MELHORANDO A EXPERIÊNCIA DO USUÁRIO EM MARATONAS DE PROGRAMAÇÃO POR MEIO DE SOFTWARES COMPLEMENTARES AO BOCA

Luiz Gustavo Albuquerque dos Santos[1] , Fabíola G. C Ribeiro[2] e Kenia S. de Oliveira[3]

Data de aprovação: 02/12/2024
Data de submissão: 27/11/2024

RESUMO

O BOCA é um sistema de gerenciamento de maratonas de programação amplamente utilizado no Brasil. Este artigo descreve o desenvolvimento e a validação de dois *softwares* de código-aberto complementares ao BOCA: o BOCADE e o BOCA Problems Builder. Esses sistemas foram criados para aprimorar a experiência do usuário (UX) de participantes e organizadores de maratonas de programação, respectivamente. O BOCADE é uma extensão do VS Code que oferece um ambiente de desenvolvimento especializado para uso durante as maratonas de programação. Por outro lado, o BOCA Problems Builder é uma aplicação *web* cujo propósito é facilitar a preparação de problemas para competições. A validação dos *softwares* foi realizada por meio de testes de usabilidade, nos quais foram avaliados diferentes aspectos da UX, como usabilidade, qualidade do *design* visual, utilidade e satisfação. Os resultados, de modo geral, indicam que ambos os sistemas aprimoram a UX de seus respectivos usuários. Portanto, é possível concluir que este trabalho apresenta duas contribuições relevantes para o domínio dos *softwares* destinados à programação competitiva.

Palavras-chave: maratonas de programação; BOCA; UX.

ABSTRACT

BOCA is a programming competition management system widely used in Brazil. This paper describes the development and validation of two open-source software programs complementary to BOCA: BOCADE and BOCA Problems Builder. These systems were designed to improve the user experience (UX) of participants and organizers of programming competitions, respectively. BOCADE is an VS Code extension that offers a specialized development environment for use in programming contests. On the other hand, BOCA Problems Builder is a web application whose purpose is to facilitate the preparation of problems for competitions. The validation of the software was conducted through usability tests, in which different aspects of UX were evaluated, including usability, visual design quality, usefulness, and satisfaction. The results, in general, indicate that both software programs improve the UX of their respective target users. Therefore, it is possible to conclude that this work presents two relevant contributions to the domain of competitive programming software.

***Keywords*:** *competitive programming*; BOCA; UX.

[1] Graduando em Sistemas de Informação no Instituto Federal Goiano - Campus Avançado Catalão. E-mail: gusalbukrk@outlook.com.
[2] Doutora em Ciência da Computação pela Universidade Federal de Uberlândia. E-mail: fabiola.ribeiro@ifgoiano.edu.br.
[3] Doutora em Ciência da Computação pela Universidade Federal de Uberlândia. E-mail: kenia.oliveira@ifgoiano.edu.br.

1 INTRODUÇÃO

Conforme descrito em Xia (2017) e Flórez et al. (2017), a programação de computadores é uma atividade cognitivamente complexa que apresenta um desafio significativo para os alunos em seu aprendizado e para os professores em sua instrução eficaz. Abordagens pedagógicas tradicionais, como exposição teórica e ênfase na sintaxe das linguagens de programação, são prevalentes, mas mostram eficácia limitada no desenvolvimento de habilidades práticas de programação e compreensão aprofundada (CHEAH, 2020). Em contrapartida, uma abordagem prática que consiste em exercícios como desafios de programação é reconhecida por contribuir para a motivação e o desempenho dos alunos (XIA, 2017).

Outra estratégia de ensino comumente utilizada é a gamificação. De acordo com Tenório e Bittencourt (2016), a gamificação é uma metodologia pedagógica em que se utiliza elementos semelhantes a jogos, como contexto, *feedback* rápido, competição e conquistas, para promover o aprendizado e facilitar a resolução criativa de problemas. Diversos autores destacam o potencial da gamificação para aprimorar o aprendizado de programação de computadores, ao tornar o processo de aprendizagem mais dinâmico e engajante (ZHAN et al., 2022; RODRIGUES; ISOTANI, 2023; CARMO NOGUEIRA; SOUZA CAMPOS; FERREIRA, 2018).

A programação competitiva é uma atividade pedagógica de caráter prático que possui elementos de gamificação, como *feedback* rápido, tabelas de classificação e problemas apresentados com narrativas envolventes (MORENO; PINEDA, 2018; COORE; FOKUM, 2019). Outra estratégia de ensino de programação amplamente reconhecida e frequentemente utilizada em maratonas de programação é a codificação colaborativa (SENTANCE; CSIZMADIA, 2017; PHAM; NGUYEN, 2019). Há uma correlação positiva entre a participação em maratonas de programação e resultados desejáveis para os alunos, tais como maior engajamento, desenvolvimento do pensamento computacional e aprimoramento das habilidades de resolução de problemas e programação (AUDRITO; CIOBANU; LAURA, 2023; MORENO; PINEDA, 2018; PIEKARSKI, A. E. et al., 2015; SILVA et al., 2023; YUEN; LIU; LEONG, 2023).

As competições de programação consistem no desenvolvimento de programas de computador para solucionar um conjunto de problemas lógico-matemáticos, nos quais os participantes são avaliados segundo critérios como exatidão do programa, tempo de execução e tempo de desenvolvimento (MAJUMDAR, 2017). Esses problemas são compostos por um nome, uma descrição detalhada, uma explicação do formato esperado para a entrada e a saída, além de exemplos ilustrativos de ambos. Em essência, tratam-se de problemas computacionalmente solucionáveis projetados para serem abordados por meio de algoritmos (CRUZ et al., 2022).

Inúmeras competições de programação são realizadas ao redor do mundo, cada uma com suas particularidades. No Brasil, duas competições proeminentes são a Olimpíada Brasileira de Informática (OBI) e a Maratona de Programação — ambas organizadas pela Sociedade Brasileira de Computação (SBC). A competição de programação da OBI ocorre anualmente desde 1999 e atualmente é dividida em quatro níveis com base no nível de escolaridade dos participantes, variando de júnior, destinado a alunos do ensino fundamental, até sênior, destinado a alunos do primeiro ano de graduação; consiste em três fases subsequentes — local, estadual e nacional (PIEKARSKI, A. E. T. et al., 2023). Em contraste, a Maratona de Programação é uma competição de programação voltada para alunos de graduação e pós-graduação que é realizada desde 1996; consiste em duas fases: regionais e finais, com esta última servindo como parte da classificatória regional latino-americana para as finais mundiais anuais da International Collegiate Programming Contest (ICPC) (MORAIS; RIBAS, 2019). Destaca-se que a ICPC é uma das competições de programação mais renomadas do mundo (ICPC FOUNDATION, 2024). Além das competições vinculadas a eventos renomados, é comum a realização de maratonas meno-

res e não oficiais, cujo objetivo é preparar os participantes para competições oficiais (ALGAR TELECOM, 2012) ou adicionar uma dimensão prática às disciplinas de programação no ensino médio e superior (PIEKARSKI, A. E. et al., 2015; SILVA et al., 2023).

O sistema de gerenciamento de maratonas de programação que tem sido usado na Maratona de Programação desde meados dos anos 2000 é o BOCA Online Contest Administrator (BOCA) (CAMPOS; FERREIRA, 2004; MORAIS; RIBAS, 2019). Esse sistema foi inicialmente desenvolvido especificamente para ser usado nas competições da Maratona de Programação, mas desde então tem sido utilizado em outras competições (ALGAR TELECOM, 2012; PIEKARSKI, A. E. et al., 2015; SILVA et al., 2023). O BOCA atende a dois grupos de usuários que desempenham papéis complementares, porém distintos: os organizadores e os participantes de maratonas de programação. Os organizadores utilizam o sistema para cadastrar problemas e participantes, avaliar as submissões de soluções e esclarecer dúvidas. Por sua vez, os participantes usam o sistema para realizar o *download* das descrições dos problemas, submeter soluções, acompanhar a classificação da competição e sanar dúvidas (CAMPOS; FERREIRA, 2004). Durante a etapa de análise exploratória, foram identificadas duas lacunas no BOCA: a falta de integração com um editor de código e a inexistência de uma ferramenta gráfica para facilitar a criação de pacotes de problemas. A primeira lacuna impacta os participantes de maratonas de programação, enquanto a segunda afeta os organizadores.

