini. *Education Sciences*, 14(12).

M S, S. R., Joy, E., e J, L. S. (2024). Websight: An ai-based approach to enhance web accessibility for the visually impaired. In *2024 International Conference on Science Technology Engineering and Management (ICSTEM)*, páginas 1–7.

Meyer, A., Rose, D., e Gordon, D. (2014). *Universal Design for Learning: Theory and Practice*. CAST Professional Publishing, Wakefield. Acesso em: 02 maio 2019.

Michel-Villarreal, R. e Vilalta-Perdomo, E. L. (2023). Challenges and opportunities of generative ai for higher education as explained by chatgpt. *Education Sciences*, 13(9):856.

Mohamed, S., Parvin, A., e Parra, E. (2024). Chatting with ai: Deciphering developer conversations with chatgpt. In *Proceedings of the 21st International Conference on Mining Software Repositories*, MSR '24, página 187–191, New York, NY, USA. Association for Computing Machinery.

Mountapmbeme, A., Okafor, O., e 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.

Nadukuda, N. (2023). Automating front-end development with ai: From code generation to intelligent debugging. *International Journal of Artificial Intelligence & Applications (IJAIAP)*, 2(1):82–88.

Nascimento, M. e Brandão, A. (2019). Um modelo de acessibilidade para cegos em sistemas de programação visual. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 8, página 1467.

Ndlovu, L., Bayaga, A., e Blignaut, S. (2023). *Acceptance of Job Access with Speech (Jaws) as an Assistive Computer Application Software*, páginas 307–318. Springer Nature Switzerland, Cham.

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

Pandey, M., Bondre, S., O'Modhrain, S., e Oney, S. (2022). Accessibility of ui frameworks and libraries for programmers with visual impairments. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, páginas 1–10. IEEE.

Pandey, M., Kameswaran, V., Rao, H. V., O'Modhrain, S., e 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. (1986). *Constructionism: A New Opportunity for Elementary Science Education*. Cambridge.

Philbin, C. A. (2023). Exploring the potential of artificial intelligence program generators in computer programming education for students. *ACM Inroads*, 14(3):30–38.

Pudari, R. (2022). *AI Supported Software Development: Moving Beyond Code Completion*. Tese de Doutorado.

Schanzer, E., Bahram, S., e Krishnamurthi, S. (2019). Accessible ast-based programming for visually-impaired programmers. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, páginas 773–779.

Sikha, V. e Others (2024). Ai-fueled transformation in application development coding. *International Journal of Communication Networks and Information Security (IJCNIS)*, 16(1):78–89.

Sribunruangrit, N., Marque, C., Lenay, C., e 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, páginas 5145–5148.

Storer, K. M., Sampath, H., e Merrick, M. A. A. (2021). " it's just everything outside of the ide that's the problem": Information seeking by software developers with visual impairments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, páginas 1–12.

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.

Vinaykarthik, B. et al. (2022). Design of artificial intelligence (ai) based user experience websites for e-commerce application and future of digital marketing. In *2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, páginas 1023–1029. IEEE.

Wataya, R. S. (2006). O uso de leitores de tela no teleduc. *Interface-Comunicação, Saúde, Educação*, 10:227–242.

Web Accessibility Initiative (WAI) (2022). Introduction to web accessibility. `https://www.w3.org/WAI/fundamentals/accessibility-intro/`. Accessed: April 12, 2023.

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

Wilkens, L., Haage, A., Lüttmann, F., e Bühler, C. (2021). Digital teaching, inclusion and students' needs: Student perspectives on participation and access in higher education. *Social Inclusion*, 9(3):117–129.

Zawacki-Richter, O., Marín, V. I., Bond, M., e Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education–where are the educators? *International journal of educational technology in higher education*, 16(1):1–27.

Zen, E. (2024). *Diretrizes de Acessibilidade em Ambientes de Desenvolvimento Integrado para Estudantes Cegos*. Tese de Doutorado, Universidade Federal de Pelotas (UFPel), Pelotas, Brasil.

Zen, E., Costa, V., e Tavares, T. (2023). Experiências educacionais em disciplinas de programação de computadores: uma análise qualitativa na perspectiva dos estudantes com deficiência visual. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, páginas 960–971, Porto Alegre, RS, Brasil. SBC.

Apêndice A

Termo de Consentimento Livre e Esclarecido (TCLE)

Título do Projeto: Avaliando o Impacto da IA em um Ambiente de Desenvolvimento Integrado (IDE) na Produtividade e Acessibilidade de Estudantes Programadores Cegos

Pesquisadores: Naiara Silva dos Santos (UESB) e Claúdia Pinto Pereira (UEFS) A inclusão digital é uma necessidade essencial, e a acessibilidade na programação é fundamental para alcançar este objetivo.

