Desafios e Percepções sobre Acessibilidade em Ambientes de Desenvolvimento Integrado

Eliana Zen, PPGC-UFPEL, IFFar, eliana.zen@iffarroupilha.edu.br
Tatiana Aires Tavares, PPGC-UFPEL, tatiana@inf.edu.ufpel.br
Vinícius Kruger da Costa, IFSul, viniciusdacosta@gmail.com

Resumo: Estudantes com deficiência visual podem enfrentar diversas dificuldades para aprender Programação de Computadores e para interagir com Ambientes de Desenvolvimento Integrado (IDEs), o que pode comprometer seu aprendizado, motivação e formação profissional. Conhecer esses obstáculos é o primeiro passo para construir ferramentas de desenvolvimento de software acessíveis. Neste sentido, este trabalho destaca os principais desafios enfrentados por esses estudantes ao utilizar IDEs e apresentada sugestões e recomendações para implementar recursos que aprimorem a acessibilidade desses ambientes e auxiliem os estudantes a superarem esses desafios. Os resultados foram obtidos por meio de revisão da literatura *ad-hoc* e estudo qualitativo e foram analisados utilizando-se a técnica de análise de conteúdo.

Palavras-chave: Acessibilidade, Deficiência Visual, Programação de Computadores.

Challenges and Perceptions on Accessibility in Integrated Development Environments

Abstract: Students with visual impairments may encounter several difficulties in learning Computer Programming and interacting with Integrated Development Environments (IDEs), which can compromise their learning, motivation, and professional development. Understanding these obstacles is the first step towards building accessible software development tools. In this regard, this work highlights the main challenges faced by these students when using IDEs and presents suggestions and recommendations to implement features that enhance the accessibility of these environments and assist students in overcoming these challenges. The results were obtained through ad-hoc literature review and qualitative study, analyzed using content analysis techniques.

Keywords: Accessibility, Visual Impairment, Computer Programming.

1. Introdução

A deficiência visual afeta um número substancial de estudantes matriculados em cursos superiores no país: em 2021, dos 63.404 alunos com deficiência, transtornos globais do desenvolvimento ou altas habilidades/superdotação matriculados em cursos de graduação, 23.654 possuíam alguma deficiência visual, sendo 20.172 com baixa visão e 3.482 cegos (INEP, 2022). Discussões a respeito da importância de garantir a acessibilidade na educação tem ganhado crescente destaque, alinhando-se aos Objetivos de Desenvolvimento Sustentável (ODS) estabelecidos pela Organização das Nações Unidas (ONU). O quarto ODS, em particular, enfatiza a necessidade de assegurar educação de qualidade, inclusiva e equitativa, criando oportunidades de aprendizado para todas as pessoas (ONU, 2015).

Em se tratando de Computação, os currículos dos Cursos Técnicos e Superiores dependem fortemente de representações visuais para ensinar conceitos-chave (Stefik; Hundhausen e Smith, 2011), impondo obstáculos tecnológicos e educacionais significativos para estudantes com deficiência visual (Stefik; Hundhausen e Smith, 2011), que precisam utilizar Tecnologia Assistiva (TA), como leitores e ampliadores de tela ou *Displays* Braile, para interagir com computadores (Potluri *et al.*, 2018). No contexto

específico de programação de computadores, os desafios enfrentados por estudantes com deficiência visual podem ser ampliados, uma vez que a programação exige um conjunto de habilidades, competências e experiências diretamente ligadas a atividades cognitivas complexas para resolver problemas do mundo real e implementá-los em um ambiente computacional (Berssanette *et al.*, 2021).

Na tentativa de facilitar a compreensão dos estudantes sobre os conceitos relacionados à programação de computadores, surgiram tecnologias que combinam técnicas de desenvolvimento visual e geração de código (Cabot, 2020), como ambientes de programação baseados em blocos ou híbridos. Estes ambientes reduzem a quantidade de codificação manual necessária para escrever um programa, elevando o seu nível de abstração. Entretanto, é importante que os estudantes de cursos ligados à área de Computação aprendam a utilizar as ferramentas de desenvolvimento de software profissionais, como os Ambientes de Desenvolvimento Integrado (IDEs), durante a sua formação acadêmica para estarem melhores preparados para o mercado de trabalho. Infelizmente, esses ambientes podem possuir acessibilidade inadequada, o que dificulta, ou mesmo impede, que usuários com deficiência visual façam uso de boa parte dos recursos oferecidos (Potluri *et al.*, 2018).