Este trabalho é de natureza aplicada, pois tem como objetivo geral o desenvolvimento de *softwares* de código-aberto que visam aprimorar a UX do BOCA. De acordo com Gerhardt e Silveira (2009), pesquisas aplicadas produzem conhecimento com foco em aplicações práticas voltadas para a resolução de problemas específicos. Considerando as lacunas mencionadas no parágrafo anterior, foram desenvolvidos dois *softwares*, cada um projetado para resolver uma das deficiências identificadas. Essas soluções são independentes e, portanto, podem ser utilizadas tanto de forma conjunta quanto separadamente. Assim sendo, as contribuições (C) desse trabalho são as seguintes:

- **C1** - BOCA Development Environment (BOCADE): extensão do Visual Studio Code (VS Code) cujo objetivo é oferecer um ambiente de desenvolvimento especializado para a programação competitiva. A extensão integra todas as funcionalidades necessárias para a participação em maratonas de programação diretamente no editor de código. Elimina-se, assim, a necessidade de utilizar o navegador para interagir com o BOCA ou visualizar arquivos do formato Portable Document Format (PDF).
- **C2** - BOCA Problems Builder: aplicação *web* que visa facilitar a preparação de pacotes de problemas para maratonas de programação. Um pacote de problemas é um arquivo ZIP contendo — entre outros arquivos — um arquivo PDF com a descrição do problema e arquivos de texto com casos de teste que possibilitam a avaliação automatizada de submissões. O desenvolvimento dessa ferramenta justifica-se pelo fato de o recurso nativo de criação de pacotes de problemas do BOCA ser bastante limitado.

A validação dos *softwares* desenvolvidos foi realizada por meio de testes de usabilidade. Para avaliar o BOCADE, dois testes foram conduzidos: um formativo e outro somativo. Por sua vez, a avaliação do BOCA Problems Builder envolveu a realização de apenas um teste, de caráter somativo. Com base nos resultados obtidos nesses testes, foi possível testar a hipótese central deste estudo, segundo a qual os *softwares* desenvolvidos contribuem de forma positiva para a UX de maratonas de programação administradas através do BOCA.

O restante deste texto está organizado da seguinte maneira: na seção 2 é abordada a metodologia utilizada para a realização deste trabalho; na seção 3 é detalhada a revisão da

literatura; nas seções 4 são 5 fornecidas uma visão geral do processo de desenvolvimento e dos testes de usabilidade do BOCADE e do BOCA Problems Builder, respectivamente; e, por fim, na seção 6 é apresentada a conclusão.

2 METODOLOGIA

A metodologia deste trabalho foi dividida em três etapas (E) distintas, conforme descrito a seguir.

- **E1** - Análise exploratória do sistema BOCA e revisão da literatura. O objetivo dessa etapa foi identificar as lacunas do sistema. De acordo com Gerhardt e Silveira (2009), a pesquisa exploratória busca aprofundar a compreensão do problema e, geralmente, vai além de levantamentos bibliográficos. Realizou-se uma revisão de literatura acerca do BOCA e dos conceitos relacionados à UX. Ademais, com o intuito de adquirir experiência prática e obter *insights* relevantes, os autores utilizaram o BOCA em maratonas de programação, desempenhando, em ocasiões distintas, os papéis de participantes e organizadores.

- **E2** - Concepção e desenvolvimento dos *softwares*. Com base no conhecimento obtido na etapa anterior, foi conduzida a fase de elicitação e análise de requisitos para ambos os *softwares*. Em seguida, o desenvolvimento foi executado.

- **E3** - Validação dos *softwares* desenvolvidos através de testes de usabilidade com a coleta de dados quali-quantitativos. Os dados quantitativos foram submetidos a uma análise estatística descritiva e os qualitativos foram analisados por meio da análise de conteúdo. Conforme Gerhardt e Silveira (2009), a análise estatística dos dados tem como objetivo processá-los de forma a torná-los mais compreensíveis, apresentando os resultados por meio de medidas resumo, gráficos e tabelas. Por outro lado, a análise de conteúdo é uma técnica de pesquisa que examina textos de maneira estruturada e objetiva, buscando identificar informações que expliquem como e por que foram geradas, relacionando os significados das palavras ao contexto em que surgiram. É importante ressaltar que os testes de usabilidade, embora sejam pesquisas com seres humanos, não foram submetidos ao comitê de ética devido a falhas no planejamento do cronograma do trabalho, o que impossibilitou a obtenção de aprovação em tempo hábil. Contudo, a ausência de coleta de dados sensíveis, bem como a natureza não invasiva dos testes, torna pouca provável que normas éticas tenham sido violadas.

3 REVISÃO DE LITERATURA

A presente seção tem como objetivo elucidar o contexto no qual o trabalho está inserido, apresentando os trabalhos relacionados e os fundamentos teóricos que embasaram sua realização.

3.1 TRABALHOS RELACIONADOS

Há uma grande variedade de *softwares* relacionados à programação competitiva. Nesta seção são discutidos exclusivamente *softwares* projetados para competições no estilo da ICPC.

De acordo com Maggiolo e Mascellani (2012), os principais desafios técnicos da organização de uma maratona de programação podem ser categorizados em três partes: (1) criação do problema, incluindo todos os seus metadados, isto é, enunciados, casos de teste e soluções; (2) configuração do ambiente do participante, em particular no que diz respeito à consistência do

ambiente e às restrições de rede; e (3) gerenciamento da competição, o que inclui a distribuição de problemas, a classificação automatizada com *feedback* e a atualizações de classificação em tempo real.

Em Maggiolo e Mascellani (2012), os autores apresentaram o Contest Management System (CMS). Esse sistema foi desenvolvido para ser utilizado na Olimpíada Internacional de Informática (IOI) — uma competição de programação semelhante à ICPC, mas destinada para alunos do ensino médio. O Programming Contest Control (PC2) é outro sistema para gerenciamento de competições e tem sido utilizado em algumas competições da ICPC (GANORKAR, 2017). Muitos esforços foram documentados na literatura acadêmica acerca do desenvolvimento de ferramentas complementares para melhorar esse sistema. Por exemplo, Ganorkar (2017) documentou o desenvolvimento do cliente Web Team Interface como uma interface alternativa para o aplicativo de *desktop* e Adithya (2011) desenvolveu uma *sandbox* para aumentar a segurança na execução de código. O sistema de gerenciamento de competições que tem sido cada vez mais utilizado nas competições recentes da ICPC — inclusive nas finais mundiais dos últimos anos — é o DOMjudge (DOMJUDGE, 2024). Em Pham e Nguyen (2019), os autores expandiram as capacidades do DOMjudge adicionando um sistema de detecção de plágio. Na Maratona de Programação da SBC, o BOCA (CAMPOS; FERREIRA, 2004) é usado para o gerenciamento da competição e o Maratona Linux (MORAIS; RIBAS, 2019) para configuração do ambiente do participante.

Assim como os demais sistemas de gerenciamento mencionados no parágrafo anterior, o BOCA fornece acesso ao painel do participante exclusivamente através de um *site*. A única exceção é o PC2, que também pode ser acessado por meio de um aplicativo de *desktop*. Os participantes também utilizam o navegador para visualizar arquivos em formato PDF com as descrições dos problemas. Entretanto, a troca frequente entre aplicações pode ser prejudicial à produtividade (MURTY; DADLANI; DAS, 2022). Ademais, os participantes usam editores de código convencionais para desenvolver as soluções para os problemas da competição. Embora esses editores sejam suficientes, eles não são otimizadas para a programação competitiva, pois não possuem funcionalidades específicas para esse contexto de uso.

