Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Escola de Informática Aplicada

ACESSIBILIDADE E PRODUTIVIDADE: UM ESTUDO DE CASO COM A FERRAMENTA DE IA PARA DESENVOLVIMENTO GITHUB COPILOT

MATEUS ESCOVINO LAMBRANHO RAMOS

Orientador
SIMONE BACELLAR LEAL FERREIRA
CAROLINA CHRISTINA DO SACRAMENTO NARDI

Rio de Janeiro, RJ – Brasil
JULHO de 2024

Catalogação informatizada pelo autor

Ramos, Mateus
R74111 ACESSIBILIDADE E PRODUTIVIDADE: UM ESTUDO DE CASO COM A FERRAMENTA DE IA PARA DESENVOLVIMENTO GITHUB COPILOT / Mateus Ramos. -- Rio de Janeiro, 2024.
57

Orientadora: SIMONE FERREIRA.
Coorientadora: CAROLINA NARDI.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2024.

1. Produtividade. 2. Desenvolvedor com Deficiência Visual. 3. Ferramenta de Inteligência Artificial para Desenvolvimento. I. FERREIRA, SIMONE, orient. II. NARDI, CAROLINA, coorient. III. Título.

ACESSIBILIDADE E PRODUTIVIDADE: UM ESTUDO DE CASO COM A FERRAMENTA DE IA PARA DESENVOLVIMENTO GITHUB COPILOT

MATEUS ESCOVINO LAMBRANHO RAMOS

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____

SIMONE BACELLAR LEAL FERREIRA (UNIRIO)

_____

CAROLINA CHRISTINA DO SACRAMENTO NARDI (UERJ | UNIRIO)

_____

SEAN WOLFGANG MATSUI SIQUEIRA (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.
JULHO DE 2024

Agradecimentos

Dizer que consegui sozinho seria mentira, eu consegui isso graças as pessoas que não desistiram de me incentivar e investir em mim. Agradeço a minha família e amigos por me cobrar, às vezes de forma amigável e às vezes não, me ajudando a crescer até quando eu não quero. Agradeço à minha orientadora, prof. Simone, que não desistiu de mim mesmo eu tendo desistido algumas vezes. Agradeço à minha coorientadora Carolina que esteve lá no começo desse trabalho e voltou perto do final para a conclusão. Também agradeço às pessoas que não são mais próximas, mas no percurso estiveram lá para me incentivar.

Agradeço principalmente aos participantes, se não fossem por vocês, literalmente, este trabalho não existiria.

Ainda que tenha sido uma jornada muito longa, e não estou falando apenas da faculdade, este trabalho, que demorei mais do que devia para terminar, foi concluído da forma como quis, então não me arrependo de nada.

RESUMO

Muitas pessoas cegas almejam uma inserção ou já estão inseridas na área de desenvolvimento de software. Atualmente se tornou comum a utilização de ferramentas de inteligência artificial para desenvolvimento. Este trabalho se propõe a analisar a acessibilidade e os impactos na produtividade do Github Copilot, uma das principais ferramentas de inteligência artificial para desenvolvimento, em relação a desenvolvedores cegos. Os resultados obtidos foram analisados para entender se esta ferramenta teria um impacto real e positivo no dia a dia desses profissionais.

Palavras-chave: Acessibilidade. Produtividade. Desenvolvedor com Deficiência Visual. Ferramenta de Inteligência Artificial para Desenvolvimento.

ABSTRACT

Many blind people aspire to be or are already working in software development. Nowadays, the use of artificial intelligence tools for development has become common. This study aims to analyze the accessibility and productivity impacts of Github Copilot, one of the main artificial intelligence tools for development, in relation to blind developers. The results obtained were analyzed to understand whether this tool would have a real and positive impact on the daily lives of these professionals.

Keywords: Accessibility. Productivity. Visually Impaired Developer. Artificial Intelligence Tool for Development.

Índice

Índice de Tabela

Índice de Figuras

1 Introdução

1.1 Motivação

O desenvolvimento de software é uma das áreas que mais cresce no mercado(U.S. Bureau of Labor Statistics, 2016), com isso muitas pessoas cegas podem se interessar em atuar (IBC, 2023). De acordo com Conde (apud Nardi, 2021), a cegueira contempla indivíduos com vários graus de visão residual, podendo ser classificada como total, quando há completa perda da visão ou parcial, abrangendo pessoas com visão restrita a contagem de dedos a curta distância, pessoas que só percebem vultos e pessoas que só possuem percepção e projeção luminosas.

De acordo com o Stack Overflow Developer Survey, o Stack Overflow é uma comunidade online para desenvolvedores compartilharem seus conhecimentos de programação, em 2022 1.7% dos desenvolvedores que responderam a pesquisa se autodeclaram com pessoas cegas (Stack Overflow Developer Survey, 2022). Para comparação, a população brasileira tem por volta de 0,75% de pessoas cegas (Ministério da Saúde, 2010), então proporcionalmente há mais desenvolvedores cegos entre os que responderam à pesquisa do Stack Overflow que pessoas cegas na população do Brasil.

Ferramentas de inteligência artificial (IA) para desenvolvimento incrementam a produtividade na criação de código fonte. O Stack Overflow Developer Survey de 2023 trouxe o quanto a utilização de ferramentas de inteligência artificial para desenvolvimento tem se tornado comuns na área de desenvolvimento de software. A pesquisa mostra que 70% dos participantes utilizam ou vão começar a utilizar as ferramentas no processo de desenvolvimento em 2023 (Stack Overflow Developer Survey, 2023).

Este uso também pode ser do interesse de desenvolvedores cegos, uma vez que essas ferramentas têm se tornado, cada dia mais, quase que obrigatórias dentro da área de desenvolvimento de software. Então entender se essas ferramentas são acessíveis e se causam um incremento de produtividade a desenvolvedores cegos é fundamental para garantir o acesso dessas pessoas à área. Este trabalho tem como objeto de estudo o

Github Copilot, por ser a ferramenta mais amplamente utilizada, aplicando métricas de acessibilidade e produtividade para avaliá-lo.

1.2 Objetivo

O presente trabalho tem como objetivo avaliar a acessibilidade do Github Copilot, com o foco no perfil de desenvolvedores cegos e analisar os impactos na produtividade de forma autoavaliativa.

Para chegar ao objetivo principal foram estipulados os seguintes objetivos intermediários:

- Identificar as barreiras de acessibilidade enfrentadas pelas pessoas cegas para interagir com a ferramenta Github Copilot incorporada em uma IDE;
- Identificar a produtividade de desenvolvedores cegos ao utilizar a ferramenta Github Copilot incorporada a uma IDE;

1.3 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, é desenvolvido da seguinte forma:

Capítulo 2: Referencial Teórico

São apresentados conceitos importantes sobre acessibilidade, expandido para acessibilidade em código-fonte, acessibilidade em ambientes de desenvolvimento integrado e como avaliá-la, produtividade, em conjunto com uma métrica para medi-la, e ferramentas de inteligência artificial para desenvolvimento.

Capítulo 3: Método de Pesquisa

Destrincha as etapas da pesquisa, expondo as escolhas dos materiais que foram utilizados, os conceitos de avaliação de acessibilidade e produtividade para montagem do estudo de caso e seleção dos participantes para o mesmo.

Capítulo 4: Estudo de caso

Apresenta os passos pós e pré a aplicação do estudo de caso, então este capítulo destrincha a aplicação do teste piloto para refinamento do estudo de caso, e a aplicação do estudo de caso.

Capítulo 5: Análise de Resultados

São apresentados os resultados das avaliações do estudo de caso que contemplou a avaliação de acessibilidade, produtividade e as observações dos participantes. Além disso, é feita uma análise sobre as respostas ao formulário de seleção.

Capítulo 6: Conclusões

Neste capítulo são apresentadas as considerações finais e sugestões de possíveis trabalhos futuros sobre o tema abordado neste trabalho

2 Referencial Teórico

Neste capítulo serão apresentados conceitos importantes sobre acessibilidade, expandido para acessibilidade em código-fonte, acessibilidade em ambientes de desenvolvimento integrado e como avaliá-la, produtividade, em conjunto com uma métrica para medi-la, e ferramentas de inteligência artificial para desenvolvimento.

2.1 Acessibilidade

Acessibilidade é definida como possibilidade e condição de alcance, percepção e entendimento para utilização, com segurança e autonomia, para pessoa independentemente de suas capacidades físico-motoras, perceptivas, culturais e sociais de participar de todas as atividades (Nicholl, 2001; ABNT, 1994). A acessibilidade digital por sua vez é mais específica, tendo como objetivo dar acesso a recursos da Tecnologia da Informação ou recursos computacionais (FERREIRA & NUNES, 2008). Logo há a necessidade para organizações de adaptarem seus hardwares e softwares para dar a acessibilidade digital necessária para suprir as limitações das pessoas com deficiência (Harrison, 2005) e assim alcançarem mais pessoas.

As organizações que atendem esta necessidade em seus produtos de forma nativa facilitando sua integração com tecnologias assistivas, que são qualquer ferramenta ou recurso destinado a proporcionar habilidades funcionais a pessoas deficientes, ou ampliar as habilidades existentes e, assim, dar-lhes maior autonomia (Enap, 2007; Bersch, 2008). Um exemplo de tecnologias assistivas para pessoas cegas são os softwares denominados “leitores de tela” (screen readers) associados a outros programas chamados de “sintetizadores de voz” que interpretam informações escritas no monitor como páginas web ou interfaces visuais de um software e geram retorno em formato auditivo (FERREIRA et al., 2007).

Pelo fato da visão ser o principal meio de interação com sistemas, por melhor que seja o projeto da interface gráfica ela não é pensada como um modelo conceitual para usuários que têm deficiência visual e sempre será uma barreira para eles (Jacko apud FERREIRA & NUNES, 2008). Então as interfaces devem ser projetadas de jeito que quando

acessadas por uma tecnologia assistiva continuem fornecendo uma interação “amigável” (FERREIRA et al., 2007). As interfaces devem fornecer sequências simples e consistentes de interação, mostrando claramente as alternativas a cada passo, sem confundir nem deixar o usuário inseguro; o usuário deve poder se fixar somente no problema que deseja resolver (Leal Ferreira, 2003).

2.1.1. Acessibilidade em Código-Fonte

Os códigos-fonte, conjunto de instruções e declarações escritas por um programador usando uma linguagem de programação de computador (ROUSE, 2017), nem sempre seguem os preceitos de acessibilidade. Para ser um desenvolvedor é necessário a compreensão do código-fonte do sistema que está sendo criado, corrigido ou adicionando novas funcionalidades. Qualquer impedimento à compreensão do programa tem o efeito de reduzir a acessibilidade ao desenvolvimento de software (Armaly et al, 2016).