Este projeto busca desenvolver e avaliar estratégias educacionais que utilizem Inteligência Artificial (IA) para tornar o ensino de programação mais acessível para estudantes com deficiência visual.

Os objetivos são compreender as necessidades e desafios desses estudantes no aprendizado de programação, desenvolver ferramentas e estratégias educacionais acessíveis utilizando IA, e avaliar a eficácia dessas ferramentas e estratégias no apoio ao aprendizado de programação para estudantes com deficiência visual.

Para isso, você será convidado a participar de entrevistas semi-estruturadas, responder a questionários online e, possivelmente, participar de grupos focais. As entrevistas e os questionários abordarão suas experiências e necessidades em relação ao aprendizado de programação.

Os riscos são mínimos e relacionados ao desconforto de responder a perguntas pessoais sobre suas experiências e necessidades. Espera-se que os resultados desta pesquisa contribuam para o desenvolvimento de ferramentas de programação mais acessíveis e eficazes, beneficiando tanto individualmente os participantes quanto a comunidade de programadores com deficiência visual como um todo.

Não há métodos alternativos específicos para este estudo; ele se concentra em entender as necessidades e desenvolver soluções personalizadas.

Você terá acesso contínuo ao pesquisador responsável para esclarecer quaisquer dúvidas antes e durante a realização da pesquisa. Sua participação é voluntária, e você pode recusar ou desistir em qualquer fase da pesquisa sem qualquer penalidade ou prejuízo.

Todas as informações coletadas serão mantidas confidenciais, e sua identidade será preservada em todas as publicações e apresentações dos resultados.

Não haverá despesas para os participantes. Caso haja alguma, será devidamente ressarcida. Com sua permissão, entrevistas poderão ser gravadas para garantir a precisão das informações. As gravações serão utilizadas apenas para fins de pesquisa.

Os dados coletados serão destruídos após a conclusão do estudo ou integrados em um banco de dados anônimo para futuras pesquisas relacionadas. Eles serão mantidos sob a guarda do pesquisador responsável na Universidade Estadual de Feira de Santana por no máximo cinco anos.

Os resultados serão compartilhados com a instituição onde a pesquisa foi realizada e, se possível, com os participantes. Os resultados poderão ser publicados em revistas científicas e apresentados em conferências acadêmicas.

Em caso de danos comprovadamente causados pela pesquisa, você tem o direito de buscar indenização. O Comitê de Ética em Pesquisa (CEP) é um órgão responsável por avaliar e acompanhar projetos de pesquisa envolvendo seres humanos, assegurando que sejam conduzidos de acordo com princípios éticos. Para dúvidas do ponto de vista ético, entre em contato com o Comitê de Ética em Pesquisa da UEFS: cep@uefs.br ou telefone (75)3161-8124.

Feira de Santana-BA, 27 de agosto de 2024.

Apêndice B

Sequência Didática

Objetivos da Sequência Didática

- Avaliar a eficiência e a acessibilidade de ferramentas de programação assistidas por IA.

- Comparar a resolução de problemas de programação com e sem o uso de IA.

- Identificar as barreiras enfrentadas por programadores cegos no uso de diferentes ferramentas e tecnologias.

- Propor melhorias para aumentar a autonomia de programadores cegos, com base no feedback dos participantes.

Considerações

- Linguagem de Programação fica a critério do participante.

- Explicar (eficiência, acessibilidade, autonomia, erros (métricas))

- Autorização para gravação de tela

Passo 1: Apresentação e Contextualização

Duração: 15 minutos

Atividade: Explicar aos participantes o objetivo do teste, o funcionamento básico das ferramentas de IA a serem usadas, como leitores de tela, sugestões de código automatizadas e autocompletar.

Ferramentas: Leitores de tela como NVDA, Dosvox ou JAWS. Ferramentas de IA como GitHub Copilot ou TabNine.

Passo 2: Primeira Atividade - Programação Sem o Uso de IA

Duração: Tempo Livre

Atividade: Os participantes receberão um problema de programação simples, como a criação de um algoritmo para ordenar uma lista de números. Eles devem resolver a tarefa sem o auxílio das ferramentas de IA, utilizando apenas o leitor de tela.

Critérios Avaliados:

-Tempo necessário para resolver o problema.

-Dificuldades relatadas pelos participantes (navegação pelo código, depuração, etc.).

-Número de erros encontrados no código final.

Passo 3: Segunda Atividade - Programação Com o Uso de IA

Duração: 45 minutos

Atividade: Os participantes devem resolver um problema semelhante ao da primeira atividade, desta vez utilizando as funcionalidades de IA (autocompletar, sugestões de código, etc.). O objetivo é avaliar a diferença de experiência ao usar a IA como ferramenta de suporte.

Critérios Avaliados:

-Tempo necessário para resolver o problema com o auxílio da IA.

- Facilidade de uso das sugestões fornecidas pela IA.