A criação de problemas de programação é uma tarefa demorada e propensa a erros (SARSA et al., 2022; ZAVALA; MENDOZA, 2018). Embora seja utilizado predominantemente como um sistema de gerenciamento, o BOCA também oferece um recurso para a preparação de pacotes de problemas. No entanto, esse recurso tem um escopo bastante limitado, pois não abrange tarefas comuns da criação de pacotes de problemas, como a inclusão de vários arquivos de casos de teste e a geração de arquivos PDF. Uma busca por alternativas de terceiros no Google e no GitHub, utilizando combinações das palavras-chave “problema", “maratona"e “BOCA", não identificou nenhuma aplicação gráfica desenvolvida para facilitar esse processo. Entretanto, foram encontradas duas ferramentas de linha de comando: Ejtools (ALVES, 2019) e DS Contest Tools (NUNES, 2024). Contudo, apenas usuários altamente técnicos tendem a utilizar ferramentas de linha de comando, em razão dos problemas de usabilidade inerentes às interfaces baseadas em texto (VAITHILINGAM; GUO, 2019).

Nesta seção, foi apresentado o estado da arte dos *softwares* voltados para competições de programação no formato da ICPC. Como é possível observar, diversas contribuições foram realizadas nesse domínio. No entanto, o levantamento não identificou nenhum trabalho anterior que propusesse um ambiente de desenvolvimento que integrasse todas as funcionalidades necessárias para a programação competitiva dentro de um editor de código ou uma ferramenta gráfica para facilitar a criação de pacotes de problemas para maratonas gerenciadas através do BOCA. Neste artigo, o BOCADE e o BOCA Problems Builder são propostos para preencher essas lacunas.

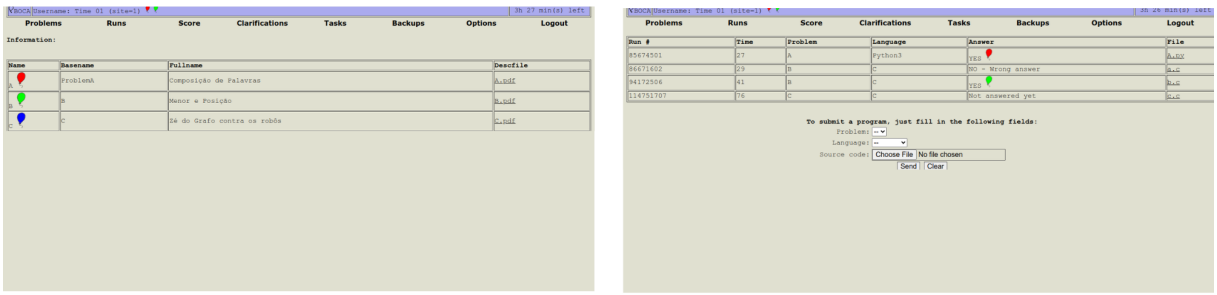
3.2 BOCA

Devido a sua primordialidade para este trabalho, o BOCA é examinado em maior detalhe nesta subseção.

Entre os diversos sistemas de gerenciamento de competições de programação disponíveis, o BOCA se destaca como uma alternativa muito popular no Brasil, possivelmente devido à sua adoção na Maratona de Programação e licenciamento de código aberto (MORAIS; RIBAS, 2019). O BOCA é frequentemente utilizado em conjunto com o Maratona Linux como uma solução de *softwares* complementares para satisfazer as necessidades técnicas de competições de programação. O Maratona Linux é uma distribuição Linux baseada em Ubuntu que integra os *softwares* necessários para competições de programação. Isso inclui editores de código populares (por exemplo, Emacs, Vim, gedit, Geany, CodeBlocks, VS Code, PyCharm, IntelliJ IDEA, Eclipse, CLion), juntamente com compiladores e interpretadores para as linguagens comumente usadas em maratonas de programação (C/C++, Java e Python). Ademais, o Maratona Linux também possui os recursos necessários para impedir qualquer acesso não autorizado à internet ou mídia física (MORAIS; RIBAS, 2019).

As funcionalidades do painel do participante do BOCA estão organizadas em sete abas. Uma breve descrição das funcionalidades fornecidas por cada aba, conforme delineado em Santos (2024), é a seguinte: (1) problemas: informações sobre os problemas da competição; (2) submissões: informações sobre soluções submetidas e formulário para submissão de novas soluções; (3) pontuação: classificação da competição em tempo real; (4) esclarecimentos: informações sobre as dúvidas submetidas e formulário para submissão de novas dúvidas; (5) tarefas: formulários para envio de arquivos para impressão e solicitação de ajuda urgente; (6) *backup*: formulário para *upload* de arquivos para *backup*; e (7) opções: formulário para edição de informações do participante, como nome de usuário, nome completo e senha.

Na Figura 1, são apresentas as interfaces das abas de problemas e de submissões do painel do participante.



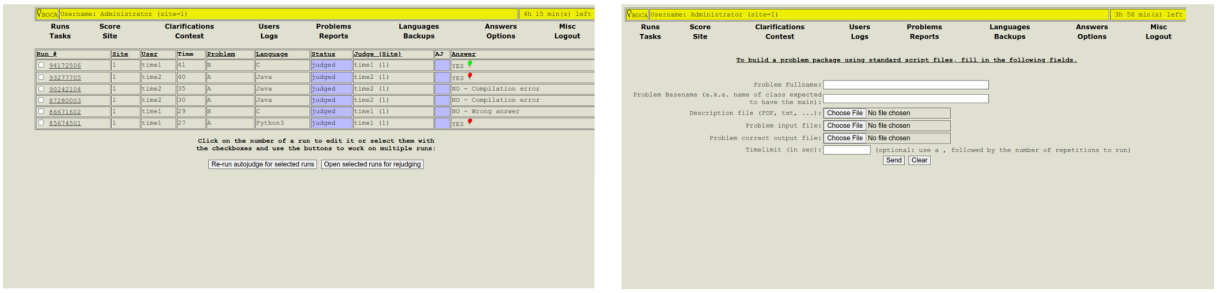
(A) (B)

Figura 1 — As interfaces das abas de problemas (A) e submissões (B) do painel do participante do BOCA. Fonte: os autores.

Por outro lado, o painel do organizador oferece funcionalidades de gerenciamento, tais como: o registro de problemas e participantes, a avaliação de submissões e o esclarecimento de dúvidas. Para o registro de um problema, o BOCA requer o *upload* de um pacote de problemas, que é um arquivo ZIP contendo vários arquivos necessários para a apresentação, execução e avaliação do problema. Os arquivos relacionados à apresentação e avaliação são específicos para cada problema, enquanto os arquivos relativos à execução do código são padronizados em todos os pacotes de problemas. Existem dois arquivos de apresentação do problema: um arquivo de texto chamado *problem.info*, que contém metadados diversos, e um arquivo PDF que fornece a descrição do problema. No que diz respeito aos arquivos relacionados à avaliação

do problema, estes consistem em pares de arquivos de texto de entrada e saída contendo os casos de teste. Durante a avaliação automatizada de submissões, o código será executado tantas vezes quanto houver arquivos de pares de entrada-saída. Cada execução utilizará um arquivo de entrada específico e a saída resultante será comparada ao arquivo de saída correspondente (CAMPOS, 2024).

Na Figura 2, são apresentas as interfaces das abas de submissões e de criação de pacotes de problemas do painel do organizador.



(A) (B)

Figura 2 — As interfaces das abas de submissões (A) e criação de pacotes de problemas (B) do painel do organizador do BOCA. Fonte: os autores.

3.3 UX

No campo de Interação Humano-Computador (IHC), usabilidade e UX são dois conceitos inter-relacionados. Conforme definido pela International Organization for Standardization (ISO) (2018), usabilidade é o grau em que os usuários podem atingir objetivos específicos com eficácia, eficiência e satisfação; enquanto UX abrange como os usuários percebem e respondem a um sistema em um contexto de uso especificado. A sobreposição entre os dois conceitos é evidente. No entanto, UX é geralmente entendida como um conceito mais amplo que abrange usabilidade e todos os outros aspectos da interação, como estética, acessibilidade e satisfação geral (LIMA; MOURA et al., 2022).