O processo para o desenvolvedor com visão é feito através de uma inspeção no código à procura de palavras chaves (Starke et al, 2009), como por exemplo invocação de funções, que o ajudam a compreender o código-fonte (Armaly et al, 2016).

O estado da prática para um desenvolvedor cego utilizando um leitor de tela é navegar pelo código fonte uma linha por vez para cima ou para baixo para compreender o código-fonte. O procedimento é o seguinte: primeiro, o programador abre um arquivo de código-fonte em um Ambiente de desenvolvimento integrados (IDE) e pressiona uma tecla para ler a linha onde o cursor está localizado, por exemplo, “float f é igual a cinco vírgula dois três ponto e vírgula” sendo um processo viável, mas impõe uma alta carga cognitiva ao programador, isso pode causar dificuldade em compreender rapidamente as relações estruturais (Albusays et al, 2017). As ferramentas usadas por ambos, os IDEs, focam em “iluminar” as palavras chaves na sua interface e outros efeitos na sintaxe, dando um retorno visual.

2.1.2. Acessibilidade em Ambientes de Desenvolvimento Integrado

Ambientes de desenvolvimento integrado (IDEs) são softwares que integram editor de texto, gerenciamento de arquivos, compilador e outras ferramentas para promover um fluxo de trabalho eficiente para programadores (Albusays et al, 2017).

Atualmente os IDEs são uma das principais ferramentas de desenvolvimento moderno (Stack Overflow, 2022), facilitam a criação e o trabalho em códigos-fonte extensos pois seus editores de texto incorporam recursos visuais, como recuo para indicar o nível de escopo, cores diferentes para sintaxe e vários outros recursos para ajudar os programadores com visão a entenderem a estrutura de códigos-fonte e navegar por elas com mais facilidade (Mealin Murphy-Hill, 2012).

Não há acessibilidade nos facilitadores citados. Na maioria das vezes as informações transmitidas através de metáforas visuais de IDEs não podem ser interpretadas por leitores de tela. E as funcionalidades avançadas de “localização/pesquisa” para o desenvolvedor se movimentar de forma rápida por códigos-fonte muito extensos não se encaixam com a forma como os desenvolvedores cegos navegam pelo código (Albusays and Ludi, 2016).

2.1.2. Avaliando a Acessibilidade em Ambientes de Desenvolvimento Integrado

Para entender a qualidade da acessibilidade de uma IDE não é tão simples quanto entender a acessibilidade de uma aplicação web, onde há objetivos exatos que são alcançados seguindo uma série de etapas. Uma IDE integra várias ferramentas para desenvolvimento que podem ser utilizadas e configuradas de formas diferentes de acordo com o fluxo de trabalho de cada desenvolvedor. Portanto, o que pode ser utilizado para avaliar a acessibilidade são os desafios gerados ao tentar utilizar essas ferramentas (Potluri et al., 2018).

Potluri et al. (2018) fizeram um levantamento desses desafios, baseando-se em suas experiências como desenvolvedores deficientes visuais e em pesquisas. Agruparam os desafios em quatro categorias: descoberta, visibilidade, navegabilidade e alerta. Estas são detalhadas no capítulo “Método de Pesquisa” na seção "Definição do método de avaliação de acessibilidade".

2.2 Ferramenta de Inteligência Artificial para Desenvolvimento

Dentre os recursos que facilitam a criação e o trabalho das pessoas que desenvolvem códigos-fonte estão as ferramentas de inteligência artificial. Elas auxiliam na escrita de código a partir de sugestões, respondendo dúvidas, construindo funções padronizadas, como CRUD (acrônimo para *Create*, *Read*, *Update* e *Delete*, que correspondem as

quatro operações básicas utilizadas em bancos de dados) ou assinaturas de funções para teste unitários, baseadas no contexto do código-fonte em que estão sendo utilizadas (Chen et al, 2021).

2.2.1. Github Copilot

A mais madura no mercado atualmente é o Github Copilot (Chen et al, 2021). Trata-se de uma extensão para o ambiente de desenvolvimento do Visual Studio Code que oferece sugestões para estender o código-fonte de um programador com base em descrições de problemas e código existentes. Com base no modelo de linguagem em grande escala, Codex, que foi treinada em uma grande quantidade de código-fonte, GitHub Copilot é mais do que uma ferramenta padrão de conclusão de código (que muitas vezes apenas sugere nomes de variáveis ou funções), pois recomenda o código-fonte de funções completas e até sugere casos de teste úteis para funções existentes (Chen et al, 2021). Além disso, de acordo com o Stack Overflow Developer Survey de 2023, o Github Copilot é a escolha geral para a maioria, com 55% dos participantes da pesquisa usando-o. O uso desta ferramenta é cerca de quatro vezes maior que a segunda escolha, com 13% (Stack Overflow Developer Survey, 2023).

2.3 Produtividade

Um dos principais ganhos na utilização das ferramentas de inteligência artificial para desenvolvimento é a produtividade (Chen at al, 2021). Produtividade é a eficiência da produção de bens ou serviços expressa por alguma medida (KALISKI, Burton S, 2001). A produtividade de desenvolvimento descreve o grau de habilidade de programadores individuais ou equipes de desenvolvimento para construir e desenvolver sistemas de software (Ramírez; Nembhard, 2004). Entretanto, a produtividade de desenvolvimento é mais complexa, tanto que após décadas de pesquisa e experiência prática em desenvolvimento, saber como medir a produtividade ou até mesmo definir a produtividade do desenvolvedor permanece indefinido (Forsgren et al, 2021).

Uma forma de tentar solucionar esse problema consiste em aplicar métricas de mais de uma dimensão que cubram os dois métodos principais de avaliação de produtividade: a medição automatizada de características do produto ou processo e relato de produtividade autoavaliada pelos desenvolvedores (Petersen, 2011) assim sendo mais assertivo (Beller et al, 2020). Há a necessidade da aplicação de mais de uma dimensão

pois a avaliação automatizada apresenta falhas como por exemplo a escolha do que será medido podendo está ser problemática e subjetiva (Beller et al, 2020), e a autoavaliação tendo como defeito os preconceitos cognitivos, sendo característicos de qualquer autoavaliação (Murphy-Hill et al, 2019).

2.3.1. Métrica de Produtividade

Um método que abarca estas duas dimensões é o método SPACE, um framework de medição de produtividade que utiliza cinco dimensões para medir a produtividade (Forsgren et al, 2021). Estas 5 dimensões cobrem os dois métodos principais de avaliação de produtividade. Duas delas por medições automatizadas de características, que devem ser utilizadas para avaliação de times de desenvolvimento, e as outras três por autoavaliação dos desenvolvedores (Forsgren et al, 2021).

2.4 Trabalhos Relacionados

Potluri et al (2018) além de mapearem as dificuldades que desenvolvedores com deficiência visual enfrentam utilizando Interfaces gráficas de utilizador (GUI), propõem e implementam uma modificação (plugin) a uma IDE que soluciona partes destas dificuldades.

Continuando nas dificuldades existentes nos IDEs para desenvolvedores cegos tem-se o trabalho de Albusays et al (2017), o qual faz um levantamento de doze funcionalidades que são facilitadoras para a compreensão e navegação no código fonte para desenvolvedores com visão. Evidenciando que muitos IDEs que são restritos a desenvolvedores com visão não fornecem suporte suficiente para que desenvolvedores cegos.

Ao pensar em soluções para as dificuldades em IDEs há o estudo de Albusays e Ludi (2016) que propuseram pistas auditivas para facilitar o desenvolvedor com deficiência visual a navegar e compreender códigos fonte. Foi testado o impacto que as sugestões de áudio teriam, avaliando-se o código fonte escrito e analisados pelos participantes. Eles concluíram que spearcon, sons gerados a partir de fala acelerada, foi o que gerou melhores resultados e foi melhor aceito pelos participantes.

Ferramentas de inteligência artificial para desenvolvimento podem gerar blocos de código com um nível de assertividade que possibilita serem integradas no fluxo de

desenvolvimento comercial, podendo gerar um ganho na produtividade. Para validar este ponto, o trabalho de Ziegler et al (2022) fez uma pesquisa entre os usuários do GitHub Copilot sobre seu impacto em suas produtividades.

Entretanto, nenhum destes trabalhos fez um estudo de caso focado na acessibilidade e na produtividade de desenvolvedores cegos ao utilizar uma ferramenta de inteligência artificial para desenvolvimento integrado a uma IDE. Ao se focar nesta abordagem não explorada há a possibilidade do entendimento de problemas que podem impactar toda uma classe que poderia estar usufruindo destas ferramentas; a contribuição deste trabalho.

3 Método de Pesquisa

Foi definido como método de pesquisa o estudo de caso, o método é utilizado para estudos empíricos para investigar a ocorrência de um fenômeno em um contexto real (FILIPPO et al., 2011).

A pesquisa tem como objetivo validar a acessibilidade e o impacto na produtividade da ferramenta de inteligência artificial para programação Github Copilot para desenvolvedores com deficiência visual.

A presente pesquisa teve as etapas listadas a seguir:

1. Escolha da ferramenta de inteligência artificial para programação.
2. Escolha do ambiente de desenvolvimento integrado.
3. Definição do público-alvo.
4. Montagem do teste.
5. Definição do método de avaliação de acessibilidade.
6. Definição do método de avaliação de produtividade.
7. Seleção dos participantes.
8. Elaboração do estudo de caso
9. Análise dos resultados obtidos no estudo de caso.

3.1 Escolha da Ferramenta de Inteligência Artificial para Programação

O Github Copilot foi escolhido como ferramenta de inteligência artificial para programação por ser alimentado por um descendente de um modelo GPT especializado (Chen et al, 2021) e ser uma das ferramentas de IA mais maduras no mercado. É considerada mais madura que as outras pois foi treinada com casos de repositórios públicos no Github(Sobania et al, 2021) e para várias linguagens existentes nestes. Está disponível como uma extensão no Visual Studio Code, no Visual Studio, no Vim, no Neovim, no pacote de IDEs da JetBrains e no Azure Data Studio. Além do Github Copilot existe também o Tabnine, AWS CodeWhisperer, Synk Code, Codeium e outras ferramentas de inteligência artificial para programação. O Tabnine é a segunda escolha dos desenvolvedores que participaram do Stack Overflow Developer Survey de 2023

sendo cerca de quatro vezes menor que a primeira escolha, o Github Copilot (Stack Overflow Developer Survey, 2023).