Para promover esforços para aumentar diversidade e apoiar os estudantes de Computação com deficiência, é importante compreender os desafios que eles precisam superar (Baker; Bennett e Ladner, 2019). Neste sentido, este trabalho busca compreender as dificuldades encontradas por estudantes com deficiência visual ao utilizar IDEs, bem como elencar possíveis melhorias que podem ser adotadas para garantir a acessibilidade para estudantes com deficiência visual. Cabe destacar que a pesquisa se limita a utilização de leitores de tela e IDEs voltados a programação baseada em texto. A utilização de outros recursos de Tecnologia Assistiva (TA) e IDEs voltados a programação baseada em blocos ou híbridos não fazem parte do escopo deste trabalho.

2. Trabalhos Relacionados

Algumas pesquisas buscaram elencar os problemas e limitações de acessibilidade enfrentados por estudantes com deficiência visual em cursos de Computação. (Alves *et al.*, 2022) identificaram que os principais desafios estão relacionados à falta de acessibilidade dos materiais e recursos tecnológicos utilizados, à metodologia de ensino adotada pelos professores, às dificuldades de comunicação interpessoal, à apresentação de gráficos, imagens e informações e à capacitação dos docentes e profissionais especializados envolvidos na formação desses estudantes. (Baker; Bennett e Ladner, 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. Dentre os desafios identificados, os autores citam problemas relacionados à interação dos estudantes com os IDEs, mas não abordam tais desafios de maneira aprofundada.

Por meio de uma Revisão Sistemática da Literatura (RSL), (Robe; Salton e Bertagnolli, 2020) identificaram recursos pedagógicos direcionados ao ensino dos conceitos abordados nas disciplinas de programação de computadores para estudantes com deficiência visual. Também por meio de RSL (Mountapmbeme; Okafor e Ludi, 2022) listaram as barreiras que afetam a produtividade de programadores com deficiência visual e a capacidade dos estudantes com deficiência visual de aprender programação de forma eficaz. Os autores ressaltam que os principais obstáculos estão relacionados à compreensão do código, depuração de código, navegação no código, edição de código e *skimming* de código (não foi encontrada uma tradução adequada para o termo "*skimming*",

portanto, decidiu-se mantê-lo em língua inglesa). (Mealin e Murphy-Hill, 2012) identificaram diversas dificuldades enfrentadas por programadores cegos ao desempenhar tarefas de programação, como a complexidade de uso e falta de acessibilidade dos IDEs. Os autores argumentam que os desafios no uso de IDEs encontrados por esses profissionais podem ser resultado da falta de treinamento no uso dessas tecnologias durante a sua formação acadêmica, afetando sua capacidade de utilizá-las adequadamente no desempenho de suas atividades profissionais.

Esses trabalhos apresentam resultados que auxiliam a compreensão das dificuldades enfrentadas por pessoas com deficiência visual ao aprender a programar e realizar atividades relacionadas à programação de computadores. No entanto, não exploram questões específicas de aprendizagem e utilização dos IDEs por estudantes com deficiência visual. Essa análise é de extrema importância, visto que essas disciplinas são componentes essenciais dos currículos dos cursos da área de Computação e desempenham um papel fundamental na resolução de problemas, no desenvolvimento de software e na criação de soluções tecnológicas.

3. Metodologia

Quanto à abordagem, a pesquisa caracteriza-se como qualitativa, pois busca compreender e explicar os significados, contextos e características dos dados coletados (Gerhardt e Silveira, 2009). Quanto à natureza, pode ser classificada como pesquisa aplicada, já que pretende gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos (Gerhardt e Silveira, 2009). Quantos aos objetivos, a pesquisa pode ser classificada como descritiva (Gil *et al.*, 2002), porque visa descrever as características de determinada população ou fenômeno.

