Programando às cegas: investigando a acessibilidade de ambientes de desenvolvimento de software

Felipe Lúcio do Nascimento[1], **Pedro Luis Saraiva Barbosa**[2], **Windson Viana**[1]

[1]Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brazil

[2]Instituto Federal do Ceará (IFCE)
Cedro – CE – Brazil

felipe.lucio@alu.ufc.br, windson@virtual.ufc.br, pedro.barbosa@ifce.edu.br

Abstract. *This work investigates the problems faced by Brazilian developers who are blind (DWB). We adopted two perspectives in the research: the accessibility assessment through checking tools and a survey with 10 DWB. We evaluated four development tools with accessibility checkers. The results still indicate the strong presence of barriers in the software development context. We also pointed out which good usage and configuration practices can strongly contribute to DWB' daily work.*

Resumo. *Este trabalho investiga os problemas enfrentados por desenvolvedores brasileiros com deficiência visual. Duas perspectivas foram adotadas na pesquisa: a avaliação de acessibilidade por meio de ferramentas de checagem e um survey com desenvolvedores brasileiros. Um questionário foi aplicado a 10 desenvolvedores com deficiência visual e quatro ferramentas comuns do ambiente de desenvolvimento foram avaliadas com checadores de acessibilidade. Os primeiros resultados indicam ainda a forte presença de barreiras no contexto de desenvolvimento. Também foi possível apontar quais boas práticas de uso e configuração podem contribuir fortemente com o trabalho diário de desenvolvedores que possuem deficiência visual.*

1. Introdução

O ciclo de desenvolvimento de software exige dos programadores o uso contínuo de um conjunto de ferramentas como editores, compiladores e ferramentas de versionamento que geralmente são integradas em um mesmo ambiente [Petrausch and Loitsh 2017]. Entretanto, outras ferramentas também fazem parte de muitos ciclos de desenvolvimento como consoles de serviços (e.g., Firebase), soluções de testes de Web Services (e.g. Postman), softwares de gestão de equipe (e.g., Jira), comunicação (e.g., Slack, Discord), emuladores de smartphones, além da própria consulta de documentação e fóruns online.

Apesar das facilidades promovidas por essas ferramentas, muitos softwares e sites usam unicamente indicações visuais para informações chaves no processo de desenvolvimento. Por exemplo, ambientes de programação que indicam erros de sintaxe por demarcações visuais nas interfaces ou linhas para delimitar blocos de um programa (indentação). Ou ainda, as ferramentas Web que não seguem padrões de acessibilidade recomendados pelo W3C por meio do guia do WCAG [1]. Os desenvolvedores cegos

[1]https://www.w3.org/WAI/standards-guidelines/wcag/

têm à sua disposição apenas a sonorização dos textos que um programa leitor de tela pode apresentar nessas interfaces [Pandey et al. 2021] [Potlurt et al. 2018]. Esses problemas de acessibilidade, em especial, das IDEs já foram apontados em estudos anteriores [Albusays and Ludi 2016][Petrausch and Loitsh 2017][Potlurt et al. 2018]. Os autores investigaram dificuldades enfrentadas por desenvolvedores cegos e suas estratégias para se adaptar e transpor as barreiras no âmbito do desenvolvimento em geral, procurando elicitar requisitos para a elaboração de novos recursos de acessibilidade ou aprimoramento dos já existentes [Luque et al. 2018][Guide 2019]. Essas investigações geraram melhorias em alguns ambientes de desenvolvimento integrado, a saber: teclas de atalho para navegação no editor de texto e nos menus, aumento e modificação nas fontes de texto dos editores e dos botões, e esquema de cores em alto contraste. Entretanto, esses problemas ainda persistem em muitas outras ferramentas do ciclo de desenvolvimento.

Diante deste contexto, este artigo apresenta os resultados de um pesquisa sobre acessibilidade com desenvolvedores com deficiência visual (DDVs) brasileiros e uma análise de ferramentas envolvidas no ciclo de desenvolvimento. Como primeira etapa da pesquisa, um questionário foi aplicado com 10 DDVs. Além disso, quatro ferramentas Web (i.e., Jira, Postman, Bugzilla, Firebase) foram avaliadas com checadores de acessibilidade. As seções seguintes do artigo detalham a metodologia dessas duas etapas iniciais e os seus principais resultados.