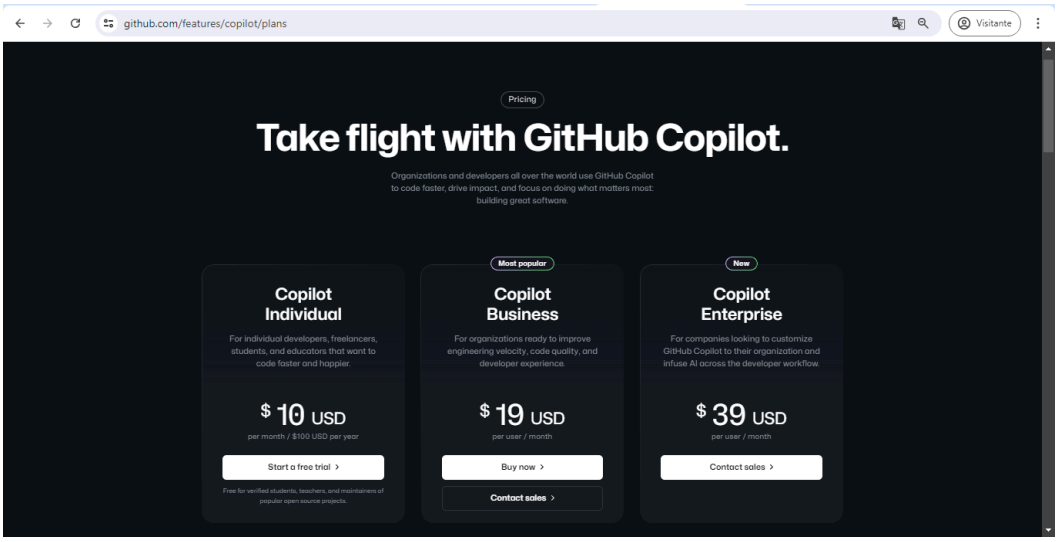

Figura 1: Planos do GitHub Copilot

Fonte: https://github.com/features/copilot/plans

A versão do Github Copilot utilizada foi a para desenvolvedores individuais, freelancers, estudantes e educadores em teste gratuito, todos os preços e planos existentes até a finalização deste trabalho podem ser vistos na figura 1. Esta versão tem as funcionalidades limitadas em comparação com a sua versão empresarial. Segue as funcionalidades presentes nessa versão (Github, 2024) :

-Chat

Incluído:

Mensagens e interações ilimitadas;

Suporte e explicações de codificação com reconhecimento de contexto;

Assistência para depuração e correção de segurança

Não incluído:

Conversas personalizadas para os repositórios da sua organização;

Respostas baseadas na base de conhecimento da sua organização;

Acesso ao conhecimento dos principais repositórios de código aberto;

Análise de diferenças de solicitação de pull;

Pesquisa na Web com tecnologia Bing (beta)

-Conclusão de código

Incluído:

Sugestões de código em tempo real;

Comentários ao código;

Não incluído:

Modelos ajustados (em breve como complemento);

-Ações inteligentes

Incluído:

Sugestões de prompt e bate-papo em linha;

Comandos Slash e variáveis de contexto;

Geração de mensagem de commit;

Não incluído:

Descrição e resumo de solicitação de pull;

-Ambientes suportados

Incluído:

IDE, CLI e GitHub Mobile;

Não incluído:

GitHub.com;

-Gerenciamento e políticas

Incluído:

Filtro de código público

Não incluído:

Gerenciamento de usuários;

Dados excluídos do treinamento por padrão;

Indenização de IP;

Exclusões de conteúdo;

Autenticação SAML SSO;

Requer GitHub Enterprise Cloud;

Para esta pesquisa foram utilizadas as funcionalidades “Mensagens e interações ilimitadas” e “Suporte e explicações de codificação com reconhecimento de contexto” do “Chat”, “Sugestões de código em tempo real” de Conclusão de código, “Sugestões de prompt e bate-papo em linha” de Ações inteligentes e “IDE, CLI e GitHub Mobile” de “Ambientes suportados” para que funcionassem nos ambientes de desenvolvimento integrados que foram utilizados.

3.2 Escolha do Ambiente de Desenvolvimento Integrado

Foi deixado em aberto essa decisão pois depende muito do desenvolvedor já estar ou não habituado com a IDE que utiliza. Em pesquisas envolvendo pessoas com deficiência, testes realizados em seus próprios ambientes, com seus equipamentos conferem mais validade ecológica aos dados coletados (LAZAR et al., 2010).

3.3 Definição do Público-alvo

Os participantes deveriam ter conhecimento mediano ou avançado de lógica de programação, padrões de desenvolvimento de software e familiaridade com um ambiente de desenvolvimento integrado que aceite o plugin do GitHub Copilot.

3.4 Montagem do Teste

O teste foi estruturado em duas fases, a primeira sem o auxílio do GitHub Copilot e a segunda com o auxílio após a explicação das principais funcionalidades do GitHub Copilot. As duas fases contaram com questionários compostos por questões similares e

elaboradas para aplicar conceitos de programação semelhantes. No final de cada fase foram feitas perguntas de autoavaliação referente à acessibilidade e produtividade.

As tabelas 1 e 2 apresentam as questões aplicadas em cada fase:

Tabela 1: Questões da fase um do Teste

Fase 1
Faça uma classe que aplique os conceitos de FIFO, mas com uma fila limitada a três inserções. Com a classe pronta, insira os números 1, 2, 3 e 4 e retorne o resultado após a inserção do 4. Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.
Faça uma método que recebe uma string de e-mails (uma lista dividida por “,”) e retorne uma lista com os e-mails incorretos (utilize Regex). Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.
Faça duas funções: uma que gere uma árvore binária a partir de um JSON e outra que receba esta árvore binária e retorne a ordem de visitação aos nós obtida com uma busca em profundidade (DFS). Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.

Tabela 2:Questões da fase dois do Teste

Fase 2
Faça uma classe que aplique os conceitos de LIFO mas com uma fila limitada a três inserções. Com a classe pronta, insira os números 1, 2, 3 e 4 e retorne o resultado após a inserção do 4. Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.
Faça uma método que recebe uma string de números de celular (uma lista dividida por “,”) e retorne uma lista com os celulares incorretos (utilize Regex). Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.
Faça duas funções: uma que gere uma árvore binária a partir de um JSON e outra que receba esta árvore binária e retorne a ordem de visitação dos nós obtida com uma busca em largura (BFS). Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.

3.5 Definição do Método de Avaliação de Acessibilidade

Para formular a avaliação de acessibilidade do GitHub Copilot foi utilizada a classificação dos desafios de acessibilidade enfrentados pelos desenvolvedores com deficiência visual ao programar utilizando IDEs, proposta por Potluri et al (Potluri et al, 2018). Os autores, com base nos dados de pesquisa de acessibilidade e suas experiências com deficiência visual bem como nos trabalhos relacionados à acessibilidade de IDEs,

classificaram os desafios de acessibilidade nas seguintes categorias e deram alguns exemplos de cenários para cada uma. São elas:

1. Descoberta;

Esta é a capacidade com a qual um desenvolvedor com deficiência visual pode encontrar recursos do sistema para aumentar sua proficiência na IDE ao longo do tempo.

2. Visibilidade;

Esta é a capacidade com a qual um desenvolvedor com deficiência visual pode adquirir informações do sistema que poderia ter deixado passar ou que somente o sistema poderia mostrar para ele de forma discreta sem interromper sua tarefa atual.

3. Navegabilidade;

Esta é a capacidade com a qual um desenvolvedor com deficiência visual pode percorrer pelo código ou pelos vários painéis encontrados na IDE.

4. Alerta.

Esta é a capacidade com a qual um desenvolvedor com deficiência visual pode receber informações que o alertam para problemas que precisam de atenção imediata ou ações em andamento, muito importantes para a depuração e erros de sintaxe.

Como mencionado na etapa 2, a escolha da IDE foi individual para cada desenvolvedor que participou da pesquisa, logo só foram utilizadas as categorias propostas por Potluri somente para avaliar a acessibilidade das novas funcionalidades adicionadas pelo plugin. Estas funcionalidades foram apresentadas antes de aplicar a fase 2 do teste de programação. Por isso não foram incluídas questões referentes às categorias: navegabilidade e alerta, pois não há funcionalidades que elas suportam.

A tabela 3 apresenta as questões aplicadas para a avaliação de acessibilidade e a categoria que a questão representa ou se não foi aplicada:

Tabela 3: as categorias que classificam os desafios de acessibilidade e as questões para avaliação de acessibilidade referente a cada uma delas se aplicada

Descoberta	Conseguiu acessar e utilizar todas as funcionalidades listadas do Github Copilot?

Visibilidade	Conseguiu notar as sugestões para o código propostas pelo Github Copilot?
Navegabilidade	Não foi aplicado neste estudo.
Alerta	Não foi aplicado neste estudo.

As perguntas foram preenchidas enquanto os desenvolvedores usuários executavam a fase 2 do teste de programação pois há perguntas que não poderiam responder sem o auxílio de alguém com visão para notar representações somente visuais do GitHub Copilot.

3.6 Definição do Método de Avaliação de Produtividade

A produtividade pode ser mensurada e avaliada de várias maneiras dependendo se for para um time ou aplicada de forma individual. Para este estudo foi decidido utilizar o framework SPACE (Forsgren et al, 2021) para avaliar produtividade de forma auto declarativa e individual. O framework é definido em cinco dimensões de avaliação:

1. Satisfação e bem-estar (Satisfaction):

O estado de espírito e o bem-estar físico dos desenvolvedores de software;

2. Desempenho (Performance):

O resultado de um sistema ou processo, sendo o principal indicador a qualidade de saída do produto;

3. Atividade (Activity):

A quantidade de trabalho realizado, medida em termos de seus resultados e ações;

4. Comunicação e colaboração (Communication):

O processo colaborativo e suporte que representa as equipes de desenvolvimento de software;

5. Eficiência e fluxo (Efficiency):

O grau em que os desenvolvedores de software podem progredir em suas tarefas, muitas vezes graças a facilitadores incluídos no processo.

Para a presente pesquisa, o framework SPACE foi adaptado e não considerou todas as cinco dimensões. Não foram consideradas as dimensões: “atividade” e “comunicação e colaboração”.

A tabela 4 apresenta as questões aplicadas para a avaliação de produtividade e a dimensão que a questão representa ou se não foi aplicada:

Tabela 4: as cinco dimensões de avaliação de produtividade do framework SPACE e as questões para avaliação de produtividade referente a cada uma delas se aplicada

Satisfação e bem-estar (Satisfaction)	Sentiu que precisou fazer um esforço menor para solucionar as questões com o auxílio do Github Copilot?
Desempenho (Performance)	Sentiu que o resultado do código obtido foi de maior qualidade com o auxílio do Github Copilot?
Atividade (Activity)	Não foi aplicado neste estudo.
Comunicação e colaboração (Communication)	Não foi aplicado neste estudo.
Eficiência e fluxo (Efficiency)	Sentiu que houveram menor impeditivos para solucionar as questões com o auxílio do Github Copilot?