O desenvolvimento do estudo foi delineado em duas etapas:

1. Identificação dos problemas de acessibilidade em IDEs encontrados por pessoas com deficiência visual, publicados na literatura e obtidos por meio de estudo qualitativo, que contou com a participação de professores e estudantes com deficiência visual (Zen E.; Costa e Tavares, 2023); e,
2. Apresentação de estratégias que podem ser adotadas para a solução dos problemas de acessibilidade encontrados.

A revisão da literatura começou com a análise dos trabalhos discutidos em duas Revisões Sistemáticas da Literatura publicadas em 2022: (Zen E.; Siedler e Tavares, 2022) e (Mountapmbeme; Okafor e Ludi, 2022). A partir dos estudos citados por esses autores, foi realizada uma revisão da literatura *ad hoc*, utilizando-se a metodologia *Backward Snowballing*, na qual a lista de referência de um artigo é usada para encontrar artigos relacionados. Foram analisados todos os artigos que tratavam de programação de computadores e pessoas com deficiência visual, bem como os trabalhos referenciados por esses artigos.

As informações obtidas foram analisadas qualitativamente utilizando a técnica de análise de conteúdo, que abrange um conjunto de métodos de análise das comunicações, visando extrair indicadores que possibilitam a inferência de conhecimentos relativos às condições de produção e recepção das mensagens, através da descrição sistemática e objetiva (Bardin, 2015). Para definir os códigos e categorias que seriam utilizados na análise, adotou-se tanto a codificação dedutiva quanto a indutiva. A abordagem dedutiva foi empregada quando as citações estavam diretamente relacionadas aos códigos predefinidos por (Mountapmbeme; Okafor e Ludi, 2022): Compreensão do Código, Depuração de Código, Navegação no Código, Edição de Código e *Skimming* de Código. Já a abordagem indutiva foi utilizada para classificar elementos que não se encaixavam nas

categorias preestabelecidas, gerando os seguintes códigos adicionais: (1) Compreensão da Saída; (2) Sobrecarga Auditiva; e, (3) Leitura de Código-fonte.

4. Resultados e Discussão

Esta seção descreve os resultados obtidos após análise dos trabalhos recuperados na revisão da literatura *ad hoc*. São apresentados os desafios de interação enfrentados por pessoas com deficiência visual ao utilizar IDEs, bem como as principais estratégias que podem ser adotadas para superar esses desafios.

4.1. Desafios de Interação

Os leitores de tela navegam pela interface gráfica do IDE e pelo código-fonte de forma linear, obrigando o usuário a percorrer o código uma linha de cada vez, em sequência (Albusays; Ludi e Huenerfauth, 2017; Mountapmbeme; Okafor e Ludi, 2022). Esse método de interação dificulta o uso de IDEs por pessoas com deficiência visual (Albusays e Ludi, 2016), resultando em vários problemas de interação, que serão detalhados nas subseções seguintes.

4.1.1. Compreensão do Código

Refere-se à habilidade de usuários com deficiência visual para ler e resumir um código-fonte usando leitores de tela (Mountapmbeme; Okafor e Ludi, 2022). Desafios identificados:

- A exploração da **estrutura de um programa** é desafiadora, pois o leitor de telas verbaliza os rótulos dos componentes gráficos apresentados por um explorador de pacotes, mas não consegue transmitir o seu significado lógico (Smith *et al.*, 2003; Mealin e Murphy-Hill, 2012); e,
- Pode ser difícil **distinguir o relacionamento entre classes e subclasses** em uma base de código, especialmente quando existem várias subclasses que herdam todas ou algumas das características da classe principal (Albusays; Ludi e Huenerfauth, 2017).

4.1.2. Depuração de Código

Quando ocorre uma falha no software, os programadores devem realizar três atividades principais para corrigir a falha (Albusays; Ludi e Huenerfauth, 2017): (1) identificar a instrução do código responsável pela falha do software; (2) compreender a origem da falha de software; e, (3) determinar a melhor maneira de remover a causa da falha do software. Desafios identificados:

- IDEs fornecem **informações sobre erros por meio de dicas visuais**, como as cores para realçar erros de sintaxe ou ícones que exibem informações sobre como corrigir o erro ao passar o mouse sobre eles. Esses recursos não estão disponíveis para desenvolvedores com deficiência visual de forma proativa (Potluri *et al.*, 2018); e,
- Informações como **valores de variáveis e pontos de interrupção** não estão disponíveis para um desenvolvedor com deficiência visual, a menos que ações explícitas sejam executadas (Potluri *et al.*, 2018).