2. Avaliação com desenvolvedores

2.1. Survey

Para a realização da primeira etapa da pesquisa, um questionário foi elaborado contendo duas partes. A primeira possuía perguntas sobre o perfil do desenvolvedor (e.g., e-mail, nome, sexo, grau de instrução) além de questões sobre sua atuação profissional, i.e., tecnologias assistivas utilizadas, área em que atua, linguagens nas quais sabe programar e seu tempo de experiência em desenvolvimento de software. A segunda parte estava dividida em função das dificuldades de acessibilidade enfrentadas pelos entrevistados em cada etapa investigada do ciclo de desenvolvimento de um software. As questões versavam sobre: dificuldades ao escrever o código da aplicação, ao fazer depuração do código da aplicação, ao fazer versionamento do código, e por fim, dificuldades em implantar a aplicação, que trata, por exemplo, sobre fazê-la funcionar em um servidor (aplicação Web), ou publicá-la em uma loja de aplicativos.

Nesta segunda parte do questionário, para facilitar seu preenchimento, cada questão possuía um conjunto de opções que se iniciavam com o trecho “Não consigo facilmente (...)”. A semântica de cada uma dessas opções é a de que, ou o desenvolvedor não consegue executar a ação ou não consegue fazer isso com facilidade. É importante, também, dizer que o respondente podia marcar mais de uma opção quando se identificava com mais de uma dificuldade listada e, além disso, a opção “Outros” abriu ao desenvolvedor a oportunidade de testemunhar sobre as suas próprias dificuldades, caso não estivessem listadas na questão. O questionário completo foi elaborado na plataforma Google Forms e o seu link foi compartilhado por listas de e-mail e grupos de WhatsApp direcionados a desenvolvedores com algum grau de deficiência visual.

2.2. Perfil dos respondentes

Ao todo, dez participantes responderam o questionário (nove homens e uma mulher). A figura 1 detalha o perfil dos respondentes. Para ser possível mencionar particularidades encontradas em cada resposta, o texto contará com os termos P1 a P10, que designaram o primeiro participante, o segundo, e assim por diante, conforme a ordem cronológica em que responderam o questionário. Apenas um participante tem menos de 20 anos, quatro participantes têm entre 20 e 29 anos, 3 participantes têm entre 30 e 35 anos. Por fim, dois dos participantes têm mais de 60 anos. Quanto ao grau de instrução dos desenvolvedores que responderam ao questionário, todos os participantes concluíram o Ensino Médio e 5 deles, possuem pelo menos o Ensino Superior completo. Quanto ao grau de deficiência visual, 80% dos respondentes declarou ter cegueira em ambos os olhos, o que inclui nessa quantidade pessoas cujos olhos ainda possuem alguma percepção luminosa, mas que não são consideradas de baixa visão. O participante P3 informou ter cegueira em um olho e baixa visão no outro e já P10 informou ter baixão visão em ambos os olhos.

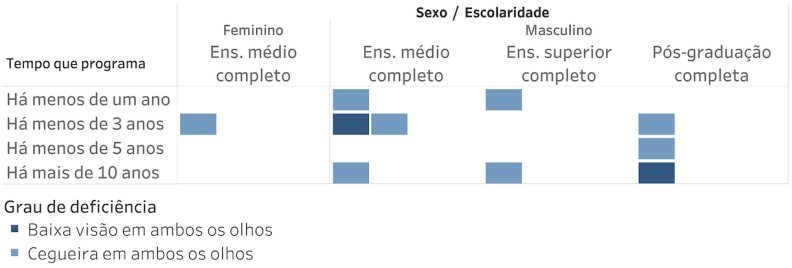

Figura 1. Caracterização dos sujeitos.