As perguntas foram preenchidas depois que os desenvolvedores usuários executaram a fase 2 do teste de programação.

3.7 Seleção dos Participantes

Para participação no estudo de caso, foi necessária uma etapa preliminar de seleção dos voluntários. Os participantes já deveriam ter conhecimento de desenvolvimento, não havendo necessidade de ser por formação acadêmica formal, para conseguirem entender e resolver as questões do estudo de caso. Além disso, não havia necessidade de experiência prévia no mercado de trabalho ou idade específica, pois isso não impactaria nos resultados do estudo. O perfil dos participantes deveria seguir os seguintes critérios:

-Ter conhecimento médio ou avançado de programação, algoritmo e estrutura de dados;

-Ser cego;

-Não precisa ter tido uma formação acadêmica formal;

-Não necessita ter experiência prévia no mercado de trabalho;

-Não foi delimitado uma faixa etária para a seleção;

O “Formulário de Seleção dos Participantes” no apêndice A foi o formulário utilizado para seleção, entregue para os participantes em formato de Google Forms. A acessibilidade do formulário no Google Forms foi validada utilizando o NonVisual Desktop Access, que é um leitor de tela livre, aberto e portátil para a Microsoft Windows. O formulário ficou disponível durante 100 dias (entre o dia 17 do mês Março e o dia 25 do mês Junho, do ano de 2024), foi respondido por dezenove pessoas e dentre essas somente dez se enquadraram no critério de seleção e demonstraram interesse em participar de etapas futuras da pesquisa. Na tabela 5 podemos ver os resultados do “Formulário de Seleção dos Participantes”.

Tabela 5: Respostas ao formulário de seleção de participantes; ”D”= Desenvolvedor.

Fonte: Coleta de dados

Participante	**Você tem quantos anos de experiência em programação, algoritmo e estrutura de dados em alguma linguagem de programação?**	**Qual é a linguagem de programação que tem experiência em programação, algoritmo e estrutura de dados?**	**Se você utiliza alguma configuração de Tecnologia Assistiva para programar, qual utiliza?**	**Qual ambiente de desenvolvimento integrado (IDE) utiliza para desenvolver?**	**Se já utilizou ou utiliza o GitHub Copilot, utilizou ou utiliza durante quanto tempo?(Em meses)**
D1	Mais que 5 anos	Intmud e Python.	NVDA configurado para ler 100% dos sinais e recuo de linhas por causa da identação.	VS Code	15
D2	Mais que 5 anos	PHP	NVDA	VS Code	5
D3	Mais que 5 anos	PHP	Leitor de tela NVDA	VS Code	0
D4	Mais que 5 anos	Python	Leitor de tela nvda	vs code	2
D5	Mais que 5 anos	Java	Leitor de tela nvda	vscode intelij	12
D6	Mais que 5 anos	C#, Java, Python, JavaScript/TypeScript	Leitor de telas NVDA	Visual Studio e Visual Studio Code	20
D7	Menos de 5 anos	Python com estrutura de dados e Java para Web	NVDA	IntelliJ e VS Code	8
D8	Menos de 5 anos	Java, python, js, c#	nvda	vs code, visual estudio, IntelliJ	6
D9	Menos de 5 anos	Python	nvda	visual studio code	0
D10	Mais que 5 anos	Delphi	Dosvox e NVDA	Não uso IDE, edito o código fonte no Edivox e compilo. Já usei um pouco o Eclipse e NetBeans.	0

3.8 Elaboração do Estudo de Caso

Estudo de caso é uma investigação aprofundada de uma ou mais situações específicas e as definições da quantidade de casos bem como da unidade de análise fazem parte das etapas do método (LAZAR et al., 2010, PIMENTEL, 2011)

Para a preparação do estudo de caso foram formuladas as questões na "Montagem do Teste" para mimetizar situações que o Github Copilot seria utilizado no dia a dia, pois utilizam conceitos que há necessidade de uma pesquisa para solucionar.

Para avaliar o Github Copilot como ferramenta para solucionar as questões do teste foi aplicado o método de avaliação de acessibilidade definido neste capítulo, sendo uma análise qualitativa focando na avaliação de pontos menos abstratos. Pensada para ser respondida pelo avaliador com o auxílio do participante.

Foi usado também o método de avaliação de produtividade definido neste capítulo, baseado em um framework de avaliação utilizado no mercado para avaliar produtividade (Forsgren et al, 2021). Sendo autoavaliativo, feito pelo participante pelo estudo de caso a ser aplicado de forma individual.

3.9 Análise dos Resultados Obtidos no Estudo de Caso

Durante e após a aplicação e resolução das duas fases de testes pelos usuários desenvolvedores, foram respondidas as perguntas referentes a acessibilidade e produtividade.

As questões de acessibilidade foram respondidas pelos desenvolvedores e avaliaram se eles venceram os desafios de acessibilidade de "Descoberta" e "Visibilidade" propostos por Potluri et al (Potluri et al, 2018) ao utilizar o Github Copilot, visto durante a resolução da fase de testes ou ao examinar as vídeo-chamadas gravadas com consentimento dos participantes.

As questões de produtividade foram respondidas pelos próprios participantes após a aplicação da segunda fase de testes, pedindo para que eles a comparassem com a primeira fase respondendo perguntas referentes a "Satisfação e bem-estar", "Desempenho" e "Eficiência e fluxo" que fazem parte do framework de avaliação de produtividade SPACE (Forsgren et al, 2021).

A análise dos resultados é exposta com mais detalhes no Capítulo 5.

3.10 Limitações da Pesquisa

Dos dez voluntários que demonstraram interesse em participar das etapas futuras da pesquisa, quatro retornaram à convocação, mas somente três participaram do estudo de caso, pois um participou do teste piloto. De acordo com Nielsen (2000) (Nielsen, 2000) para cobrir 85% dos problemas de usabilidade há necessidade de cinco usuários para avaliação, uma quantidade que não foi alcançada para este trabalho.

4 Estudo de caso

Neste capítulo são detalhadas as etapas do estudo de caso. Para manter o anonimato dos participantes, eles serão referenciados utilizando a coluna “Participante” da tabela 5.

Tabela 6: Respostas ao formulário de seleção de participantes que seguiram para a segunda fase da pesquisa, tabela gerada a partir da tabela 5; ”D”= Desenvolvedor.

Fonte: Coleta de dados

Participante	Você tem quantos anos de experiência em programação, algoritmo e estrutura de dados em alguma linguagem de programação?	Qual é a linguagem de programação que tem experiência em programação, algoritmo e estrutura de dados?	Se você utiliza alguma configuração de Tecnologia Assistiva para programar, qual utiliza?	Qual ambiente de desenvolvimento integrado (IDE) utiliza para desenvolver?	Se já utilizou ou utiliza o GitHub Copilot, utilizou ou utiliza durante quanto tempo?(Em meses)
D1	Mais que 5 anos	Intmud e Python.	NVDA configurado para ler 100% dos sinais e recuo de linhas por causa da identação.	VS Code	15
D3	Mais que 5 anos	PHP	Leitor de tela NVDA	VS Code	0
D5	Mais que 5 anos	Java	Leitor de tela nvda	vscode intelij	12
D6	Mais que 5 anos	C#, Java, Python, JavaScript/TypeScript	Leitor de telas NVDA	Visual Studio e Visual Studio Code	20

4.1 Aplicação do Teste Piloto

Antes de iniciar os testes com os usuários foi realizado um teste piloto com participante D1. Os resultados do teste piloto foram computados para análise, servindo para ajustar as tarefas.

O ideal para um teste piloto é ser aplicado até que não haja mais necessidade de modificação das informações para a aplicação real, do tempo para execução ou equipamento (PRATES & BARBOSA, 2003). Entretanto, como a pesquisa envolve pessoas com deficiência, um teste piloto feito por um ou dois usuários já pode validar as necessidades (LAZAR et al., 2010). Com isso, o teste piloto foi aplicado ao participante D1.

Para o teste piloto foi necessário que o participante se conectasse ao Github Copilot, utilizando uma conta Github criada somente para este trabalho ou uma conta própria do participante, ativado para qualquer uma das contas previamente, na IDE utilizada pelo participante. Máquina e a IDE foram fornecidas pelo próprio participante por já estar configurada para suas necessidades.

Além disso, foi criado um Termo de Consentimento Livre e Esclarecido (TCLE), formulado para o participante declarar que foi informado sobre o objetivo do trabalho e sobre os testes que foram feitos assim concordando ou não em participar, podendo desistir em qualquer momento de participar e aceitar que às informações coletadas fossem utilizadas para o trabalho sem vincular estas diretamente ao participante. O termo de consentimento livre pode ser lido no anexo C. O TCLE foi enviado em conjunto com a apresentação da pesquisa, que pode ser encontrada no anexo B, e o convite para participar dos testes.

O teste piloto teve como objetivo validar o tempo necessário para realização, inicialmente idealizado para 140 minutos, e se as questões elaboradas estavam claras, sem margem para dúvidas ou má interpretação.

O usuário que participou foi o desenvolvedor D1, cujas informações podem ser vistas na tabela 5. Antes de iniciar o teste piloto foi informado ao usuário que tudo que ele pensasse sobre o teste deveria ser dito em voz alta enquanto ele fazia as questões, caracterizando o protocolo verbal Think aloud (pensar alto).

O teste piloto teve duração de 120 minutos, assim validando o tempo inicial idealizado para aplicação. Antes do início do teste piloto, o avaliador perguntou ao participante sobre sua experiência com o Github Copilot. Em resposta, o voluntário explicou que começou a utilizar por exigência da empresa que trabalha.

Durante a aplicação do teste, o usuário explanou que as questões estavam confusas e muito extensas, tornando difícil de conseguir lembrar de todo enunciado, o fazendo perguntar às vezes o que ainda faltava ser feito. Ele pontuou que a terceira questão das duas fases (sobre árvore binária) como um conhecimento que alguém sem uma formação acadêmica não conseguiria resolver e, portanto, foi removida do teste piloto. Segue como ficou às questões das fases após as modificações sugeridas:

Tabela 7: Tabela 1 após as modificações sugeridas no teste piloto

Fase 1
Faça uma classe fila que deve ter: Um atributo array para uma fila de números de tamanho máximo 3; Um método de inserção na fila aplicando o conceito de FIFO. Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.
Faça uma classe de validação de endereço de e-mail que deve ter: Um método para validação de endereço de e-mail; Um método que recebe uma string de endereço de e-mail (uma lista dividida por “,”) e retorne uma lista com os incorretos (utilize Regex). Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.