4.1.3. Navegação no Código

Diz respeito às estratégias para se obter um entendimento geral do código, envolvendo a dificuldade de encontrar informações em uma base de código sem perder a posição de

foco, bem como a navegação e acesso aos recursos disponibilizados pelo ambiente de programação (Mountapmbeme; Okafor e Ludi, 2022). Desafios identificados:

- Alguns ícones não possuem **rótulos suficientemente claros** para transmitir seu significado ou a funcionalidade associada (Potluri *et al.*, 2018);
- A **movimentação pelo código** usando as teclas de seta não é suficiente devido ao *layout* e à estrutura do código-fonte (Albusays e Ludi, 2016);
- É menos eficiente e mais difícil para um usuário com deficiência visual **localizar informações específicas em uma base de código** (Albusays; Ludi e Huenerfauth, 2017), limitando-se à funcionalidade de pesquisa por meio de palavras-chave ou alguns outros recursos de navegação fornecidos pelos IDEs (Potluri *et al.*, 2018). A busca por meio de palavras-chave pode ser demorada e frustrante, já que uma palavra-chave pode aparecer em vários locais na mesma base de código (Albusays; Ludi e Huenerfauth, 2017; Mealin e Murphy-Hill, 2012);
- Pessoas com deficiência visual podem ter **dificuldade em retornar rapidamente a uma linha específica** ao revisar instruções do código-fonte (Albusays; Ludi e Huenerfauth, 2017; Mealin e Murphy-Hill, 2012) ou quando é necessário trabalhar com múltiplos arquivos de código-fonte simultaneamente;
- Pode ser difícil **compreender o nível do escopo** enquanto se navega por estruturas profundamente aninhadas (Albusays; Ludi e Huenerfauth, 2017; Zen E.; Costa e Tavares, 2023), bem como **detectar o início e o fim de um bloco de código** (Zen E.; Costa e Tavares, 2023);
- As interfaces dos IDEs concentram-se quase que exclusivamente **em estímulos visuais**, utilizando estruturas de menus encapsuladas, caixas de diálogo, ícones e outros elementos gráficos que são difíceis de interagir por meio de leitores de tela (Stefik *et al.*, 2009; Mealin e Murphy-Hill, 2012);
- **Números de linha**, nos editores de código-fonte dos IDEs, não são acessíveis aos leitores de tela (Albusays; Ludi e Huenerfauth, 2017);
- Alguns recursos são abertamente visíveis na interface gráfica do IDE, mas podem ficar **ocultos dentro de vários níveis de hierarquia de navegação** e são difíceis de acessar quando se utiliza leitores de tela (Potluri *et al.*, 2018);
- O processo de **instalação das ferramentas** pode exigir o preenchimento de campos de formulários ou seleção de itens em menus e diretórios que, por vezes, são inacessíveis (Zen E.; Costa e Tavares, 2023);
- Cada **nova versão** de um IDE pode **adicionar novos recursos** e as **funcionalidades já existentes podem sofrer modificações**. Muitas dessas mudanças são representadas visualmente e não existe uma abordagem estruturada para que os desenvolvedores com deficiência visual sejam informados sobre elas (Potluri *et al.*, 2018);
- Não existe uma maneira acessível de consultar os **atalhos** para as funcionalidades da IDE (Baker; Bennett e Ladner, 2019; Potluri *et al.*, 2018; Zen E.; Costa e Tavares, 2023). É necessário solicitar auxílio de outras pessoas ou realizar uma pesquisa externa. Além disso, é necessário memorizar esses atalhos ou manter um arquivo externo com essas informações (Zen E.; Costa e Tavares, 2023);
- Existem situações em que tanto o IDE quanto o leitor de tela utilizam o **mesmo atalho para acessar funcionalidades diferentes**. Há, ainda, casos em que IDEs diferentes utilizam combinações de teclas diferentes para a mesma funcionalidade (Zen E.; Costa e Tavares, 2023);
- Pessoas com **daltonismo** ou baixa visão podem ter dificuldade para **distinguir a imagem de um ícone da imagem do ponteiro do mouse**, quando este último está posicionado sobre o ícone (Zen E.; Costa e Tavares, 2023); e,