Com respeito a tecnologias assistivas usadas pelos entrevistados, nove deles usam leitor de tela e um usa a fonte do sistema ampliada. Três deles usam linha Braille. Os participantes P1 e P3 usam Dosvox e a baixa resolução de tela, o que torna os itens maiores no monitor. Quanto aos leitores de tela, 5 pessoas responderam usar o NVDA. Um desenvolvedor declarou navegar com o Voice Over, do sistema Mac, outro disse usar o Emacspeak, e um terceiro participante respondeu usar o Talkback, este disponível no sistema Android. O Dosvox é usado por único participante.

3. Resultados do Survey

3.1. Perfil de atuação

Com relação às principais ferramentas usadas pelos DDVs, os resultados indicaram as IDEs Eclipse (5 deles), o Visual Studio (3 deles) e um apontou IDE's da Jetbrains. A nona pergunta do questionário versou sobre a área de atuação dos usuários. A maioria deles trabalha com a programação para *desktop* (5) e para a Web (3 deles). Dois participantes afirmaram trabalhar com a área de programação para dispositivos móveis. P1 escreveu que trabalha com testes de acessibilidade de software e o P7 escreveu que trabalha com qualidade e testes. P3 além de programar para *desktop*, selecionou também programação de jogos. Já P10 escreveu que trabalha com a área acadêmica/científica. A Figura 2 apresenta uma relação entre editor/IDE e áreas de atuação. Como pode ser visto, o Eclipse é o mais usado para a programação para a WEB e o Notepad é o mais utilizado para a programação para *desktop*.

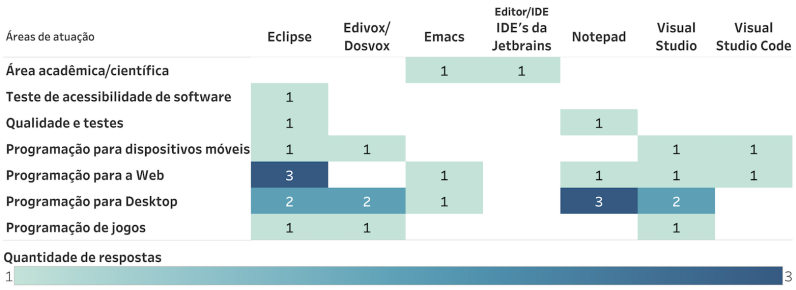

Figura 2. Editor/IDE por área de atuação.

A respeito de quais linguagens de programação os desenvolvedores utilizam em seu cotidiano, de acordo com as respostas, 6 dos participantes declararam usar Python. A linguagem C# aparece em segundo lugar com 4 participantes. As linguagens Java, Javascript e C/C++ aparecem em terceiro lugar com 3 participantes, cada uma. As linguagens Pascal/Delphi foram citadas uma vez cada. Dos entrevistados que escolheram a opção "Outros", P3 relatou utilizar BGT, e P5 disse que usa Bash em suas atividades. O que chama atenção é o uso da linguagem Python. Por ser uma linguagem que não possui sinais gráficos que delimitam os blocos de código (parênteses, colchetes e chaves), seria natural se esperar que poucos a usassem devido à dificuldade de identificar a indentação. Entretanto, a existência de um interpretador, permitindo que as instruções sejam testadas antes da sua inserção no código, e o número de propósitos para os quais a linguagem pode ser empregada, facilitam sua popularidade entre os DDVs.

3.2. Ciclo de desenvolvimento

A segunda fase do questionário trata das dificuldades enfrentadas, levando-se em conta cada etapa do ciclo de desenvolvimento de software (edição, depuração, versionamento e implantação). Os problemas listados correspondem a possíveis ações que o desenvolvedor pode executar e que podem ser malsucedidas. Cada questão do formulário corresponde a uma etapa do ciclo e os participantes foram convidados a selecionar, dentre os problemas listados, quais eles enfrentam durante o seu trabalho (mais de uma opção, se necessário), bem como tiveram a liberdade de relatar outros problemas pelos quais passam e que não foram elencados no questionário. Dos dez participantes, apenas **oito** responderam essas questões. Um dos participantes (P1) trabalhava apenas com teste de software e reportou ter maiores dificuldades no acesso a páginas Web.