- Qualidade e precisão do código comparado à primeira atividade.

Passo 4: Discussão e Feedback

Duração: 30 minutos

Atividade: Após a conclusão das tarefas, será realizado um momento de discussão, onde os participantes poderão expressar suas impressões sobre a experiência, como:

- Sugestões para melhorar a acessibilidade dessas ferramentas. Barreiras enfrentadas em cada atividade (com e sem IA).

- Como as sugestões de autocompletar e a IA impactaram a produtividade.

- Dificuldades específicas com o leitor de tela (se houver), especialmente em relação às sugestões de código.

Ferramentas: Questionário acessível para coletar dados.

Resultados Esperados

Comparação objetiva entre o tempo de resolução e o número de erros com e sem o uso de IA.

Informações detalhadas sobre a experiência dos programadores cegos, focando nas dificuldades e benefícios das ferramentas de IA.

Sugestões práticas para melhorar a acessibilidade de ferramentas de programação baseadas em IA.

Identificação de barreiras que ainda precisam ser superadas para garantir total independência dos programadores cegos ao usar essas tecnologias.

Apêndice C

Roteiro da Entrevista

Objetivo Geral da Entrevista: Obter uma compreensão detalhada das necessidades, percepções e experiências de estudantes com deficiência visual no aprendizado de programação.

Objetivos Específicos da Entrevista:
1. Identificar as principais barreiras enfrentadas por estudantes com deficiência visual no aprendizado de programação.
2. Avaliar a eficácia das ferramentas e tecnologias atuais, incluindo aquelas baseadas em Inteligência Artificial (IA), utilizadas por estudantes e educadores.
3. Coletar sugestões e recomendações para o desenvolvimento de estratégias educacionais e ferramentas de programação mais acessíveis e eficazes.
4. Entender as práticas de ensino e recursos adicionais que poderiam apoiar melhor o aprendizado de estudantes com deficiência visual.
5. Obter respostas sobre as expectativas e preferências dos estudantes em relação às ferramentas de programação acessíveis.

Público-alvo: Estudantes com deficiência visual que estão aprendendo ou aprenderam programação.

Norteadores para a Realização da Entrevista:
Estrutura Semiestruturada: A entrevista será conduzida de forma semiestruturada, permitindo que o entrevistado tenha a liberdade de responder às perguntas da maneira que achar mais apropriada.
Não Interrupção: O entrevistador não poderá interromper o entrevistado enquanto ele estiver respondendo.
Neutralidade do Entrevistador: O entrevistador não deverá orientar ou influenciar as respostas do entrevistado.
Evitar Vieses: Deverão ser evitados vieses e julgamentos de valor em relação às respostas dos entrevistados para garantir a imparcialidade e a integridade dos dados coletados.
Ambiente Confortável: Criar um ambiente confortável e acolhedor para que os

entrevistados se sintam à vontade para compartilhar suas experiências e opiniões. Questionários Online: Além das entrevistas individuais, poderão ser utilizados grupos focais para promover discussões interativas e questionários online para alcançar um número maior de participantes e coletar dados quantitativos e qualitativos.

Critérios para o Convite de Profissionais para a Entrevista: Estudantes com Deficiência Visual em cursos de TICs no Brasil:

o Ter experiência no aprendizado de programação.
o Usar ou ter usado tecnologias assistivas no processo de aprendizagem.

Atividades Pré-entrevista: Elaborar e revisar o roteiro de entrevista, garantindo que todas as questões norteadoras estejam cobertas.

Selecionar e convidar os participantes com base nos critérios definidos.

Agendar as entrevistas, considerando a disponibilidade dos participantes.

Preparar os materiais e ferramentas necessárias para a gravação e anotação das entrevistas.

Obter a aprovação do Comitê de Ética para a realização das entrevistas, garantindo que todos os aspectos éticos sejam considerados.

Realização da Entrevista: - Receber os participantes e fornecer uma breve explicação sobre o objetivo da entrevista e o uso das informações coletadas.
- Obter o consentimento informado dos participantes, garantindo que entendam seus direitos e a confidencialidade das informações. - Conduzir a entrevista seguindo o roteiro preparado, mantendo a flexibilidade para explorar questões emergentes relevantes.
- Registrar as respostas dos participantes por meio de gravação de áudio/vídeo e/ou anotações detalhadas.
- Agradecer aos participantes pela contribuição e oferecer informações de contato para futuras perguntas ou esclarecimentos.

Após a Entrevista: - Transcrever as entrevistas gravadas e organizar as anotações.
- Analisar as transcrições e anotações para identificar temas e padrões recorrentes.
- Classificar as informações em categorias temáticas, como barreiras, soluções, sugestões, etc. - Incorporar os dados obtidos na análise geral do estudo, desenvolvendo recomendações e estratégias baseadas nas respostas dos participantes. - Compartilhar um resumo dos resultados com os participantes, caso tenham interesse, e garantir que suas contribuições sejam reconhecidas no relatório final.