Conforme destacado por Lima e Benitti (2021), uma abordagem confiável para obter boas métricas de usabilidade é seguir princípios consolidados da área de IHC, frequentemente organizados em conjuntos de diretrizes, tais como as Heurísticas de Usabilidade, as Regras de Ouro da Usabilidade e os Critérios Ergonômicos. Esses autores compararam vários conjuntos de diretrizes e identificaram seis princípios-chave de usabilidade: (1) tratamento adequado de erros; (2) *design* consistente; (3) *feedback* do usuário sobre ações e eventos significativos; (4) adaptabilidade às necessidades do usuário; (5) redução da carga cognitiva; e (6) controle maximizado do usuário sobre a interface.

Em Riihiaho (2018), o autor fornece uma visão geral acerca de testes de usabilidade — uma técnica de avaliação amplamente difundida, na qual usuários representativos executam um conjunto específico de tarefas para avaliar a usabilidade e, possivelmente, outros aspectos da UX de um sistema, sob a supervisão de um facilitador. Os testes de usabilidade podem ser categorizados em dois tipos principais: formativos e somativos. A avaliação formativa envolve a coleta de *feedback* do usuário ainda durante o desenvolvimento para orientar melhorias iterativas, enquanto a avaliação somativa é conduzida no final do desenvolvimento para avaliar se o produto final atende aos requisitos de usabilidade. As avaliações formativas são normalmente mais curtas e menos abrangentes. Não há consenso entre os pesquisadores sobre o número ideal

#	Item
1	Eu acho que gostaria de usar esse sistema frequentemente.
2	Eu achei esse sistema desnecessariamente complexo.
3	Eu achei esse sistema fácil de usar.
4	Eu achei que precisaria de ajuda de um técnico para ser capaz de usar esse sistema.
5	Eu achei que as várias funções desse sistema foram bem integradas.
6	Eu acho que o sistema apresenta muita inconsistência.
7	Eu imagino que a maioria das pessoas pode aprender a usar esse sistema rapidamente.
8	Eu achei esse sistema muito pesado para usar.
9	Eu me senti muito seguro usando o sistema.
10	Eu precisei aprender muitas coisas antes que pudesse utilizar esse sistema.

Quadro 1 — Questionário SUS em português do Brasil, conforme traduzido e validado por (LOURENÇO; CARMONA; LOPES, 2022).

de participantes para um teste de usabilidade, mas estudos indicam que um grupo de cinco a nove usuários é geralmente suficiente para identificar aproximadamente 80% dos problemas de usabilidade.

Ainda de acordo com Riihiaho (2018), o protocolo *think aloud* e o uso de questionários são dois métodos amplamente empregados para coletas de dados em testes de usabilidade. No protocolo *think aloud*, os participantes são solicitados a verbalizar seus pensamentos enquanto executam as tarefas designadas. Em relação aos questionários, recomenda-se o uso de questionários padronizados, pois eles proporcionam maior confiabilidade. A Escala de Usabilidade de Sistema (SUS, em inglês) emergiu como o questionário padrão para testes de usabilidade. Esta ferramenta é composta por dez itens, apresentados no Quadro 1, os quais devem ser classificados em uma escala Likert de cinco pontos. Os resultados são agregados em um único número, variando de 0 a 100, conhecido como pontuação SUS; essa pontuação serve como um indicador conciso da usabilidade geral de um sistema. De acordo com Lewis (2018), a escala de classificação curva de Sauro-Lewis classifica as pontuações SUS atribuindo notas de letras com base em intervalos de pontuação padronizados, com a nota mais alta (A+) abrangendo 84,1–100 e a nota mais baixa (F) variando de 0–51,6. Uma revisão de trabalhos que apresentam avaliações de usabilidade para sistemas de diversos tipos aponta para uma pontuação média do SUS de 70,8, o que corresponde a uma nota C na escala Sauro-Lewis.

Outra ferramenta padronizada comumente utilizada em IHC é o Inventário de Estética Visual de *Sites* (VisAWI, em inglês) — um questionário de 18 itens usado para medir a qualidade do *design* visual de *sites*. No VisAWI original, há oito itens com formulação negativa. Porém, em (PERRIG et al., 2023), os autores desenvolveram e validaram uma versão inteiramente positiva, denominada VisAWI-Pos, para mitigar possíveis problemas associados a itens formulados negativamente. Os itens do VisAWI-Pos estão listados no Quadro 2.

Ademais, o Modelo de Aceitação de Tecnologia (TAM, em inglês) é uma teoria de sistemas de informação usada para entender a aceitação e o uso de uma tecnologia (CHEAH et al., 2023). De acordo com o TAM, a aceitação de uma inovação tecnológica é baseada em dois fatores principais: utilidade percebida (PU, em inglês) e facilidade de uso percebida (PEU, em inglês). Existem várias versões do modelo TAM e questionários relacionados. Em Davis (1989), o autor propôs dois questionários distintos de seis itens para avaliar separadamente a utilidade e a facilidade de uso. No Quadro 3, são apresentados os itens dessa versão do TAM.

#	Item
1	O *layout* tem uma aparência limpa.
2	O *layout* é fácil de entender.
3	O *layout* tem uma aparência bem estruturada.
4	O *site* tem uma aparência uniforme.
5	Tudo combina neste *site*.
6	O *design* é interessante.
7	O *layout* é inventivo.
8	O *design* tem uma aparência inspirada.
9	O *layout* tem uma aparência dinâmica.
10	O *layout* é agradavelmente variado.
11	A composição de cores é atraente.
12	A escolha de cores é perfeita.
13	As cores combinam.
14	As cores são atraentes.
15	O *layout* parece ter sido projetado profissionalmente.
16	O *layout* é atualizado.
17	O *site* foi projetado com cuidado.
18	O *design* do *site* tem um conceito claro.

Quadro 2 — Questionário VisAWI-POS (PERRIG et al., 2023), traduzido pelos autores devido à ausência de traduções validadas.

#	Item
PU-1	Usar o [nome da tecnologia] me permitiria realizar tarefas mais rapidamente.
PU-2	Usar o [nome da tecnologia] melhoraria meu desempenho no meu trabalho.
PU-3	Usar o [nome da tecnologia] aumentaria a minha produtividade.
PU-4	Usar o [nome da tecnologia] aumentaria minha eficácia no meu trabalho.
PU-5	Usar o [nome da tecnologia] tornaria mais fácil fazer o meu trabalho.
PU-6	Eu consideraria o [nome da tecnologia] útil no meu trabalho.
PEU-1	Minha interação com o [nome da tecnologia] seria clara e compreensível.
PEU-2	Eu acharia o [nome da tecnologia] flexível para interagir.
PEU-3	Eu acharia fácil fazer com que o [nome da tecnologia] faça o que eu quero.
PEU-4	Aprender a operar o [nome da tecnologia] seria fácil para mim.
PEU-5	Seria fácil para mim me tornar habilidoso no uso do [nome da tecnologia].
PEU-6	Eu acharia o [nome da tecnologia] fácil de usar.

Quadro 3 — Questionário TAM (DAVIS, 1989), traduzido pelos autores devido à ausência de traduções validadas.

4 BOCADE

Nesta subseção, é apresentada uma visão geral do processo de desenvolvimento e dos testes de usabilidade do BOCADE.

O ambiente de desenvolvimento padrão do BOCA consiste no uso de duas aplicações distintas: (1) um navegador para a interação com o painel do participante e a visualização de arquivos PDF e (2) um editor de código para a codificação de soluções. O BOCADE é um ambiente de desenvolvimento alternativo que integra todas as funcionalidades essenciais para a programação competitiva diretamente no editor de código. Essas funcionalidades incluem a integração do painel do participante do BOCA e de um visualizador de PDFs, entre outras. O propósito da extensão é o de melhorar a UX de participantes de maratonas de programação.

4.1 REQUISITOS

De acordo com Kurtanović e Maalej (2017), os requisitos de *software* podem ser categorizados em dois tipos distintos: funcionais e não funcionais. Os requisitos funcionais (RF) definem as capacidades específicas que o sistema deve oferecer e os requisitos não funcionais (RNF) descrevem suas propriedades e restrições (como, por exemplo, desempenho, segurança, usabilidade, etc.).