3.2.1. Dificuldades ao escrever o código da aplicação

Nesta primeira questão da segunda etapa do questionário, as dificuldades abordadas são as de escrita, revisão e compreensão de código, momentos em que o programa pode potencialmente ser editado pelo desenvolvedor. Assim, conforme os resultados (apresentados na Figura 3), 7 dos participantes responderam enfrentar dificuldades para posicionar elementos gráficos em layouts; 5 dos entrevistados relataram dificuldades em passar para outros arquivos que desejam modificar. Quatro participantes apontaram a questão do alinhamento e o recuo no código da aplicação como problemas. Já as dificuldades de saber o tipo das variáveis no meio de um código, navegar entre as pastas e arquivos do projeto, acessar o campo de busca da IDE e depurar erros com facilidade tiveram uma resposta,

cada. Ressalta-se que P3, o participante com baixa-visão, reportou três problemas apesar de combinar a baixa resolução do monitor com leitores de tela. Além disso, o participante P4 declarou na opção "Outros" que não consegue livrar o código de erros com facilidade.

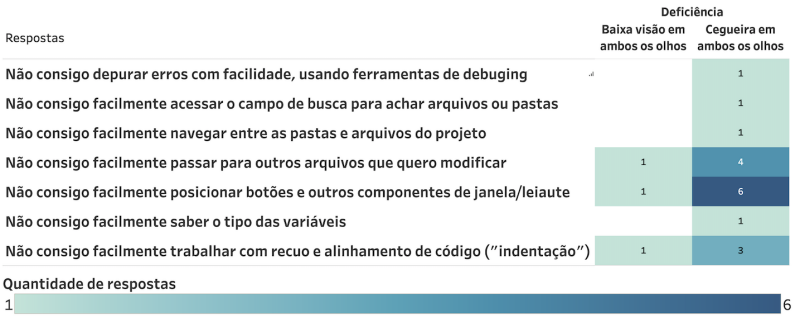

Figura 3. Dificuldades ao escrever o código da aplicação.

3.2.2. Dificuldades ao fazer depuração/debugging no código da aplicação

Esta questão versava sobre dificuldades enfrentadas na identificação de falha no projeto como um todo durante a compilação ou execução ou na localização da posição de um erro no código. Para essa população de desenvolvedores entrevistada, quase todas as dificuldades de depuração, de uma maneira geral, são vivenciadas por eles. Os resultados estão apresentados na Figura 4. Os problemas em executar a aplicação sem usar atalhos do teclado e a dificuldade em acompanhar a execução da aplicação em modo de depuração foram relatadas por 6 participantes. As dificuldades em compilar a aplicação e executá-la sem usar os atalhos do teclado, e também em navegar pela janela de retorno do programa até o ponto que causou o erro foram marcadas por 5 entrevistados. A dificuldade em identificar o ponto do código onde houve erro de sintaxe foi selecionada por 4 participantes; a dificuldade em adicionar pontos de parada (*breakpoints*) foi declarada por 2 participantes; já o problema em saber se houve erro de compilação foi respondido por P4.

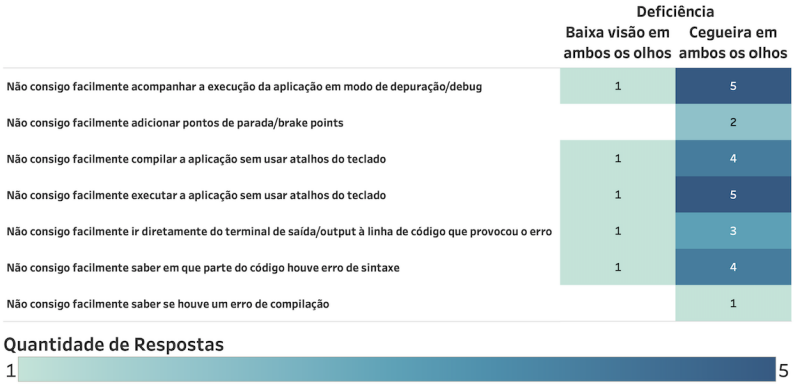

Figura 4. Dificuldades ao fazer depuração/debugging do código da aplicação.