Algumas questões sobre acessibilidade

Em sua opinião, quais são os principais desafios que você enfrenta ao aprender ou trabalhar com programação?

As ferramentas que você utiliza atualmente atendem suas necessidades de acessibilidade? Por favor, explique.

Quais funcionalidades você considera mais importantes em ferramentas de programação acessíveis?

O que você espera de uma ferramenta ou estratégia baseada em IA para ajudar no aprendizado ou trabalho com programação?

Apêndice D

Questionário Online

Seção 1: Perfil do Participante
Nome:
Idade:
Gênero:
( ) Masculino
( ) Feminino
( ) Prefiro não informar

Nível de experiência em programação:
( ) Iniciante
( ) Intermediário
( ) Avançado

Você é:
( ) Estudante curso Técnico
( ) Estudante curso Superior
( ) Estudante curso Pós-Graduação

Linguagem de programação que você mais utiliza:

Ambiente de desenvolvimento que você utiliza:

Qual leitor de tela você usa? (Pode selecionar mais de um)
( ) JAWS
( ) NVDA
( ) VoiceOver
( ) Dosvox

( ) Outro (qual?):

Seção 2: Barreiras no Ensino e na Prática da Programação Quais são as principais dificuldades que você enfrenta ao programar? (Pode marcar mais de uma opção)
( ) Navegação na IDE
( ) Depuração e correção de erros
( ) Interpretação de mensagens de erro
( ) Leitura e escrita de código de forma eficiente
( ) Acesso a materiais didáticos acessíveis
( ) Dificuldade de usar ferramentas visuais
( ) Outro (qual?):

Você acredita que seu leitor de tela é eficiente para a programação?
( ) Sim
( ) Parcialmente
( ) Não

Como você costuma resolver problemas ou depurar seu código?
( ) Utilizo apenas meu conhecimento prévio
( ) Busco ajuda em tutoriais e fóruns online
( ) Utilizo leitores de tela para interpretar erros e mensagens
( ) Peço ajuda para colegas ou professores
( ) Utilizo IA
( ) Outro (qual?):

Quais são as principais barreiras físicas que você enfrenta no aprendizado ou na prática da programação (ambiente acadêmico)? ( ) Infraestrutura inadequada
( ) Ausência de Materiais Adaptados
( ) Falta de suporte especializado (tutores, instrutores com capacitação em acessibilidade)
( ) Dificuldade no uso de equipamentos de hardware
( ) Outro (qual?):

No contexto educacional, quais dificuldades você encontrou ao aprender programação?

Seu material de estudo ou avaliações são adaptados para acessibilidade? Como?

Seção 3: Uso de Inteligência Artificial na Programação Você já utilizou alguma ferramenta de Inteligência Artificial para auxiliar na programação?
( ) Sim
( ) Não
( ) Não sei

Se sim, quais ferramentas você já utilizou? (Pode marcar mais de uma opção)
( ) GitHub Copilot
( ) ChatGPT
( ) Gemini
( ) Claude.ai
( ) Outro (qual?):

Como você avalia a utilidade das ferramentas de IA para programadores cegos?
( ) Muito útil
( ) Útil em algumas situações
( ) Pouco útil
( ) Não vejo diferença

Você já enfrentou dificuldades ao usar IA para programação?
( ) Sim (quais?):
( ) Não

Quais dificuldades você percebe ao usar IA para acessibilidade na programação?

Como você acredita que ferramentas de IA poderiam ser melhoradas para auxiliar programadores cegos?

Seção 4: Percepção sobre a Acessibilidade e Melhorias Em comparação com métodos tradicionais, você acredita que a IA pode tornar a programação mais acessível para pessoas cegas?
( ) Sim
( ) Depende da ferramenta
( ) Não

Você sente que a IA melhora sua produtividade na programação?
( ) Sim, melhora significativamente
( ) Sim, mas ainda há barreiras
( ) Não percebi diferença
( ) Não

Você acha que as instituições de ensino oferecem suporte adequado para programadores cegos?
( ) Sim
( ) Parcialmente
( ) Não

Que melhorias você sugeriria para tornar o ensino de programação mais acessível?

Seção 5: Metodologia de Ensino O professor adaptou a forma de ensino para atender melhor às suas necessidades? Se sim, de que forma?

Você teve apoio suficiente de professores e tutores para superar barreiras no aprendizado?

Você conseguiu acompanhar as aulas práticas de maneira satisfatória?

Seção 6:Considerações Finais Você gostaria de participar de futuras pesquisas sobre acessibilidade na programação?
( ) Sim
( ) Não
Algum outro comentário ou sugestão sobre o tema?

Obrigada por sua participação!