A seguir, são descritos de forma sucinta os requisitos funcionais e não funcionais do BOCADE.

- **C1-RF1**: desenvolvimento do painel integrado do participante do BOCA. As funcionalidades do painel a serem replicadas na extensão são aquelas presentes nas abas de problemas, execuções, pontuação e esclarecimentos. Essa decisão foi tomada porque as funcionalidades contidas nas outras abas são usadas com muito menos frequência. A proposta para esse requisito surgiu da observação de que os participantes de maratonas de programação eram restritos a interagir com o BOCA através de um navegador, apesar de passarem a maior parte do tempo em um editor de código.

- **C1-RF2**: desenvolvimento do visualizador integrado de PDF. Essa funcionalidade se justifica devido à prática prevalente de distribuir as descrições de problemas através de arquivos PDF. Os dois primeiros requisitos funcionais estão relacionados, já que, juntos, eliminam a necessidade do uso de um navegador durante as maratonas de programação. Dessa forma, não é mais preciso alternar entre aplicativos para interagir com o BOCA, uma tarefa que, embora necessária apenas de forma esporádica ao decorrer da competição, poderia comprometer a produtividade dos participantes. Conforme discutido em (MURTY; DADLANI; DAS, 2022), a necessidade de alternar frequentemente entre aplicativos configura em uma desvantagem, pois pode impactar negativamente a produtividade devido ao tempo necessário para que os usuários se ajustem ao aplicativo, seu contexto semântico e sua finalidade.

- **C1-RF3**: desenvolvimento do painel de extração e execução de casos de teste. O intuito dessa funcionalidade é agilizar significativamente a validação do código. Como é feito atualmente, os participantes de uma maratona precisam digitar cada entrada de amostra individualmente e, posteriormente, realizar uma comparação mental entre a saída de amostra e a saída gerada por seu código. A funcionalidade do painel de casos de teste visa automatizar esse processo e é inspirada em uma funcionalidade similar disponível em uma extensão para o VS Code, voltada para maratonas de programação com formato distinto ao da ICPC (AGRAWAL, D., 2020). O painel será composto pelos seguintes

elementos: uma interface para gerenciar (isto é, visualizar, criar, editar e excluir) casos de teste, um botão para extrair casos de teste de arquivos PDF e um botão para executar arquivos de código-fonte contra um conjunto de casos de teste.

- **C1-RF4**: desenvolvimento de um botão de organização do espaço de trabalho. O objetivo desse recurso é automatizar o processo de organização das abas do editor de código e outros elementos de *layout* relevantes de uma forma que otimize o espaço de trabalho para navegação rápida em maratonas de programação. Esse botão, quando acionado, abre o painel de casos de teste e divide a área do editor em dois grupos de editores. As abas do arquivo de código-fonte são movidas para o grupo esquerdo, e as abas da interface dos participante do BOCA e dos arquivos PDF, para o grupo direito. A inclusão dessa funcionalidade é particularmente pertinente porque a integração do painel do participante do BOCA e do visualizador de PDF levará os usuários a manter várias abas do editor de código abertas ao mesmo tempo.

- **C1-RF5**: desenvolvimento de um botão de alternância entre os modos claro e escuro para facilitar a seleção do modo preferido pelo usuário.

- **C1-RNF1**: adesão aos seis princípios-chave de usabilidade identificados na subseção 3.3 da revisão da literatura.

- **C1-RNF2**: criação de uma interface intuitiva e de fácil uso. Ademais, a interface deve ter a mesma estrutura fundamental do painel *web* do BOCA para garantir a familiaridade dos usuários experientes, além de oferecer suporte aos modos claro e escuro nos dois elementos de *layout* personalizados contribuídos pela extensão — isto é, o painel BOCA e o painel de casos de teste.

4.2 IMPLEMENTAÇÃO

Nessa seção, é detalhado o conjunto de soluções utilizado para satisfazer os requisitos do BOCADE. É importante ressaltar que as versões de *software* alvo usadas durante o desenvolvimento da extensão foram: BOCA 1.5.17, Maratona Linux 20231006 e VS Code 1.81.1.

O VS Code emergiu como a plataforma ideal para a extensão devido à sua gratuidade, compatibilidade multiplataforma, extensibilidade e popularidade (VERMA, 2020; STACK OVERFLOW, 2023). Através de sua Application Programming Interface (API), os desenvolvedores de extensões podem personalizar quase todos os aspectos do editor e estender suas funcionalidades para abranger ferramentas personalizadas (UZAYR, 2022). Além disso, o VS Code está incluído por padrão no Maratona Linux e oferece suporte abrangente para todas as linguagens de programação comumente usadas em competições de programação. Isso contrasta com outros editores de código disponíveis no Maratona Linux, que oferecem suporte apenas a algumas linguagens.

TypeScript (MICROSOFT, 2024a) foi escolhido como a linguagem de programação do projeto devido à sua popularidade e à experiência prévia do desenvolvedor com a linguagem. Ademais, as bibliotecas utilizadas na implementação de cada um dos requisitos da aplicação são apresentadas a seguir.

- **C1-RF1**. Um desafio técnico encontrado durante a implementação desse requisito foi a ausência de uma API no BOCA. Esse obstáculo foi superado por meio da utilização de *web scraping*, que é uma técnica comumente empregada quando não há uma API disponível (GLEZ-PEÑA et al., 2014). Um *web scraper* possibilita a extração sistemática

de dados ao simular o comportamento de navegação humana na *web*. Inicialmente, foi considerado acesso direto ao banco de dados, pois essa abordagem permitiria uma recuperação de dados um pouco mais rápida em comparação ao *web scraping*. No entanto, essa abordagem apresentou um problema crítico de percepção do usuário. Embora a API do VS Code ofereça mecanismos para o armazenamento seguro de informações sensíveis, exigir que os organizadores de maratona forneçam suas credenciais de banco de dados diretamente na extensão, pode ser mal interpretado como inseguro. Portanto, a abordagem de *web scraping* foi preferida ao acesso direto ao banco de dados. Devido à sua popularidade e à simplicidade de sua API, a biblioteca jsdom (JSDOM, 2024) foi escolhida como a solução para *web scraping*.

- **C1-RF2**. Não foi necessário implementar uma nova solução para possibilitar a visualização de arquivos PDF dentro do editor de código. Por meio de uma pesquisa realizada no *marketplace* do VS Code, utilizando a palavra-chave "PDF", foi possível encontrar a extensão vscode-pdf (TOMOKI, 2023), a qual disponibiliza um visualizador de PDF integrado. Portanto, essa extensão foi incorporada ao projeto como uma dependência, garantindo sua instalação automática durante a instalação do BOCADE.

- **C1-RF3**. Duas bibliotecas foram utilizadas para satisfazer os requisitos funcionais do painel de casos de teste: pdfplumber, (SINGER-VINE, 2024) para a extração de casos de teste de arquivos PDF, e compile-run (AGRAWAL, V., 2020), para a execução de um arquivo de código-fonte contra casos de teste.

- **C1-RF4** e **C1-RF5**. A implementação dos botões de organização do espaço de trabalho e de alternância do modo claro e escuro dependeu exclusivamente da API de extensão do VS Code, sem a necessidade de bibliotecas adicionais.

- **C1-RNF1**. Os seis princípios-chave de usabilidade identificados na fundamentação teórica foram incorporados à extensão da seguinte forma: (1) o tratamento adequado de erros é assegurado por meio da exibição de mensagens de erro em caso de falha; (2) a consistência nos elementos da interface foi obtida através da utilização de componentes do Webview UI Toolkit; (3) o *feedback* do usuário é fornecido quando ações significativas ocorrem, como mensagens de sucesso após o envio bem-sucedido de formulários; (4) a adaptabilidade às necessidades do usuário é reforçada pelo uso do Webview UI Toolkit, que fornece suporte integrado para o modo claro e escuro; (5) a extensão reduz a carga cognitiva eliminando a necessidade de manter vários aplicativos abertos e simplificando significativamente o processo de validação do código; e, por fim, (6) a extensão promove o controle do usuário ao disponibilizar um conjunto de funcionalidades modulares.