- Pessoas com **daltonismo** ou baixa visão podem precisar **ampliar os elementos da interface gráfica**. Embora a maioria dos IDEs permita configurar o tamanho da fonte utilizada para escrita do código, nem todos possibilitam ajustar o tamanho de outros elementos da interface gráfica. É preciso recorrer aos recursos oferecidos pelo sistema operacional, como as opções de configuração de exibição ou o uso de uma lupa (Zen E.; Costa e Tavares, 2023).

4.1.4. Edição de Código

Envolve a identificação de identação do código (utilizado para indicar nível de escopo), utilização de cores diferentes (para realçar a sintaxe do código), escrita correta dos comandos, além do uso extensivo de caracteres não alfanuméricos e problemas relacionados à complexidade da sintaxe das linguagens de programação (Mountapmbeme; Okafor e Ludi, 2022). Desafios identificados:

- Recursos visuais, como o **alinhamento de caracteres especiais ou recuos que marcam o início e o fim do aninhamento**, não são facilmente acessíveis por meio de um leitor de tela (Albusays; Ludi e Huenerfauth, 2017; Huff *et al.*, 2020; Baker; Milne e Ladner, 2015);
- Algumas linguagens de programação usam **recuo de espaço em branco para delimitar blocos de código**, em vez de usar palavras-chave ou chaves (Albusays; Ludi e Huenerfauth, 2017), obrigando pessoas com deficiência visual a realizar a leitura de todos os espaços em branco em cada linha (Potluri *et al.*, 2018);
- Recursos como **preenchimento automático** (que auxiliam o programador a recordar a nomenclatura de classes, métodos, atributos, dentre outros), são exibidos por meio de menu *pop-up* contendo previsões de digitação que podem ser realizadas. Esses recursos não são acessíveis aos leitores de tela (Albusays; Ludi e Huenerfauth, 2017; Zen E.; Costa e Tavares, 2023);
- Os **comentários** são utilizados para tornar um código-fonte mais legível, facilitar tarefas de manutenção ou depuração, documentar o funcionamento de um trecho de código, auxiliar na navegação, localizar erros ou *bugs* no código ou para destacar instruções de código que requerem revisão adicional (Albusays; Ludi e Huenerfauth, 2017). Usuários de leitores de tela podem ter dificuldade para ignorá-los ou navegar até o final desses comentários (Potluri *et al.*, 2018); e,
- Para algumas pessoas com deficiência visual, é desafiador identificar a ausência, excesso ou posicionamento inadequado de **caracteres, operadores e parênteses** no código (Albusays; Ludi e Huenerfauth, 2017).

4.1.5. *Skimming* de Código

Aborda a dificuldade em obter uma visão geral de alto nível do código, decorrentes da leitura sequencial realizada pelos leitores de tela ou mesmo pela falta de acesso a recursos como dobra de código (recurso que permite ocultar e exibir seletivamente seções de um arquivo de código-fonte) (Mountapmbeme; Okafor e Ludi, 2022). Desafios identificados:

- O recurso de **dobra de código** não é acessível aos leitores de tela e, por esse motivo, dificulta a identificação de áreas ocultas no código-fonte (Potluri *et al.*, 2018); e,
- Ao contrário de usuários com visão, que podem obter uma **visão geral da estrutura do código** rolando rapidamente para cima e para baixo em uma página (Potluri *et al.*, 2018), os leitores de tela forçam os usuários com deficiência visual a ler um documento inteiro (Mealin e Murphy-Hill, 2012).

4.1.6. Compreensão da Saída

Pessoas com deficiência visual têm dificuldade para **compreender e verificar se a saída gráfica** gerada por um código-fonte atende às especificações desejadas (Mealin e Murphy-Hill, 2012; Zen E.; Costa e Tavares, 2023). Essas dificuldades podem estar relacionadas à interpretação e compreensão do *layout* da interface e verificação do posicionamento, alinhamento e formatação dos elementos.