Tabela 8: Tabela 2 após as modificações sugeridas no teste piloto

Fase 2
Faça uma classe fila que deve ter: Um atributo array para uma fila de números de tamanho máximo 3; Um método de inserção na fila aplicando o conceito de LIFO. Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.
Faça uma classe de validação de números de telefone celular que deve ter: Um método para validação de endereço de email; Um método que recebe uma string de números de telefone celular (uma lista dividida por “,”) e retorne uma lista com os incorretos (utilize Regex). Faça como se estivesse desenvolvendo para um sistema cuja manutenção será feita por mais de uma pessoa.

4.2 Execução das Avaliações

Após o período de disponibilidade para a resposta do “Formulário de Seleção dos Participantes” começou a convocação dos voluntários.

A convocação, o TCLE e a apresentação da pesquisa foram enviados por meio de e-mail para os voluntários. Em caso de não resposta do voluntário foi enviada uma mensagem pelo aplicativo “Whatsapp”, seguida pelo TCLE e a apresentação da pesquisa na mesma conversa pelo aplicativo. Novamente em caso de não resposta do voluntário foram feitas para o número de celular. Os endereços de e-mail para envio e números de celular foram coletados no “Formulário de Seleção dos Participantes”.

Os testes foram realizados por videoconferência, com os participantes em suas máquinas, nas suas residências, e com a IDE escolhida por eles. Houve interrupções em alguns dos testes por instabilidade da internet e por pedidos dos participantes para que fossem repetidos os enunciados das questões.

Antes do início da avaliação de acessibilidade, foi feita a configuração da IDE de cada participante para poder utilizar o Github Copilot. A configuração resumiu-se em instalar as extensões/plugins necessários para utilizar o Github Copilot e logar com uma conta Github, criada para este estudo com o Github Copilot ativado, na IDE do participante. Foi pedido ao participante declarar se estava de acordo ou não em participar da pesquisa e se ele tinha lido e concordava com o TCLE.

A linguagem utilizada por cada participante não vai ser mencionada, pois não impacta a acessibilidade das funcionalidades do Github Copilot. A IDE usada por todos os voluntários foi a VS Code e o leitor de tela, o NVDA.

Para a etapa de avaliação de acessibilidade todas as ações feitas durante as questões da fase 1 foram desconsideradas, pois não foi utilizado o Github Copilot. Essas questões existem para facilitar que os participantes respondam às perguntas referentes à produtividade ao utilizar o Github Copilot.

Avaliações do participante D3

Avaliação de Acessibilidade

Após a configuração e a solução das questões da fase 1, o avaliador perguntou para o desenvolvedor se ele tinha familiaridade com Github Copilot. Como não tinha, foram apresentadas a ele todas as funcionalidades avaliadas do Github Copilot.

Foi lido o enunciado da primeira questão da fase 2 do teste. O participante pareceu não ter lembrado, de início, o conceito de “FILO”. Então o avaliador mencionou o nome mais utilizado em português para a estrutura de dados, que é “pilha”.

Inicialmente, o usuário não conseguiu acessar a funcionalidade “Sugestões de prompt e bate-papo em linha” sem o auxílio do avaliador; nem o avaliador nem o participante entenderam o motivo, pois o suporte de acessibilidade da IDE gera um alerta sonoro como pontuado por outros participantes deste estudo.

Após o auxílio do avaliador, o voluntário começou a utilizar a funcionalidade e deixou claro que ela é bem acessível, como, por exemplo, a tradução de imagens em palavras notadas pelo avaliador quando o participante repetia em voz alta. Mesmo assim, perdeu um pouco a orientação de onde estava na IDE, pois a funcionalidade abre um editor de código dentro do editor de código para validação das sugestões.

Foi lido o enunciado da segunda questão da fase 2 do teste. O participante utilizou a funcionalidade “Sugestões de prompt e bate-papo em linha” sem o auxílio do avaliador, mas se baseando na experiência anterior de como acessar a funcionalidade. Assim solucionou a segunda questão rapidamente, ainda perdendo um pouco a orientação de onde estava na IDE mas, dessa vez, se encontrando mais rápido.

Tabela 9: Avaliação de acessibilidade do participante D3
Fonte: Coleta de dados

Questão de acessibilidade	Resultado
Conseguiu acessar e utilizar todas as funcionalidades listadas do Github Copilot?	Não
Conseguiu notar as sugestões para o código propostas pelo Github Copilot?	Não

Para este voluntário, o GitHub Copilot não teve bons resultados nas questões de acessibilidade, como mostra a tabela 8. O participante só conseguiu acessar uma funcionalidade com o auxílio do avaliador. Ao fim da avaliação, o participante foi questionado sobre o porquê não utilizou as funcionalidades do "Chat" e ele respondeu que foi por ter um acesso muito burocrático, havendo a necessidade de sair da área do editor de código e voltar depois. A figura 2 ilustra a interface do VS Code, para facilitar o entendimento da movimentação do voluntário dentro da IDE. A área do editor de código e o chat estão marcadas em amarelo, sendo a da esquerda o chat e a da direita a área do editor de código . Além disso, não conseguiu encontrar a funcionalidade "Sugestões de código em tempo real".

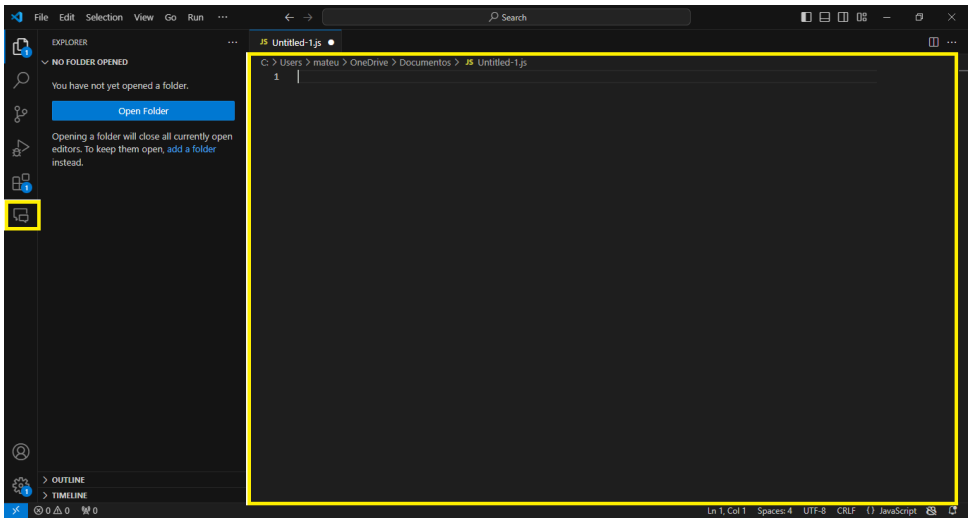

Figura 2: Captura de tela do VS Code, com destaque às áreas do editor de código e do chat.

Avaliação de Produtividade

Tabela 10: Avaliação de produtividade do participante D3
Fonte: Coleta de dados

Questão de produtividade	Resultado
Sentiu que precisou fazer um esforço maior, menor ou igual para solucionar as questões com o auxílio do Github Copilot?	Maior
Sentiu que o resultado do código obtido foi de maior, menor ou igual qualidade com o auxílio do Github Copilot?	Igual
Sentiu que houveram maiores, menores ou iguais impeditivos para solucionar as questões com o auxílio do Github Copilot?	Igual

Para este participante, o GitHub Copilot não teve bons resultados nas questões de produtividade, como mostra a tabela 9. Ao responder à primeira questão de produtividade, o voluntário pontuou que seu esforço deve ter sido maior por ter pouca experiência com ferramentas. Na segunda questão, ele afirmou que não houve grande diferença das soluções que encontrou para as do GitHub Copilot. Para a última, disse que os impeditivos só mudaram, sendo que o principal foi sua pouca experiência com a ferramenta.

Avaliações do participante D5

Avaliação de Acessibilidade

Após a configuração e a solução das questões da fase 1, o avaliador perguntou ao desenvolvedor se ele tinha familiaridade com o GitHub Copilot. Como ele tinha, não houve a necessidade de apresentar todas as funcionalidades avaliadas do GitHub Copilot.

Foi lido o enunciado da primeira questão da fase 2 do teste. O participante pareceu não ter dúvidas sobre os conceitos cobrados, mesmo assim, o avaliador mencionou o nome mais utilizado em português para a estrutura de dados cobrada na questão.

O usuário completou a primeira questão da fase 2 do teste utilizando a funcionalidade "Sugestões de prompt e bate-papo em linha" sem o auxílio do avaliador. O avaliador, então, utilizou essa oportunidade para entender como o voluntário recebia o aviso do comando utilizado para abertura da caixa de texto da funcionalidade.

O participante explicou que a IDE gera um alerta sonoro como se fosse um "Warning" (aviso) padrão da própria IDE, semelhante ao gerado quando há um erro de sintaxe no código-fonte. Esse mesmo aviso sonoro também alerta sobre as "Sugestões de código em tempo real".

Foi lido o enunciado da segunda questão da fase 2 do teste. O participante utilizou novamente a funcionalidade "Sugestões de prompt e bate-papo em linha" sem o auxílio do avaliador, gerando código-fonte comentado. Durante a resolução, também foram propostas soluções por meio da funcionalidade "Sugestões de código em tempo real".

Durante toda a aplicação do teste, o desenvolvedor pediu para que as questões fossem repetidas mais de uma vez. Ao final, o voluntário apontou que estava bem cansado, pois o horário agendado para o teste foi às nove da noite, e a aplicação terminou por volta da meia-noite.

Tabela 11: Avaliação de acessibilidade do participante D5
Fonte: coleta de dados

Questão de acessibilidade	Resultado
Conseguiu acessar e utilizar todas as funcionalidades listadas do Github Copilot?	Sim
Conseguiu notar as sugestões para o código propostas pelo Github Copilot?	Sim

Para este voluntário, o GitHub Copilot teve resultados próximos do ideal nas questões de acessibilidade, como mostra a tabela 10. O participante conseguiu acessar todas as funcionalidades. Ao fim da avaliação, o usuário foi questionado sobre as funcionalidades do "Chat". Como ele informou que sabia utilizá-las, mas não as utilizou durante o teste, respondeu que foi porque não eram de fácil acesso.