3.2.3. Dificuldades ao fazer versionamento e implantação

Uma vez que o propósito de uma IDE é integrar a maior parte possível do ciclo de implementação de software, o versionamento também está presente nessas ferramentas, visto que é importante para o contexto atual de desenvolvimento em equipe. Entretanto, este recurso também é um foco de barreiras à continuidade do fluxo de trabalho por parte dos desenvolvedores com deficiência visual. Os resultados estão apresentados na Figura 5. As dificuldades em saber o histórico de alterações do repositório, e em resolver conflitos foram relatadas por 4 entrevistados. O problema em mudar de *branch* e o problema em mesclar *branches* foi relatado por 3 participantes. Por fim, as dificuldades em fazer download do repositório, fazer *upload* e realizar *commits* foram relatadas por 2 deles.

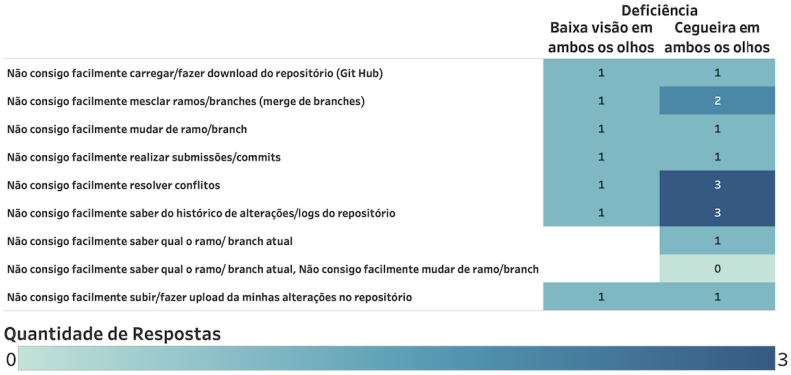

Figura 5. Dificuldades ao fazer versionamento de código dentro da IDE.

Com relação às fases de implantação de um software, os resultados estão apresentados na Figura 6. Tiveram 4 pessoas que indicaram ter dificuldades em configurar um repositório que faz alterações em um servidor automaticamente; 3 participantes disseram enfrentar problemas ao tentar gerar um arquivo executável para *desktop*; 2 deles não conseguem facilmente testar a sua aplicação a partir de uma pasta local; por fim, P5 respondeu não encontrar facilidade ao tentar gerar um executável para dispositivo móvel dentro do Android Studio.

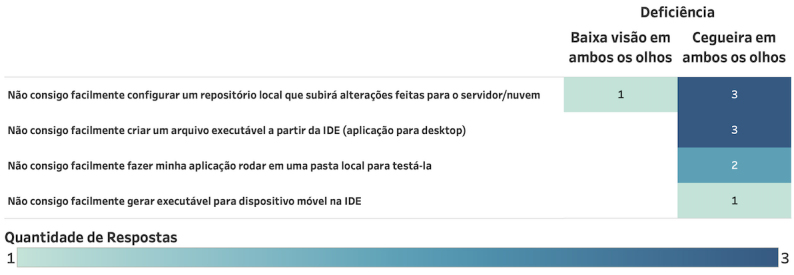

Figura 6. Dificuldades ao implantar a aplicação.

4. Avaliação de acessibilidade com checadores

O intuito dessa segunda etapa da pesquisa foi realizar avaliações de acessibilidade de ferramentas Web por meio de inspeção e análise automatizada. A avaliação de inspeção simulava o uso do site com o teclado e um leitor de tela ativado (no caso, o Narrador do

Windows[2]) e foi feita por um dos autores da pesquisa, que tem experiência em avaliação de acessibilidade. Já a análise automática envolveu o uso de três softwares: WAVE [3], Acessibility Insights [4] e Lighthouse do Google Chrome [5]. Essas ferramentas checam automaticamente os arquivos HTML e CSS dos sites em busca de inconformidades de acessibilidade, em geral, baseadas no WCAG 2.1.