- **C1-RNF2**. As interfaces dos dois elementos de *layout* contribuídos pela extensão — o painel do participante do BOCA (C1-RF1) e o painel de casos de teste (C1-RF3) — foram construídas usando React (META OPEN SOURCE, 2024) e Webview UI Toolkit (MICROSOFT, 2024b). O React foi escolhido por sua ampla popularidade no domínio do desenvolvimento de interface. A biblioteca de componentes Webview UI Toolkit foi escolhida por sua capacidade de garantir coesão entre componentes e com a *design* visual do editor, além de proporcionar suporte automático para modos claro e escuro (C1-RNF2).

A interface BOCADE é exibida na Figura 3. Ademais, vídeos de demonstração das funcionalidades da extensão estão disponíveis no repositório do projeto[1].

[1] https://github.com/gusalbukrk/bocade

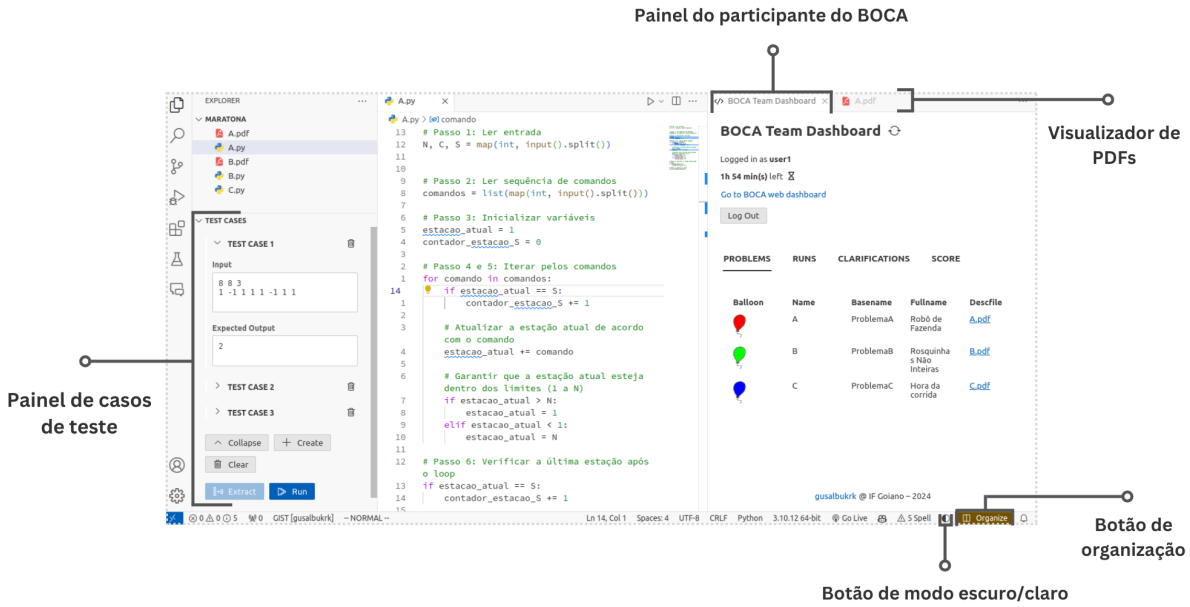


Figura 3 — A interface do BOCADE. Fonte: os autores.

4.3 AVALIAÇÕES

Esta subseção tem como objetivo apresentar a metodologia, os resultados e a discussão referentes a cada um dos dois testes de usabilidade conduzidos para validar o BOCADE. Os arquivos relacionados às avaliações estão disponibilizados publicamente[2].

4.3.1 Avaliação Formativa

A avaliação formativa foi conduzida não apenas para identificar problemas de usabilidade, mas também para verificar o impacto da extensão na UX de participantes de maratonas de programação. Essa avaliação ocorreu no estágio do desenvolvimento em que apenas os dois primeiros requisitos funcionais — a interface do participante do BOCA e o visualizador de PDF — haviam sido implementados.

Metodologia. Durante a fase de teste, os participantes foram encarregados de submeter soluções para quatro problemas de programação. As duas primeiras submissões foram realizadas por meio da interface tradicional do BOCA — isto é, o *site*. Por sua vez, as últimas duas foram feitas através do VS Code com o BOCADE. Para agilizar o teste, as soluções para os problemas foram fornecidas juntamente com os enunciados. O facilitador estava prontamente disponível para responder a quaisquer dúvidas durante as sessões de teste. Para cada problema, os participantes precisavam: acessar a aba de problemas, baixar a descrição do problema disponível em arquivo PDF, digitar a solução fornecida no VS Code, salvá-la como um arquivo de código-fonte, acessar a aba de execuções e submeter o arquivo de código-fonte. Após a sessão de teste, os participantes responderam a um formulário que continha as seguintes três perguntas abertas: (1) “Na sua opinião, quais são as possíveis melhorias no *design* e na UX do *site*?"(2) “Na sua opinião, quais são as possíveis melhorias no *design* e na UX da extensão?"(3) “Na sua opinião, qual, entre o *site* e a extensão, oferece o melhor *design* e UX?". O formulário utilizado

[2] https://drive.google.com/drive/folders/12dsTjcqSldL_AJzWWIdhvACPBjPPtV4y?usp=sharing

nesta avaliação não tinha um campo de consentimento, comprometendo assim a adesão às normas de ética de pesquisas acadêmicas. Contundo, a ausência de coleta de dados pessoais atenua a gravidade desse lapso.

Participantes. O teste foi realizado com 15 indivíduos com formação ou em formação em cursos relacionados à Ciência da Computação — 6 alunos do curso Técnico em Informática integrado ao Ensino Médio, 6 alunos de graduação em Sistemas de Informação e 3 indivíduos com mestrado em Ciência da Computação.

Execução. O teste foi conduzido em março de 2024 nos Laboratórios de Informática do Instituto Federal Goiano — Campus Avançado Catalão.

Resultados. Foi realizada uma análise de conteúdo das respostas. As respostas às duas primeiras perguntas da pesquisa revelaram uma preferência pelo BOCADE, como evidenciado pela postura mais crítica dos participantes em relação ao *site*. As respostas à pergunta 1 revelaram críticas direcionadas à interface do *site* e, em particular, ao seu *design* visual desatualizado. Uma análise das respostas à pergunta 2 revelou que as recomendações para a extensão se concentraram inteiramente em pequenos ajustes de interface. Nenhum problema de usabilidade foi encontrado. As respostas à pergunta 3 — o item mais pertinente para validar o *software* — revelaram uma preferência unânime dos usuários pela extensão em comparação com o *site*. Em particular, muitos respondentes enfatizaram a capacidade da extensão de aumentar a eficiência do usuário.

Discussão. Os participantes forneceram um *feedback* unanimemente positivo sobre o BOCADE. Portanto, os resultados indicaram que a extensão tem o potencial de melhorar a UX em comparação com a interface tradicional do BOCA. Ademais, os participantes não identificaram nenhum problema de usabilidade e os autores não consideraram nenhum dos pequenos ajustes sugeridos como significativo ou adequado o suficiente para justificar sua implementação.

Limitações. O questionário teve um escopo bastante limitado. No entanto, foi suficiente para atingir o objetivo da avaliação formativa de obter a validação inicial. Além disso, é importante ressaltar que o questionário conciso não comprometeu a capacidade de identificar problemas de usabilidade, pois os testes foram conduzidos individualmente, permitindo que o facilitador observasse de perto os usuários durante toda a sessão de teste.

4.3.2 Avaliação Somativa

A avaliação somativa ocorreu após a implementação de todas as funcionalidades planejadas e adotou uma metodologia mais rigorosa para assegurar uma avaliação robusta da extensão. O maior rigor foi alcançado por meio da utilização de um questionário mais abrangente e da condução da avaliação dentro do contexto de uso para o qual o BOCADE foi projetado — uma maratona de programação.