Avaliação de Produtividade

Tabela 12: Avaliação de produtividade do participante D5
Fonte: Coleta de dados

Questão de produtividade	Resultado
Sentiu que precisou fazer um esforço maior, menor ou igual para solucionar as questões com o auxílio do Github Copilot?	Menor
Sentiu que o resultado do código obtido foi de maior, menor ou igual qualidade com o auxílio do Github Copilot?	Igual

Sentiu que houveram maiores , menores ou iguais impeditivos para solucionar as questões com o auxílio do Github Copilot?	Menor

Para este participante, o GitHub Copilot teve bons resultados nas questões de produtividade, como mostra a tabela 11. Ao responder à primeira questão, ele pontuou que se sentiu cansado durante o teste devido ao horário, como mencionado anteriormente, e que a ferramenta foi de grande ajuda. Na segunda questão, afirmou que a forma como o código foi gerado, com comentários excessivos, não o auxiliou como desenvolvedor cego. Para a última questão, afirmou que a ferramenta foi muito útil na resolução, pois envolvia um conhecimento muito específico.

Avaliações do participante D6

Avaliação de Acessibilidade

Após a configuração e a solução das questões da fase 1, o avaliador perguntou ao desenvolvedor se ele tinha familiaridade com o GitHub Copilot. Ele tinha, além de afirmar que utilizava há muito tempo. Então, não houve necessidade de apresentar as funcionalidades avaliadas do GitHub Copilot.

Foi lido o enunciado da primeira questão da fase 2 do teste. O participante pareceu não ter nenhuma dúvida sobre os conceitos cobrados falando em voz alta o nome em português para a estrutura de dados.

O usuário completou a primeira questão da fase 2 do teste utilizando a funcionalidade "Sugestões de prompt e bate-papo em linha", mas por possuir muita experiência com a ferramenta, tentou utilizar somente um comentário, uma das formas para ativar a funcionalidade. Infelizmente não teve resposta do GitHub Copilot, então mudou quase que imediatamente para a caixa de texto padrão.

Após gerar a solução, o participante a aplicou ao código-fonte para revisar, afirmando que prefere assim pois é mais fácil para revisar o código. No meio do processo de revisão encontrou um problema em um trecho de código e ativou novamente a "Sugestões de prompt e bate-papo em linha" para regerar o trecho. Este comportamento se difere dos outros avaliados, que modificaram manualmente quando tiveram o mesmo tipo de problema.

Foi lido o enunciado da segunda questão da fase 2 do teste. O participante utilizou novamente a funcionalidade "Sugestões de prompt e bate-papo em linha" igual ao participante D5, mas o prompt foi completo, o que fez com que o Github Copilot sugerisse a solução completa sem necessidade de correção ou modificação para chegar na resposta.

Tabela 13: Avaliação de acessibilidade do participante D5
Fonte: Coleta de dados

Questão de acessibilidade	Resultado
Conseguiu acessar e utilizar todas as funcionalidades listadas do Github Copilot?	Sim
Conseguiu notar as sugestões para o código propostas pelo Github Copilot?	Sim

Para este voluntário, o GitHub Copilot teve resultados próximos do ideal nas questões de acessibilidade, como mostra a tabela 12. Ao fim da avaliação, o participante foi questionado sobre as funcionalidades do "Chat". Ele afirmou que não as utilizava muito para tirar dúvidas graças a sua senioridade na área. O avaliador perguntou se poderia ser por causa do acesso burocrático também, mas o participante explicou que já havia solucionado este problema. Ao invés de navegar por dentro da mesma janela da IDE para acessar as funcionalidades, ele abriu uma janela auxiliar, assim podendo utilizar, nas palavras dele, “a memória muscular do tab” referenciando a comando Ctrl+Tab para viajar entre janelas no Windows.

Ele também mencionou, sem dar muita importância, um pequeno problema com as sugestões constantes da funcionalidade 'Sugestões de código em tempo real': a emissão de sinais sonoros contínuos para cada sugestão no código-fonte.

Avaliação de Produtividade

Tabela 14: Avaliação de produtividade do participante D5
Fonte: Coleta de dados

Questão de produtividade	Resultado
Sentiu que precisou fazer um esforço maior, menor ou igual para solucionar as questões com o auxílio do Github Copilot?	Menor
Sentiu que o resultado do código obtido foi de maior, menor ou igual qualidade com o auxílio do Github Copilot?	Igual
Sentiu que houveram maiores , menores ou iguais impeditivos para solucionar as questões com o auxílio do Github Copilot?	Menor

Para este participante, o GitHub Copilot teve bons resultados nas questões de produtividade, como mostra a tabela 13. Ao responder à primeira questão, ele mencionou brevemente a última questão da fase dois do teste, na qual a proposta de solução da ferramenta forneceu a resposta completa. Na segunda questão, ele mencionou a primeira questão da fase dois do teste, que seguiu uma linha de raciocínio idêntica à que ele utilizou na primeira questão da fase um do teste. Para a última questão, ele novamente usou a primeira questão da fase dois do teste como exemplo, pois quando resolveu a primeira questão da fase um do teste, perdeu alguns minutos testando por não chegar à solução de imediato, o que não aconteceu com o auxílio da ferramenta.

5 Análise de Resultados

Neste capítulo são apresentados os resultados das avaliações do estudo de caso que contemplou a avaliação de acessibilidade, produtividade e as observações dos participantes. Além disso, análise sobre as respostas ao formulário de seleção.

5.1 Análise das Respostas ao Formulário de Seleção

Mesmo com a baixa adesão à participação do estudo de caso, as respostas ao formulário de seleção compreendem o perfil dos desenvolvedores cegos, público-alvo deste trabalho.

Como observado na tabela 14, a maioria dos desenvolvedores utiliza Python seguido pelo Java. Entretanto, muitos deixaram claro que essas não eram as únicas linguagens que dominavam ou que utilizavam em seu local de trabalho. A tabela 14 apresenta uma contagem de quantas vezes cada linguagem foi mencionada pelos desenvolvedores.

Tabela 15: Linguagens de programação mencionadas e quantidade de menções na tabela 5
Fonte: Coleta de dados

Linguagem de programação	Menção
Python	6
Java	4
C#	2
PHP	2
JavaScript/TypeScript	2
Intmud	1
Delphi	1

Na tabela 5 é possível perceber que não existe uma relação entre experiência profissional e a utilização da ferramenta Github Copilot. O desenvolvedor D1 na tabela 5, que participou do teste piloto, afirmou que mesmo tendo muita experiência desenvolvendo, somente começou a utilizar a ferramenta por imposição da empresa que trabalha. Os desenvolvedores D7 e D8 na tabela 5 supostamente são um contraponto,

pois utilizam a ferramenta mesmo com menos tempo de experiência. É suposto pois não houve entrevista para investigar em profundidade a utilização da ferramenta por esses voluntários.

A utilização do NonVisual Desktop Access (NVDA) foi unânime. Além dessa unanimidade, o Visual Studio Code (VS Code) está presente em nove das dez respostas referente a IDE utilizada, os dois sendo gratuitos. A utilização do Github Copilot chegou perto também, dos dez desenvolvedores, somente três ainda não utilizavam a ferramenta, um deles por falta de suporte à linguagem utilizada.

5.2 Análise das Avaliações de Acessibilidade

Para esta avaliação não foi levada em consideração a conclusão das tarefas propostas de forma correta, mas sim se a ferramenta Github Copilot foi acessível para o participante. Por isso, houve necessidade de uma participação mais ativa do avaliador. Este teve a tarefa de documentar, a partir da observação dos participantes, as estratégias para solução das tarefas propostas nos testes, e registrar respostas para as perguntas sobre a acessibilidade das funcionalidades. A Tabela 15 mostra os resultados da avaliação de acessibilidade para cada participante e sua experiência.

Tabela 16: Pergunta 1: Conseguiu acessar e utilizar todas as funcionalidades listadas do Github Copilot?
Pergunta 2: Conseguiu notar as sugestões para o código propostas pelo Github Copilot?
Tempo de Experiência é referente a ferramenta GitHub Copilot em meses.
Fonte: Coleta de dados

Participante	Pergunta 1	Pergunta 2	Tempo de Experiência
D3	Não	Não	0
D5	Sim	Sim	12
D6	Sim	Sim	20

A diferença de experiência na ferramenta pareceu ser o fator determinante nas avaliações de acessibilidade. Notou-se que os participantes D5 e D6, com mais experiência como desenvolvedores, não tiveram a mesma dificuldade para solucionar o teste que D3, sem experiência. Então, a partir dos resultados, mesmo que a maioria dos participantes tenha tido resultados positivos sobre as perguntas sobre acessibilidade, a ferramenta Github Copilot não parece ser realmente acessível para desenvolvedores cegos iniciantes. A ferramenta utiliza soluções que com o tempo tornam-se parte do fluxo de desenvolvimento, mas que necessitam que os desenvolvedores se habituem a

utilizá-la. Os sinais sonoros da funcionalidade “Sugestões de código em tempo real” são um exemplo. Eles não são literais, mas se o desenvolvedor tiver um tempo para analisar e entender, pode se habituar a utilizá-los como um sinalizador da funcionalidade.

A ferramenta utiliza soluções que com o tempo torna-se parte do fluxo de desenvolvimento, mas que necessitam que os desenvolvedores se habituem a utilizá-la. Os sinais sonoros da funcionalidade “Sugestões de código em tempo real” são um exemplo disso, não são literais, mas se o desenvolvedor tiver um tempo para analisar e entender o'que indicam pode se habituar a interpretá-los.

5.3 Análise das Avaliações de Produtividade

Para esta avaliação, não levou-se em consideração a conclusão das tarefas propostas de forma correta, mas o impacto que a ferramenta Github Copilot teve na comparação entre a solução das questões das fases um e dois do teste de acordo com a opinião dos participantes.

Tabela 17: Pergunta 1: Sentiu que precisou fazer um esforço maior, menor ou igual para solucionar as questões com o auxílio do Github Copilot?
Pergunta 2: Sentiu que o resultado do código obtido foi de maior, menor ou igual qualidade com o auxílio do Github Copilot?
Pergunta 3: Sentiu que houveram maiores, menores ou iguais impeditivos para solucionar as questões com o auxílio do Github Copilot?
Tempo de Experiência é referente a ferramenta GitHub Copilot em meses.
Fonte: coleta de dados

Participante	Pergunta 1	Pergunta 2	Pergunta 3	Tempo de Experiência
D3	Maior	Igual	Igual	0
D5	Menor	Igual	Menor	12
D6	Menor	Igual	Menor	20

Novamente a diferença de experiência na ferramenta pareceu ser o principal diferencial, mas dessa vez nas avaliações de produtividade. Pela tabela 16 os participantes com mais tempo de experiência geraram resultados iguais ou positivos de produtividade enquanto os participantes com pouca ou nenhuma os resultados foram iguais ou negativos em relação a produtividade.