4.1. Ferramentas e procedimento

Nesta fase inicial da pesquisa, foram escolhidas quatro plataformas Web comumente usadas em ambientes de desenvolvimento de software: Jira, Postman, Bugzilla e Firebase. Sua escolha foi feita a partir de relatos informais de alunos e desenvolvedores cegos (entre eles um dos autores do artigo) que tinham enfrentado dificuldades em utilizá-las. A avaliação consistiu em acessar apenas uma página dessas ferramentas em que fosse possível executar alguma tarefa. Por exemplo, no caso do Postman, foi checada a página de teste de um serviço Web. No caso do Bugzilla, checou-se uma página em que se pudesse ler detalhes de um bug [6]. Para o Firebase, foi escolhida a página que contém a árvore JSON do Realtime Database e, por fim, para o Jira, foi avaliada uma página do Jira do projeto Kafka do Apache [7].

4.2. Resultados

	WAVE	Accessibiity Insights	Google Lighthouse
Jira do Apache	43 erros graves (37 links vazios); 23 erros de contraste; 49 alertas.	11 erros graves; 22 erros de contraste.	87
Firebase	24 erros graves (22 elementos sem label); 1 erro de contraste; 2 alertas.	27 erros graves.	85
Bugzilla	21 erros graves (14 elementos com texto faltando); 0 erros de contraste; 43 alertas.	24 erros graves.	83
Postman	9 erros graves; 21 erros de constraste (9 em dark mode); 41 Alertas.	6 erros graves; 22 erros de contraste (7 em dark mode).	61

Figura 7. Avaliação de Acessibilidade.

A Figura 7 apresenta um resumo dos resultados da avaliação. Vale ressaltar que o Lighthouse indica um valor de 0 a 100, onde 100 indica total conformidade aos padrões de acessibilidade. Como pode ser percebido, o **Postman** foi a ferramenta com pior nível de acessibilidade. Embora tenha menos erros quantitativamente, a gravidade dos erros é maior devido a regiões que não são acessadas a partir do teclado (usando o Tab ou teclas de atalho). A nota baixa do Lighthouse se refere exatamente à ausência de *tabindex* dos elementos da página. A maior parte dos erros do WAVE, por exemplo, se referia a *labels* sem descrição (3), links vazios ou botões sem ações. Os alertas se referiam a elementos visualmente pequenos (ícones). O teste de inspeção somente com o teclado e o leitor de

[2]https://support.microsoft.com/pt-br/windows/guia-completo-do-narrador
[3]https://wave.webaim.org/
[4]https://accessibilityinsights.io/
[5]https://developers.google.com/web/tools/lighthouse
[6]https://bugzilla.mozilla.org/show_bug.cgi?id=1710358
[7]https://tinyurl.com/c2f3hzdn

tela (Narrador) mostrou saltos de navegação, regiões inacessíveis sem o uso do mouse e perda de contexto ao selecionar opções de envio de parâmetros de uma requisição. Isso é confirmado pelo rastro de navegação do Accessibility Insights (o Menu da esquerda é inacessível mesmo após 30 *tabs*) como mostrado na Figura 8. Esses resultados corroboram com a inacessibilidade do Postman que já foi inclusive reportada por meio de uma *issue* ainda não resolvida [8].

A página do **Bugzilla**, contém 14 problemas de texto alternativos faltantes conforme o WAVE. O Accessibility Insight detectou também os elementos sem texto alternativo e alguns *ids* duplicados (que podem dificultar a navegação). o Lighthouse mostrou a falta de acessibilidade dos nomes de componentes e identificou os campos sem texto alternativo. Apesar disso, o teste de inspeção com o teclado permitiu acesso à toda região da página e a leitura dos elementos-chave pelo Narrador. No caso do **Firebase**, os três checadores identificaram 22 elementos sem *label*. A maior parte deles compõem a árvore JSON armazenada pelo Realtime Database (Figura 9). Ao ligar o Narrador, percebe-se que a página possui um recurso de acessibilidade para saltar diretamente para o conteúdo. Apesar da aparente conformidade com os padrões, alguns elementos da página estão com a descrição inglês embora a interface esteja em português indicando que não há a tradução dos elementos alternativos. Além disso, percorrer a árvore com o Narrador é bem complexo e não se tem noção da estrutura da mesma, a leitura e edição dos valores dos campos são difíceis de ser feitas somente com o teclado. Por fim, 37 erros foram apontados pelo WAVE na avaliação do **Jira**. Links vazios que atrapalham o entendimento da navegação e problemas de contraste que dificultam a leitura foram encontrados. O Accessibility insights e Lighthouse apontaram problemas na região central (que descreve uma *issue*) tanto de contraste como ids duplicados. Mesmo assim, a exploração por teclado mostrou o alcance de todas regiões e a leitura correta dos elementos.