Metodologia. A fase de testes teve duração de duas horas e consistiu em uma competição de programação em duplas, a qual foi dividida em duas partes. Em cada parte da maratona, os participantes foram encarregados de resolver dois problemas. Diferentemente do que ocorreu na primeira avaliação, as soluções não foram fornecidas juntamente com as descrições dos problemas. Na primeira parte, os participantes desenvolveram soluções no VS Code e as submeteram através da interface tradicional do BOCA — isto é, o *site*. Na parte subsequente, os participantes codificaram e enviaram suas soluções inteiramente através do VS Code com o BOCADE. Após a fase de testes, um questionário de cinco seções foi administrado para coletar informações sobre as experiências dos participantes com ambas as interfaces. A seção inicial do questionário se concentrou em coletar os endereços de e-mail dos participantes e garantir seu consentimento para o uso dos dados coletados, garantindo a adesão às normas de ética em pesquisa acadêmica. Depois disso, o questionário se aprofundou em dados demográficos e ex-

periências anteriores (como, por exemplo, gênero, idade e experiência anterior com o BOCA). Para comparar a usabilidade de ambas as interfaces, o questionário empregou o SUS duas vezes — uma para o *site* e outra para a extensão. O questionário SUS utilizado foi a versão padrão descrita por (LEWIS, 2018), com a tradução para o português brasileiro realizada pelos autores desse trabalho. Por fim, a última seção continha perguntas diversas, fechadas e abertas, destinadas a avaliar melhor a UX da extensão e obter uma comparação direta com a UX do *site*. Essas perguntas diversas abrangeram tópicos como a percepção do usuário sobre o *design* visual e a experiência de navegação para ambas as interfaces, classificação das funcionalidades da extensão por utilidade e intenção de uso futuro.

Participantes. O amostra de participantes foi composta por 16 alunos do último ano do curso Técnico em Informática integrado ao Ensino Médio.

Execução. O teste foi conduzido em maio de 2024 em um dos Laboratórios de Informática do Instituto Federal Goiano — Campus Avançado Catalão.

Resultados. A distribuição de gênero dos participantes consistiu em 10 indivíduos do sexo masculino e 6 do sexo feminino, com idade média de 17,4 anos. Em relação às aspirações acadêmicas, 4 participantes expressaram interesse em cursar uma graduação relacionada à Ciência da Computação, 7 estavam indecisos e 4 não pretendiam cursar uma graduação nessa área. A amostra incluiu nove participantes com experiências anteriores em maratonas de programação administradas através do BOCA, seis dos quais haviam participado da avaliação formativa. No geral, as respostas indicaram uma preferência dos usuários pela extensão em comparação ao ambiente tradicional, embora menos pronunciada do que a observada anteriormente nos resultados da avaliação formativa.

Em resposta a uma pergunta direta sobre qual interface preferiam para competições futuras, 14 participantes indicaram a extensão, 1 participante preferiu o *site* e 1 participante não expressou preferência. No entanto, as pontuações SUS apresentaram resultados menos favoráveis para a extensão, conforme ilustrado na Figura 4. O *site* atingiu uma pontuação SUS média de 51,3 (mediana de 52,5, nota F), enquanto a extensão obteve uma pontuação média de 74,3 (mediana de 75, nota B) — uma diferença significativa de 23 pontos. As diferenças individuais das pontuações SUS apresentaram uma mediana de 7,5 e sua distribuição pode ser dividida em quatro grupos: quatro participantes atribuíram pontuações ligeiramente (isto é, 10 pontos ou menos) mais altas para o *site*, um teve exatamente a mesma pontuação para ambas as interfaces, quatro tiveram pontuações ligeiramente mais altas para a extensão e sete tiveram pontuações significativamente mais altas (isto é, acima de 10 pontos) para a extensão.

Além disso, as respostas revelaram uma preferência pronunciada dos usuários pelo *design* visual da extensão, refletida nas pontuações médias (em uma escala de 5) de 4,31 (mediana de 4) em comparação com apenas 2,75 (mediana de 2,5) para o *site*. Os resultados também revelaram uma preferência pela extensão em termos de experiência de navegação — uma clara maioria (10) favoreceu a extensão, enquanto 5 usuários relataram nenhuma diferença significativa entre as interfaces e apenas um usuário achou a navegação do *site* mais eficiente. Com base na classificação das funcionalidades da extensão, foram calculadas, para cada funcionalidade, as pontuações média e mediana de utilidade, respectivamente, em uma escala de até 5. Isso resultou na seguinte classificação: visualizador de PDF integrado (3,69 e 4), painel de casos de teste (3,44 e 4,5), interface de participante BOCA integrada (2,81 e 3), botão de organização (2,75 e 2,5) e botão de alternância entre os modos claro e escuro (2,31 e 2). Notadamente, os participantes não encontraram nenhum problema de usabilidade durante a avaliação.

Discussão. Os resultados corroboram os achados anteriores, evidenciando a preferência dos usuários pela extensão em comparação ao *site*. No entanto, os resultados das pontuações SUS apresentam como ressalva a discrepância considerável entre a média e a mediana das dife-

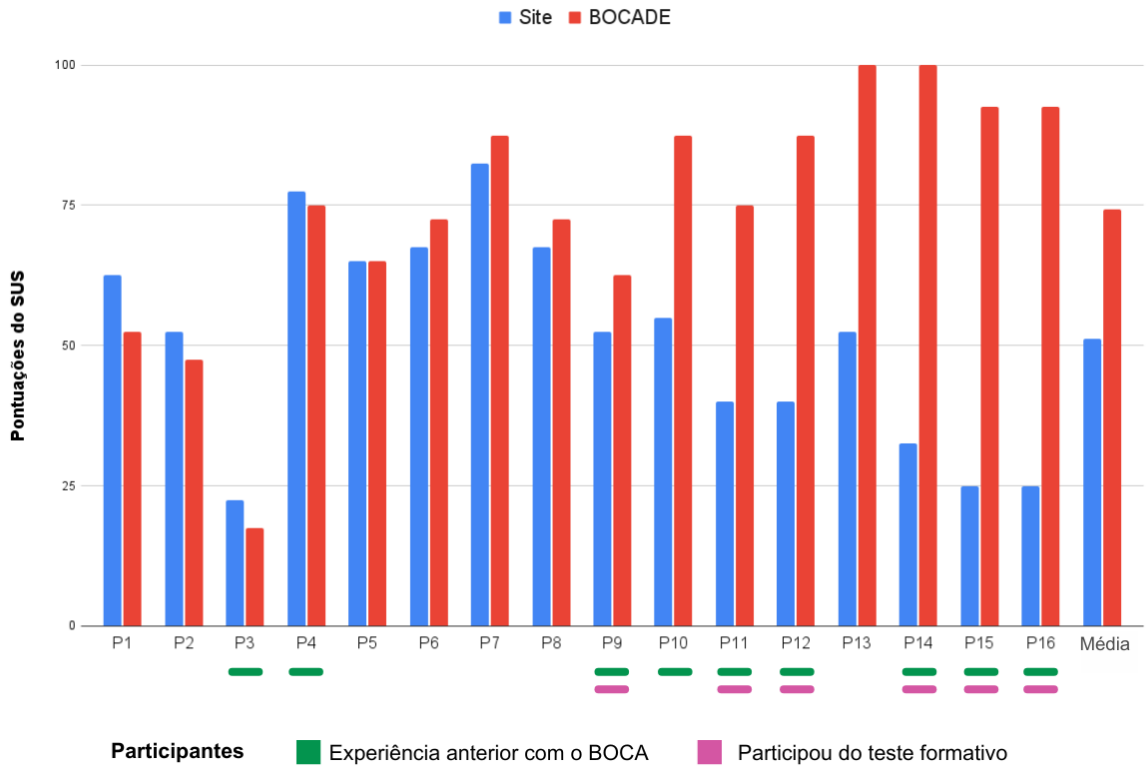


Figura 4 — As pontuações SUS de todos os 16 participantes para ambas as interfaces e as médias. Fonte: os autores.

renças das pontuações SUS (23 vs 7,5). Por conseguinte, realizamos uma análise das respostas para investigar as razões subjacentes a essa diferença. A análise revelou uma relação entre a diferença das pontuações SUS e a participação na avaliação formativa. Os participantes que haviam participado da avaliação formativa exibiram uma diferença de pontuação média do SUS muito maior (49) do que os participantes que não participaram da avaliação formativa (7). Particularmente notável é a pontuação média muito alta do SUS de 85 (mediana de 90, nota A+) atribuída à extensão pelos participantes que participaram da avaliação anterior. A explicação mais plausível para explicar uma disparidade tão significativa é que os participantes da avaliação anterior provavelmente se beneficiaram de um efeito de aprendizagem decorrente de sua exposição prévia à extensão. Consequentemente, essa familiaridade resultou em uma experiência de uso mais satisfatória. Outro fator relacionado que pode ter contribuído para a disparidade é que a extensão pode não ter sido adequadamente introduzida para os participantes. Devido à metodologia e ao número de usuários participando simultaneamente do teste, o facilitador não conseguiu fornecer orientação individualizada na mesma medida que na avaliação formativa.