Entretanto mesmo para os participantes com pouca ou nenhuma experiência os resultados tendem mais para o igual que para o negativo, logo com o ganho de experiência com a ferramenta esses resultados podem melhorar, como visto nos participantes com mais experiência.

A Pergunta 2, “Sentiu que o resultado do código obtido foi de maior, menor ou igual qualidade com o auxílio do Github Copilot?”, teve resultados iguais para todos os participantes com visto na tabela 16. Os voluntários não notaram uma grande diferença nas soluções propostas pelo GitHub Copilot para as questões da fase 2 em comparação com as soluções dadas pelos próprios participantes na fase 1, sendo este um sinal que as conclusões do GitHub Copilot se assemelham muito as conclusões de desenvolvedores.

5.4 Análise da Relação Entre as Avaliações

Quando analisada a tabela 17 com todos os resultados das avaliações o saldo da ferramenta GitHub Copilot é neutro positivo. Há um ganho de produtividade na maioria dos participantes ao utilizá-la, além de que para esses mesmos participantes é uma ferramenta completamente acessível.

Mesmo assim, o participante D3 não se enquadrou neste saldo, uma vez que este o desenvolvedor, sem nenhuma experiência prévia com a ferramenta, teve um resultado inverso para a avaliação de acessibilidade em relação aos outros dois. Parece que para conseguir utilizar a ferramenta é necessário um período de adequação, o que faz sentido por ela ser voltada para mercado de trabalho e não uma ferramenta para o grande público. Logo, a ferramenta exige conhecimentos específicos, prévios e experiência.

Tabela 18: Pergunta de Acessibilidade 1: Conseguiu acessar e utilizar todas as funcionalidades listadas do Github Copilot?
Pergunta de Acessibilidade 2: Conseguiu notar as sugestões para o código propostas pelo Github Copilot?
Pergunta de Produtividade 1: Sentiu que precisou fazer um esforço maior, menor ou igual para solucionar as questões com o auxílio do Github Copilot?
Pergunta de Produtividade 2: Sentiu que o resultado do código obtido foi de maior, menor ou igual qualidade com o auxílio do Github Copilot?
Pergunta de Produtividade 3: Sentiu que houveram maiores, menores ou iguais impeditivos para solucionar as questões com o auxílio do Github Copilot?
Fonte: Coleta de dados

Participante	Pergunta de Acessibilidade 1	Pergunta de Acessibilidade 2	Pergunta de Produtividade 1	Pergunta de Produtividade 2	Pergunta de Produtividade 3
D3	Não	Não	Maior	Igual	Igual
D5	Sim	Sim	Menor	Igual	Menor
D6	Sim	Sim	Menor	Igual	Menor

Sem um tempo prévio para adequação do participante D3 não há como afirmar se os resultados de produtividade poderiam melhorar. Por ser algo particular, é difícil mensurar quanto tempo e empenho seria necessário para alcançar uma maestria com a

ferramenta igual, próxima ou maior que a dos outros participantes. Possivelmente as métricas de acessibilidade possam ser um indicativo do ponto de experiência necessário para começar a ter um impacto na produtividade. Felizmente para este estudo o impacto na produtividade do participante D3 não foi negativo, tendendo mais para o neutro como visto na tabela 17.

De forma geral foi notado que, para maioria dos casos, o Github Copilot é acessível e gera um ganho de produtividade. Os casos que discordam desta constatação ocorreram por ser a primeira vez em que a ferramenta estava sendo utilizada ou por falta de experiência do participante. Além disso, foram identificados pontos para melhoria que impactaram todos os participantes, alguns deles contornados por participantes com mais experiência na utilização da ferramenta.

Os principais pontos de melhoria observados foram o aviso sonoro da funcionalidade de "Sugestões de código em tempo real", que parece não ser muito intuitivo, ao mesmo tempo que pode ser muito incômodo e a navegação interna da IDE ser muito burocrática, ao mesmo tempo que é a forma padrão de acessar a funcionalidade "Chat".

6 Conclusão

Neste capítulo são apresentadas as considerações finais e sugestões de possíveis trabalhos futuros sobre o tema abordado neste trabalho.

6.1 Considerações finais

Neste trabalho foi analisada a acessibilidade e os impactos na produtividade para desenvolvedores cegos ao utilizarem a ferramenta de inteligência artificial para programação GitHub Copilot, escolhido por este ser um padrão entre os desenvolvedores de acordo com Stack Overflow Developer Survey 2023 (Stack Overflow Developer Survey, 2023).

O objetivo principal foi medir a acessibilidade e o impacto de produtividade de forma autoavaliativa e assim entender se diferente de várias ferramentas dos IDEs esta teria um impacto real e positivo no dia a dia desses profissionais.

Para isto, o método de pesquisa aplicado foi o estudo de caso, duas fases com questões de conhecimento de programação foram criadas e aplicadas aos participantes que se enquadraram no público-alvo. Utilizando trabalhos anteriores foram propostas perguntas para avaliação da acessibilidade do GitHub Copilot em uma IDE e perguntas de autoavaliação de produtividade, para estas foi utilizada a métrica S.P.A.C.E. Para este trabalho a IDE utilizada foi Visual Studio Code por ser a ferramenta utilizada por todos os participantes.

Entretanto houve uma limitação neste trabalho: não participaram desenvolvedores o bastante para chegar a cinco pessoas avaliadas, não cobrindo, de acordo com Nielsen, o mínimo de participantes ou avaliadores necessários para identificação dos 85% dos problemas de acessibilidade na interação com o sistema em avaliação. Este trabalho considerou somente três participantes, diante da dificuldade de recrutar desenvolvedores cegos.

Os resultados de dois dos participantes para as avaliações de acessibilidade foram positivos, mas para o último o resultado foi negativo. Os mesmos participantes com resultados positivos nas avaliações de acessibilidade tiveram resultados neutros positivos nas avaliações de produtividade. Isso se deu diante da constância ou melhora

na produtividade de uma fase de teste para outra. O voluntário com resultado negativo na avaliação de acessibilidade teve resultado neutro negativo na avaliação de produtividade, por ter tido constância ou piora na produtividade.

No formulário de seleção de participantes notou-se uma diferença de perfil para o participante que apresentou resultados negativos: era a sua primeira experiência utilizando o GitHub Copilot. Mesmo assim, os resultados dos outros participantes não podem ser descartados por suas experiências prévias.

Então, ao analisar de forma conjunta os resultados, percebeu-se que o impacto de produtividade não foi negativo. Esta métrica apresentou-se sempre como nula ou positiva, havendo uma melhora com o ganho de experiência no GitHub Copilot, o que faz sentido por ser um ferramenta de trabalho e não uma solução para o grande público.

A acessibilidade pareceu seguir a mesma linha. Na avaliação dos experientes a ferramenta se mostrou acessível e para o que nunca utilizou, não. O formato de metrificar a acessibilidade foi booleano, então não houve pontuação nula. O caso nulo dificilmente existiria numa avaliação de acessibilidade já que não ocorreu mudança na acessibilidade da ferramenta.

O GitHub Copilot apresentou-se como uma ferramenta adequada às pessoas com deficiência neste estudo, mas com pontos para se resolver. Estes pontos para melhoria, impactaram todos os participantes, alguns deles contornaram por experiência na utilização da ferramenta.

6.2 Trabalhos futuros

Este estudo não contou com a participação de muitos desenvolvedores cegos para análise, assim gerando poucos resultados. Uma forma de validar os resultados encontrados seria envolver mais desenvolvedores cegos e repetir a aplicação do estudo de caso, para avaliar se a pouca quantidade de voluntários teve, de fato, impacto nos resultados.

Futuramente pode-se buscar entender a experiência na ferramenta como um diferencial real. Por exemplo, investigar se a experiência realmente aumenta as métricas de produtividade e acessibilidade. Para isso, seria necessário acompanhar um grupo de desenvolvedores com e sem experiência, por um período de tempo, e comparar os resultados em relação a este trabalho.

Também pode ser considerada a utilização da ferramenta em um contexto de complexidade maior de código. Neste trabalho a ferramenta não foi avaliada com intuito de auxiliar um desenvolvedor cego a solucionar problemas como, por exemplo, em um contexto de código fonte já existente de uma aplicação, mas na criação de pequenos trechos de código fonte não muito complexos.

Referências Bibliográficas

ALBUSAYS, K.; LUDI, S. **Eliciting programming challenges faced by developers with visual impairments: exploratory study**. In: INTERNATIONAL WORKSHOP ON COOPERATIVE AND HUMAN ASPECTS OF SOFTWARE ENGINEERING, 9., 2016. New York: ACM, 2016. p. 82-85. Disponível em: https://doi.org/10.1145/2897586.2897616 .

ALBUSAYS, K.; LUDI, S.; HUENERFAUTH, M. **Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges**. In: INTERNATIONAL ACM SIGACCESS CONFERENCE ON COMPUTERS AND ACCESSIBILITY, 19., 2017, New York. New York: ACM, 2017. p. 91-100. Disponível em: https://doi.org/10.1145/3132525.3132550 .

ARMALY, A. **An Empirical Study of Blindness and Program Comprehension**. In: IEEE/ACM INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING COMPANION, 38., 2016, Austin. Piscataway: IEEE, 2016. p. 683-685. Disponível em: https://dl.acm.org/doi/10.1145/2889160.2891041 .

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). N**BR 9050: acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. Rio de Janeiro**, 1994. Disponível em: http://acessibilidade.unb.br/images/PDF/NORMA_NBR-9050.pdf. Acesso em: 22 jul. 2024.

BELLER, Moritz et al. **Mind the gap: on the relationship between automatically measured and self-reported productivity**. arXiv, 2020. Disponível em: https://arxiv.org/abs/2012.07428.

BERSCH, R. **Tecnologia assistiva**. Assistiva, [2024?]. Disponível em: https://www.assistiva.com.br/. Acesso em: 25 jan. 2024.

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

ESCOLA NACIONAL DE ADMINISTRAÇÃO PÚBLICA (ENAP). **e-Mag — modelo de acessibilidade de governo eletrônico**. Brasília, 2007. Curso a distância. Disponível em: https://emag.governoeletronico.gov.br/ .

FERREIRA, S. B. L.; NUNES, R. **e-Usabilidade**. 1. ed. Rio de Janeiro: LTC, 2008.