4.1.7. Sobrecarga Auditiva

Desenvolvedores com deficiência visual podem experimentar estresse decorrente de **sobrecarga auditiva**, ao receber muitas informações transmitidas no canal de áudio em uma interface (Albusays; Ludi e Huenerfauth, 2017). Além disso, pode ser necessário alterar a voz utilizada pelo leitor de telas, pois com o tempo ela se torna cansativa (Zen E.; Costa e Tavares, 2023).

4.1.8. Leitura de Código-fonte

A **pronúncia das palavras-reservadas das linguagens de programação e das mensagens de erro** (em geral, em língua inglesa), realizada pelo leitor de tela, nem sempre é correta, o que pode atrapalhar o entendimento do conteúdo transmitido. Isso acontece porque, normalmente, o leitor de telas pode estar configurado para leitura na língua da interface em uso (língua portuguesa) e não consegue realizar uma leitura do código-fonte que faça sentido para o ouvinte (Zen E.; Costa e Tavares, 2023).

4.2. Estratégias de enfrentamento

Por meio da revisão da literatura *ad-hoc* e do estudo qualitativo realizados, foram identificadas estratégias e recomendações de acessibilidade que podem oferecer soluções eficazes para os desafios identificados, quais sejam:

- **Visualização e/ou navegação em árvore hierárquica**. Esse recurso permitiria apresentar a estrutura de uma base de código de forma mais clara, assemelhando-se a uma árvore, o que é especialmente útil para ocultar a complexidade inerente do código (Albusays; Ludi e Huenerfauth, 2017). Esse mecanismo forneceria um resumo geral da estrutura do código, contendo detalhes a respeito de *namespaces*, classes e funções do arquivo, permitindo navegar até o componente de código desejado (Potluri *et al.*, 2018);
- ***Feedback* auditivo/sonoro**. Permitiria auxiliar a monitorar processos em segundo plano enquanto são realizadas outras tarefas, auxiliando os programadores a dividir sua atenção entre uma tarefa imediata e a espera pelo resultado de algum processo em execução (Albusays; Ludi e Huenerfauth, 2017). Sons também poderiam ser usados para fornecer informações sobre erros de sintaxe, instruções inválidas e localização atual do cursor no código (Albusays; Ludi e Huenerfauth, 2017), bem como para informar os valores de uma variável em um determinado ponto (Potluri *et al.*, 2018; Stefik *et al.*, 2009).
- **Marcadores ou *tags***. Poderiam ser utilizados para saltar para uma instrução ou linha de código específica. O usuário poderia marcar essa instrução ou linha de código e retornar a ela posteriormente para modificações adicionais (Albusays; Ludi e Huenerfauth, 2017).
- **Indicação de escopo e nível de aninhamento**. Um recurso desse tipo poderia ler em voz alta a localização atual do cursor quando uma combinação especial de teclas de atalho fosse pressionada (Albusays; Ludi e Huenerfauth, 2017).

- **Lista dos erros gerados após a compilação do código-fonte**. O usuário poderia navegar por essa lista e mover o cursor para a linha que contém o erro (Potluri *et al.*, 2018).
- **Resumo das informações relevantes do IDE**. Esse recurso forneceria um resumo das características e funcionalidades que estão distribuídas visualmente pelas múltiplas janelas da interface gráfica da IDE, apresentando-as em formato conciso (Potluri *et al.*, 2018).
- **Utilização de técnicas de navegação granular**. Incorporar abordagens semelhantes à navegação na *Web*, utilizando recursos correspondentes aos elementos da linguagem HTML (títulos, controles de formulário, links, etc.) para facilitar a navegação no código-fonte por meio de classes, funções e blocos de código internos (Potluri *et al.*, 2018).
- **Calcular e verbalizar a quantidade total de espaços em branco**. Nos casos de linguagens que utilizam identação baseada em espaçamento, o leitor de telas deveria contar a quantidade total de espaços em branco e informar o usuário sobre o comprimento exato de um recuo individual (Albusays; Ludi e Huenerfauth, 2017; Zen E.; Costa e Tavares, 2023).
- **Verificar automaticamente uma interface gráfica**. Quando o usuário escrever um código-fonte que implemente uma interface gráfica, a IDE deveria fornecer validadores automatizados para avaliar a consistência do *layout* elaborado a partir das principais normas e diretrizes de acessibilidade para sistemas digitais (Potluri *et al.*, 2019).
- **Alternar em formatos de interação**. Permitir que o usuário possa escolher entre utilizar uma interface gráfica ou um interface textual simplificada (Zen E.; Costa e Tavares, 2023).
- **Fornecer mecanismos para que os usuários possam consultar, anotar e configurar atalhos**. A IDE ofereceria documentação e recursos de ajuda integrados, tornando mais fácil para o usuário aprender a usar, configurar e lembrar os atalhos e comandos para usar as funcionalidades dos IDEs (Zen E.; Costa e Tavares, 2023).