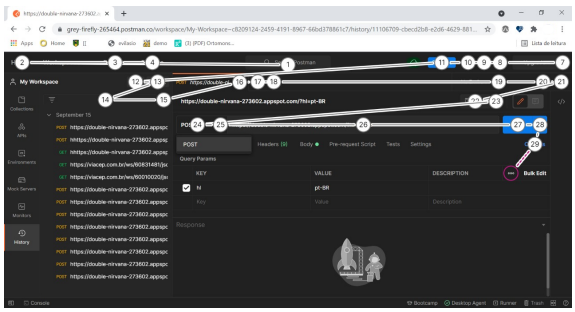

Figura 8. Teste de navegação no Postman.

5. Discussão

Este trabalho preliminar mostrou que o ferramental envolvido no ciclo de desenvolvimento de um software ainda apresenta barreiras que afetam a produtividade dos desenvolvedores com deficiência visual. Além disso, essas dificuldades acontecem em diversos contextos de atuação, como no mercado de trabalho, na área científica ou em projetos pessoais. Os desenvolvedores com deficiência visual precisam acessar os mesmos recursos que outros desenvolvedores têm a disposição, para que também estejam integrados de forma satisfatória aos modelos atuais de desenvolvimento de software.

[8] https://github.com/postmanlabs/postman-app-support/issues/3121

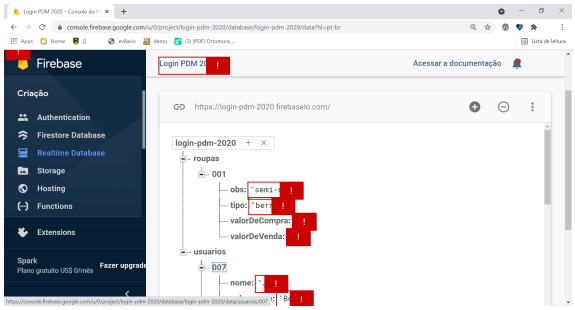

Figura 9. Teste de acessibilidade do Realtime Database

Um dos primeiros problemas abordados neste trabalho foi a dificuldade durante a **edição de código**, que diz respeito tanto à criação como à manutenção de um software já existente. A dificuldade em posicionar itens em layouts foi um dos problemas mais apontados. Uma sugestão é a de que modelos físicos de uma interface possam ser criados para que o desenvolvedor tenha em mente um modelo e saiba onde cada item possa ou deva estar posicionado para que, mesmo utilizando o modo de texto para editar essa interface, a lógica de funcionamento seja bem entendida pelo programador. Outros problemas pertinentes apresentados se referem ao recuo de texto nos códigos (indentação) e à dificuldade em navegar entre arquivos de um projeto. Quanto ao código, por vezes é lido por leitores de tela, que acabam em alguns momentos (ou em todos) não dizendo para o usuário que a linha a ser lida sofreu um recuo em relação à borda do editor. Assim como já existem para o Eclipse e Visual Studio, recursos de acessibilidade precisam ser desenvolvidos. Eles devem ser melhor integrados com os leitores de tela, possuindo pequenos retornos sonoros que indiquem o recuo de texto e a mudança entre diferentes blocos de código.