FERREIRA, S. B. L.; SANTOS, R.; SILVEIRA, D. S. **Panorama da Acessibilidade na Web Brasileira**. Revista de Controle e Administração, v. 3, n. 2, p. 206-235, jul./dez. 2007. Disponível em: https://www.researchgate.net/publication/283051591_Panorama_da_Acessibilidade_na_Web_Brasileira .

FORSGREN, Nicole et al. **The SPACE of Developer Productivity: There's more to it than you think**. Queue, v. 19, n. 1, p. 20-48, 2021. Disponível em https://dl.acm.org/doi/10.1145/3454122.3454124 .

GITHUB. GitHub Copilot. [2024?]. Disponível em: https://github.com/features/copilot/plans. Acesso em: 19 jul. 2024.

HARRISON, S. M. **Opening the eyes of those who can see to the world of those who can't: a case study**. In: SIGCSE — TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 36., 2005, St. Louis. New York: ACM, 2004. v. 37. p. 22-26. Disponível em: https://dl.acm.org/doi/10.1145/1047344.1047368

INSTITUTO BENJAMIN CONSTANT (IBC). **Mercado de Trabalho na área de TI para Pessoas com Deficiência Visual**. YouTube, 22 jun. 2023. 1 vídeo (1h 44min). Disponível em: https://www.youtube.com/watch?v=TQaM-KLBuBA. Acesso em: 22 jun. 2024.

J. Starke, C. Luce and J. Sillito, **Searching and skimming: An exploratory study**, 2009 IEEE International Conference on Software Maintenance, Edmonton, AB, Canada, 2009, pp. 157-166, Disponível em: https://ieeexplore.ieee.org/document/5306335.

KALISKI, Burton S. (ed.). **Encyclopedia of business and finance**. New York: Macmillan Reference USA, 2001. ISBN 0028650654. OCLC 45403115.

LAZAR, J.; FENG, J. H.; HOCHHEISER, H. **Research Methods in Human-Computer Interaction**. John Wiley & Sons, 2010.

MEALIN, S.; MURPHY-HILL, E. **An exploratory study of blind software developers**, 2012. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Innsbruck, Austria, 2012, pp. 71-74,. Disponível em: https://ieeexplore.ieee.org/document/6344485 .

MINISTÉRIO DA SAÚDE (BR). Biblioteca Virtual em Saúde. 13 de dezembro – Dia do Cego. 2010. Disponível em: https://es.wiktionary.org/wiki/removido. Acesso em: 19 jul. 2024.

MURPHY-HILL, E. et al. **What Predicts Software Developers' Productivity?**. IEEE Transactions on Software Engineering, 2019.

NARDI, C. C. S. **Diretrizes para produção de alternativas ao conteúdo visual em mídias sociais sob a perspectiva de pessoas com deficiência visual**. 2021. 439p. Tese (Doutorado em Informática) – Programa de Pós-Graduação em Informática, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2021. Disponível em: http://nau.uniriotec.br/images/pdf/orientacoes/doutorado/2021-tese-carolina-sacramento.pdf

NIELSEN, J. **Why You Only Need to Test with 5 Users**. Nielsen Norman Group, 2000. Disponível em: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. Acesso em: maio 2024.

PETERSEN, K. **Measuring and predicting software productivity: A systematic map and review**. Information and Software Technology, 2011. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0950584910002156 .

PIMENTEL, M. **Sistemas de comunicação para colaboração**. In: PIMENTEL, M.; FUKS, H. (eds.). Sistemas Colaborativos. 1. ed. Rio de Janeiro: Elsevier, 2011. cap. 25. Disponível em: https://sistemascolaborativos.uniriotec.br/wp-content/uploads/sites/18/2019/06/SC-cap5-comunicacao.pdf .

POTLURI, Venkatesh et al. **CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers**. In: CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2018, Montreal. Proceedings... New York: ACM, 2018. p. 1-11. https://doi.org/10.1145/3173574.3174192 .

PRATES, R. O.; BARBOSA, S. D. J. **Avaliação de Interfaces de Usuário - Conceitos e Métodos**. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 23.; JORNADAS DE ATUALIZAÇÃO EM INFORMÁTICA, 22., 2003. Anais... [S. l.: s. n.], 2003.

RAMÍREZ, Y. W.; NEMBHARD, D. A. **Measuring knowledge worker productivity: A taxonomy**. Journal of Intellectual Capital, v. 5, p. 602-628, 2004.

ROUSE, M. **Source Code. Techopedia**, 4 jan. 2017. Disponível em: https://www.techopedia.com/definition/547/source-code. Acesso em: 29 mar. 2024.

SOBANIA, Dominik; BRIESCH, Martin; ROTHLAUFL, Franz. **Choose Your Programming Copilot: A Comparison of the Program Synthesis Performance of GitHub Copilot and Genetic Programming**. arXiv, 2021. Disponível em: https://arxiv.org/abs/2111.07875.

STACK OVERFLOW. **Stack Overflow Developer Survey**. 2022. Disponível em: https://survey.stackoverflow.co/2022. Acesso em: 19 jul. 2024.

STACK OVERFLOW. **Stack Overflow Developer Survey**. 2023. Disponível em: https://survey.stackoverflow.co/2023. Acesso em: 19 jul. 2024.

Apêndice

Apêndice A

Este é o formulário para seleção de participantes para o estudo de caso de Acessibilidade e Impactos de Produtividade do GitHub Copilot, o trabalho de conclusão de curso do aluno Mateus Escovino Lambranho Ramos do curso de Bacharelado em Sistemas de Informação da Universidade Federal do Estado do Rio de Janeiro, orientado pela Dra. Simone Bacellar Leal Ferreira professora do curso de Sistemas de Informação do Departamento de Informática Aplicada da UNIRIO.

Você é uma pessoa com deficiente visual?(Pergunta obrigatória)
o Sim
o Não

As informações requeridas a seguir são para termos o perfil pessoal.

Nome Completo:(Pergunta obrigatória)

E-mail:(Pergunta obrigatória)

Telefone Celular:(Pergunta obrigatória)

Qual o nível da sua deficiência visual?(Pergunta obrigatória)
o Leve
o Moderado
o Acentuada
o Total

Você tem quantos anos de experiência em programação, algoritmo e estrutura de dados em alguma linguagem de programação?(Pergunta obrigatória)
o Menos de 5 anos
o 5 anos
o Mais que 5 anos
o Não tenho experiência

As informações requeridas a seguir são para termos o perfil técnico do participante.

Qual é a linguagem de programação que tem experiência em programação, algoritmo e estrutura de dados?(Pergunta obrigatória)

Se você utiliza alguma configuração de Tecnologia Assistiva para programar, qual utiliza?(Pergunta obrigatória)

Qual ambiente de desenvolvimento integrado(IDE) que utiliza para desenvolver?(Pergunta obrigatória)

Se já utilizou ou utiliza o GitHub Copilot, utilizou ou utiliza durante quanto tempo?(Em meses)(Pergunta obrigatória)

Teria interesse em participar de etapas futuras?(Pergunta obrigatória)
o Sim
o Não

Muito obrigado por responder o formulário entraremos em contato se for de seu interesse participar das etapas futuras desta pesquisa.

Apêndice B

Apresentação da Pesquisa

Prezado colaborador,

Meu nome é Mateus Escovino Lambranho Ramo, sou aluno de graduação na Universidade Federal do Estado do Rio de Janeiro (UNIRIO) e estou realizando um estudo sobre acessibilidade e produtividade na ferramenta de inteligência artificial para programação GitHub Copilot no uso por deficientes visuais, sendo orientado pela Dra. Simone Bacellar Leal Ferreira professora do curso de Sistemas de Informação do Departamento de Informática Aplicada da UNIRIO.

Os dados para a pesquisa serão coletados por questionários e observações durante a realização de questões em dois testes de programação. Com a sua permissão, gravaremos o que acontecerá na tela do computador e nossa conversa. Seu rosto não será filmado e, o mais importante, estamos testando a acessibilidade na ferramenta e não você.

Solicitamos sua colaboração respondendo a algumas perguntas durante e depois dos testes. Isto não tomará mais que duas horas e será uma contribuição importante para a pesquisa sobre esse tema no Brasil. Não há respostas certas ou erradas em relação a qualquer questão dos testes. Os dados de identificação não serão mencionados no relatório da pesquisa, o que preservará o anonimato e sigilo dos respondentes. Se houver necessidade de maiores esclarecimentos, por favor, envie um e-mail para os responsáveis pela pesquisa:

Mateus Escovino Lambranho Ramos: mateus.ramos@uniriotec.br

Simone Bacellar Leal Ferreira: simone@uniriotec.br
CV Lattes: http://lattes.cnpq.br/0926018459123736

Apêndice C

Termo de Consentimento

Prezado colaborador,

Convido você a participar de um estudo sobre avaliação de acessibilidade e produtividade da ferramenta de inteligência artificial para programação.

O estudo ocorrerá da seguinte maneira: você fará dois testes de três questões, o primeiro teste você fará sem o auxílio da ferramenta de inteligência artificial para programação e o segundo você fará com o auxílio (às funcionalidades da ferramenta serão explicadas caso nunca tenha utilizado), ao mesmo tempo que você estiver fazendo o segundo teste serão feitas duas perguntas sobre o auxílio da ferramenta de inteligência artificial para programação e o fim do segundo teste serão feitas três perguntas sobre como você se sentiu sobre a realização das questões com auxílio da ferramenta de inteligência artificial para programação.

O observador estará em uma chamada online com você para fazer a leitura das instruções sobre cada questão e para tirar suas dúvidas. As questões poderão ser gravadas para que os dados possam ser analisados depois.

A sua participação é voluntária. Você pode desistir de participar a qualquer momento, sem sofrer penalidades.

Para garantir sua privacidade, a sua identidade não será revelada. Os resultados do estudo serão divulgados exclusivamente pelo pesquisador e por sua orientadora na literatura especializada ou em congressos e eventos científicos.

Suas dúvidas podem ser esclarecidas a qualquer momento. Basta entrar em contato através do e-mail:
Mateus Escovino Lambranho Ramos: mateus.ramos@uniriotec.br

Declaração de Consentimento

Li as informações contidas neste documento antes de assinar este Termo de Consentimento. Declaro que toda a linguagem utilizada na descrição do estudo foi explicada e que recebi respostas para todas as minhas dúvidas. Confirmo que recebi uma cópia deste Termo de Consentimento. Compreendo que posso me retirar do estudo a qualquer momento, sem sofrer qualquer penalidade.
Dou meu consentimento de livre e espontânea vontade para participar deste estudo.

_____	____/_____/_____
Assinatura do Participante	Data
_____	____/_____/_____
Assinatura do Observador	Data