IDEs são recursos essenciais para aprimorar a produtividade e eficiência dos desenvolvedores e aprender a utilizá-los durante a formação acadêmica é crucial para a inserção dos estudantes no mercado de trabalho. No entanto, essas ferramentas ainda precisam ser aprimoradas para garantir a acessibilidade para pessoas com deficiência visual. As estratégias de solução elencadas podem representar um passo significativo na direção da melhoria da acessibilidade dos IDEs para esses estudantes. No entanto, reconhece-se que a acessibilidade é uma questão multifacetada e é fundamental verificar se esse conjunto de recomendações atende à totalidade das necessidades de acessibilidade desse público.

5. Considerações Finais

Este trabalho se propôs a listar as dificuldades enfrentadas por estudantes com deficiência visual ao interagirem com IDEs durante o aprendizado de programação. Além disso, é apresentada uma lista de estratégias recomendadas para aprimorar a acessibilidade dessas ferramentas. Os resultados foram obtidos por meio de revisão da literatura *ad-hoc* e estudo qualitativo com docentes e estudantes com deficiência visual.

A pesquisa demonstrou a existência de vários desafios enfrentados por esses estudantes ao utilizar IDEs, incluindo a dificuldade de compreender a estrutura de um código-fonte, encontrar e corrigir erros no código, navegar pelo código e pela

interface da IDE, editar um arquivo de código e obter uma visão geral de sua estrutura. Além disso, foram identificadas dificuldades associadas à compreensão de uma saída gráfica gerada pelo código, sobrecarga auditiva e problemas na compreensão da leitura de código realizada pelo leitor de tela. Cabe destacar que a maioria dos desafios apresentados decorre da leitura sequencial realizada pelos leitores de tela. Tais desafios podem comprometer o aprendizado dos estudantes, influenciar na sua motivação e comprometimento, e em alguns casos, levar a evasão dos cursos de Computação.

A utilização de IDEs durante a formação acadêmica dos estudantes é fundamental para que eles se tornem profissionais capacitados. É, portanto, um componente crucial para garantir que esses estudantes adquiram as habilidades necessárias e estejam devidamente preparados para enfrentar com sucesso os desafios do mercado de trabalho na área de computação. É imperativo que a pesquisa e o desenvolvimento contínuos de soluções de acessibilidade abordem esses desafios de maneira abrangente e eficaz, a fim de construir IDEs verdadeiramente inclusivos e acessíveis.

Sendo assim, fases subsequentes desta pesquisa abrangem a elaboração e a validação de um conjunto de recomendações de acessibilidade em IDEs voltadas para estudantes com deficiência visual. O propósito é proporcionar-lhes uma maior autonomia e eficiência na aprendizagem e na realização de tarefas relacionadas à programação de computadores. Além disso, ao promover a acessibilidade em IDEs para pessoas com deficiência visual, não apenas se fomenta a educação em Computação desses estudantes, mas também se contribui para a indústria, incentivando uma formação mais inclusiva e ampliando as oportunidades de participação no mercado de trabalho.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Albusays, K.; Ludi, S. Eliciting programming challenges faced by developers with visual impairments: exploratory study. In: **Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering**. [S.l.: s.n.], 2016. p. 82–85.

Albusays, K.; Ludi, S.; Huenerfauth, M. 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**. [S.l.: s.n.], 2017. p. 91–100.