Já quando o problema de **navegação dentro do ambiente** é considerado, o qual é corriqueiro tanto na codificação como para a depuração, seria interessante que o usuário tomasse conhecimento de atalhos que forneçam uma navegabilidade básica dentro do programa. Sugere-se que as IDEs e ferramentas possam fornecer uma breve introdução logo no primeiro uso do programa com um pequeno conjunto de atalhos que permitam ao usuário navegar por todo o ambiente. E em cada seção, é importante a presença da descrição de um item na tela e a combinação de teclas que permitem que a tarefa seja acionada, o que proporcionará ao desenvolvedor a expansão do seu conhecimento sobre a operação do programa. O **versionamento**, por sua vez, não parece ser a maior fonte de dificuldades para a maioria dos entrevistados, visto que as respostas registradas representam sempre as dificuldades de pelo menos dois dos participantes e no máximo 4 deles. Dito isso, há a importância de saber se esse resultado é decorrente da conformação com o versionamento integrado oferecido pelos ambientes ou se essas ferramentas são preteridas pelos desenvolvedores com deficiência visual em detrimento de ferramentas externas, de apresentação aparentemente mais simples e mais fáceis de operar, além de uma melhor integração com leitores de tela.

Com relação à **análise de acessibilidade por inspeção**, os resultados iniciais indicam a presença de problemas graves tanto para os desenvolvedores com baixa visão (tamanhos de ícones, contraste), como a ausência de descrição alternativa dos elementos e erros de *tabindex* que impedem a navegação pelo teclado. Outro dado relevante foi a descrição alternativa em uma língua distinta dos elementos de interface, um indicativo que

a parte de acessibilidade não está sendo suportada pelo recurso de internacionalização.

6. Conclusão e Trabalhos Futuros

Este artigo visa despertar o interesse da comunidade de Engenharia de Software brasileira sobre a acessibilidade de artefatos digitais envolvidos no ciclo de desenvolvimento com o intuito de, a médio prazo, propor extensões acessíveis destes artefatos. Duas perspectivas foram adotadas na pesquisa: um questionário foi aplicado com 10 (dez) desenvolvedores com deficiência visual e quatro ferramentas Web foram avaliadas com checadores de acessibilidade. Os resultados iniciais mostram que muitos enfrentam problemas ao usarem ferramentas de escrita, depuração e versionamento de código, e ainda, de implantação das aplicações. Os resultados também indicam a forte presença de barreiras no contexto de desenvolvimento e a falta de conformidade das ferramentas Web analisadas com relação às recomendações do WCAG. É importante salientar que o grupo reduzido de respondentes que efetivamente trabalham no ciclo de desenvolvimento de software (i.e., oito participantes) não permite generalizações sobre os resultados, apenas refletem as dificuldades impetradas pelos respondentes em seu dia a dia de desenvolvimento. Uma pesquisa futura e mais aprofundada, usando entrevistas semi-estruturadas, e com um público maior de desenvolvedores deverá ser realizada. Ela permitirá uma observação mais detalhada da realidade e poderá confirmar os resultados emergentes aqui relatados, além de direcionar esforços para uma maior acessibilidade tanto das IDEs como das ferramentas Web usadas.

7. Agradecimentos

Windson Viana teve a pesquisa parcialmente financiada pelo CNPQ pelos projetos 409184/2021-7 e 314425/2021-7.

Referências

Albusays, K. and Ludi, S. (2016). Eliciting programming challenges faced by developers with visual impairments: Exploratory study. *9th International Workshop on Cooperative and Human Aspects of Software Engineering, Austin, Texas, Estados Unidos*, p. 82-85.

Guide, E. I. U. (2019). Eclipse foundation.

Luque, L., Brandão, L. O., Kira, E., and Brandão, A. A. (2018). On the inclusion of learners with visual impairment in computing education programs in brazil: practices of educators and perceptions of visually impaired learners. *Journal of the Brazilian Computer Society*, 24(1):1–12.

Pandey, M., Kameswaran, V., Rao, H. V., O'Modhrain, S., and Oney, S. (2021). Understanding accessibility and collaboration in programming for people with visual impairments. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1).

Petrausch, V. and Loitsh, C. (2017). Accessibility analysis of the eclipse ide for users with visual impairment.

Potlurt, V., Vaithilingam, P., Iyengar, S., Vithya, Y., Swaminathan, M., and Srinivasa, G. (2018). Codetalk: Improving programming environment accessibility of visual impaired developers. *2018 CHI Conference on Human Factors in Computing Systems, Montreal, Quebec, Canadá*.