Alves, L. F. *et al.* Estudantes com deficiência visual em computação: participação, perspectivas e desafios enfrentados. In: SBC. **Anais do II Simpósio Brasileiro de Educação em Computação**. [S.l.], 2022. p. 67–76.

Baker, C. M.; Bennett, C. L.; Ladner, R. E. Educational experiences of blind programmers. In: **Proceedings of the 50th ACM Technical Symposium on Computer Science Education**. [S.l.: s.n.], 2019. p. 759–765.

Baker, C. M.; Milne, L. R.; Ladner, R. E. Structjumper: A tool to help blind programmers navigate and understand the structure of code. In: **Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems**. [S.l.: s.n.], 2015. p. 3043–3052.

Bardin, L. **Análise de Conteúdo**. [S.l.]: Edições 70, 2015.

Berssanette, J. H. *et al.* Metodologias ativas de aprendizagem e a teoria da carga cognitiva para a construção de caminhos no ensino de programação de computadores. Universidade Tecnológica Federal do Paraná, 2021.

Cabot, J. Positioning of the low-code movement within the field of model-driven engineering. In: **Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings**. [S.l.: s.n.], 2020. p. 1–3.

Gerhardt, T. E.; Silveira, D. T. Métodos de pesquisa. universidade aberta do brasil–uab/ufrgs. **Porto Alegre: Editora da UFRGS**, p. 120, 2009.

Gil, A. C. *et al.* **Como elaborar projetos de pesquisa**. [S.l.]: Atlas São Paulo, 2002. v. 4.

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

INEP. **Censo da Educação Superior 2021**. [S.l.]: Ministério da Educação, Diretoria de Estatísticas Educacionais, Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, 2022. ⟨https://download.inep.gov.br/educacao_superior/censo_superior/documentos/2021/apresentacao_censo_da_educacao_superior_2021.pdf⟩. (Accessed on 05/08/2023).

Mealin, S.; Murphy-Hill, E. An exploratory study of blind software developers. In: IEEE. **2012 ieee symposium on visual languages and human-centric computing (vl/hcc)**. [S.l.], 2012. p. 71–74.

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

ONU, O. d. N. U. **Objetivos de Desenvolvimento Sustentável**. 2015. ⟨https://brasil.un.org/pt-br/sdgs⟩. (Accessed on 09/29/2023).

Potluri, V.; He, L.; Chen, C.; Froehlich, J. E.; Mankoff, J. A multi-modal approach for blind and visually impaired developers to edit webpage designs. In: **Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility**. [S.l.: s.n.], 2019. p. 612–614.

Potluri, V. *et al.* Codetalk: Improving programming environment accessibility for visually impaired developers. In: **Proceedings of the 2018 chi conference on human factors in computing systems**. [S.l.: s.n.], 2018. p. 1–11.

Robe, R.; Salton, B. P.; Bertagnolli, S. Recursos pedagógicos para o ensino de programação de estudantes com deficiência visual: uma revisão sistemática da literatura. **RENOTE**, v. 18, n. 1, 2020.

Smith, A. C. *et al.* Nonvisual tool for navigating hierarchical structures. **ACM SIGACCESS Accessibility and Computing**, ACM New York, NY, USA, n. 77-78, p. 133–139, 2003.

Stefik, A.; Haywood, A.; Mansoor, S.; Dunda, B.; Garcia, D. Sodbeans. In: IEEE. **2009 IEEE 17th International Conference on Program Comprehension**. [S.l.], 2009. p. 293–294.

Stefik, A. M.; Hundhausen, C.; Smith, D. On the design of an educational infrastructure for the blind and visually impaired in computer science. In: **Proceedings of the 42nd ACM technical symposium on Computer science education**. [S.l.: s.n.], 2011. p. 571–576.

Zen E.; Costa, V. K. d.; Tavares, T. A. Understanding the accessibility barriers faced by learners with visual impairments in computer programming. In: **XXII Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais**. [S.l.: s.n.], 2023.

Zen E.; Siedler, M. d. S. C. V. K. d.; Tavares, T. A. Assistive technology to assist the visually impaired in the use of icts: A systematic literature review. In: **XVIII Brazilian Symposium on Information Systems**. [S.l.: s.n.], 2022. p. 1–8.