Ademais, as respostas às perguntas abertas foram bastante concisas, apenas reforçando as respostas objetivas sem oferecer informações complementares e, portanto, não produziram nenhum *insight* acionável. Por outro lado, a diferença significativa nas pontuações do SUS com base na familiaridade do usuário sinalizou a importância de incorporar uma seção de ajuda na extensão. A interface *web* do BOCADE não possui recursos relacionados à ajuda e, por essa razão, a inclusão de uma seção de ajuda representaria mais uma melhoria proporcionada pela extensão.

Ameaça à validade. O efeito de aprendizagem observado nas diferenças das pontuações SUS entre os participantes com e sem exposição prévia à extensão indica que a usabilidade percebida da extensão pode não ser tão significativa quanto a diferença geral de 23 pontos su-

gere. A diferença mais modesta de 7 pontos entre os participantes sem experiência prévia parece indicar que o desempenho superior da extensão em usabilidade é menos pronunciado quando avaliado por usuários iniciantes. No entanto, é necessário levar em conta a possibilidade de que as classificações de usabilidade menos favoráveis de usuários iniciantes decorram de orientação insuficiente durante os testes. Ademais, a considerável diferença entre as médias atribuídas pelos participantes com experiência prévia em maratonas administradas através do BOCA (35,9 para o *site* e 85 para a extensão) é bastante favorável à extensão. Isso porque, esses usuários estavam familiarizados com ambas as interfaces e, portanto, puderam oferecer uma comparação mais bem fundamentada, independentemente de quaisquer deficiências instrucionais durante o teste.

5 BOCA PROBLEMS BUILDER

Nesta subseção, é apresentada uma visão geral do processo de desenvolvimento e do teste de usabilidade do BOCA Problems Builder.

A criação de problemas de programação é uma tarefa demorada e propensa a erros (SARSA et al., 2022; ZAVALA; MENDOZA, 2018). Por isso, os organizadores de competições de programação frequentemente optam por utilizar problemas existentes de fontes de terceiros em vez de elaborar problemas originais. Tanto a OBI quanto a Maratona de Programação oferecem acesso público aos problemas das competições anteriores em seus respectivos *sites*[3]. Todavia, a curadoria e a adaptação de problemas existentes também requerem considerável tempo e esforço.

No BOCA, é necessário fornecer um pacote de problemas ao registrar um problema para uma maratona. Um pacote de problemas é um arquivo ZIP contendo — entre outros arquivos — um arquivo PDF com a descrição do problema e arquivos de texto com casos de teste para a avaliação automatizada de submissões. O recurso de criação de pacotes de problemas do BOCA é bastante limitado, pois não abrange tarefas comuns da criação de pacotes de problemas, como a inclusão de vários arquivos de casos de teste e a geração de arquivos PDF (CAMPOS, 2024).

O BOCA Problems Builder é uma ferramenta de criação de pacote de problemas alternativa ao criador nativo do BOCA que oferece um conjunto mais robusto de funcionalidades. O propósito da ferramenta é aprimorar a UX de organizadores de maratonas ao facilitar e agilizar a criação de pacotes de problemas.

5.1 REQUISITOS

A seguir, são descritos de forma sucinta os requisitos funcionais e não funcionais do BOCA Problems Builder.

- **C2-RF1**: desenvolvimento de um gerenciador de problemas para possibilitar a visualização, criação, edição e exclusão de problemas.
- **C2-RF2**: elaboração de um catálogo de problemas de programação, retirados dos arquivos da OBI e da Maratona de Programação, cujo o objetivo é facilitar a busca e a importação de problemas existentes.
- **C2-RF3**: desenvolvimento de um recurso para a personalização de metadados da competição, permitindo que os usuários editem parâmetros, como o nome e o logotipo da maratona, a serem exibidos nos arquivos PDF.

[3] https://olimpiada.ic.unicamp.br/passadas/ e https://maratona.sbc.org.br/hist/index.html

- **C2-RF4**: desenvolvimento de um recurso de persistência e *backup*. Devido à arquitetura puramente *client-side* da aplicação, os dados devem ser armazenados no navegador *web* para garantir a persistência entre sessões. Ademais, a funcionalidade de *backup* é crucial para permitir a preservação de dados de longo prazo e a transferência de dados entre dispositivos.

- **C2-RF5**: desenvolvimento de um recurso de geração de arquivos PDF. Essa funcionalidade é pertinente, pois as descrições dos problemas são comumente distribuídas nesse formato.

- **C2-RF6**: desenvolvimento de um recurso de geração de arquivos ZIP. Essa funcionalidade é pertinente, pois os pacotes de problemas são arquivos do formato ZIP.

- **C2-RNF1**: criação de uma interface intuitiva e de fácil uso.

5.2 IMPLEMENTAÇÃO

O BOCA Problems Builder foi projetado como uma aplicação *web* visando à facilidade de uso, uma vez que tais aplicações são independentes do sistema e não demandam instalação. Uma arquitetura inteiramente *client-side* foi adotada para eliminar a necessidade de uma infraestrutura de *back-end*, evitando assim quaisquer custos relacionados à hospedagem. A escolha do conjunto de soluções foi influenciada principalmente pela popularidade das alternativas disponíveis. TypeScript (MICROSOFT, 2024a) foi escolhido como a linguagem de programação para a aplicação. Ademais, as bibliotecas utilizadas na implementação de cada um dos requisitos da aplicação são apresentadas a seguir.

- **C2-RF1**. As funcionalidades de gerenciamento de problemas não exigiram o uso de bibliotecas.

- **C2-RF2**. A implementação desse recurso envolveu a utilização de *web crawling*, que é uma técnica destinada à descoberta e extração automatizadas de informações de um conjunto específico de páginas da *web*. O *web crawling* ético deve estar em conformidade com o protocolo de exclusão de robôs de um *site*, que, por meio de um arquivo *robots.txt*, define as permissões e limitações relacionadas às páginas que os *web crawlers* têm permissão para minerar dados (GOLD; LATONERO, 2017). Durante a fase de elicitação de requisitos deste *software*, os arquivos *robots.txt* dos *sites* da OBI e da Maratona de Programação foram examinados, e nenhuma restrição aplicável ao *web crawling* foi identificada. Para realizar o *web crawling*, o utilitário de linha de comando wget (RÜHSEN; SHAH; SCRIVANO, 2024) foi utilizado para baixar todos os arquivos relacionados a competições anteriores dos servidores da OBI e da Maratona de Programação. As tarefas restantes de extração e transformação de dados foram realizadas em Python, utilizando as seguintes bibliotecas: pdfplumber (SINGER-VINE, 2024), para a extração de dados de arquivos PDF, pypdf (CIMON; THOMA; PEVELER, 2024), para a divisão de arquivos PDF, e Googletrans, (HAN, 2024) para a tradução dos problemas disponíveis exclusivamente em inglês. Os *scripts* relacionados a esse requisito estão versionados no mesmo repositório que o código-fonte da aplicação *web*. Ademais, o catálogo de problemas resultante está disponível *online*[4] e é composto por mais de 600 problemas extraídos da OBI (abrangendo os anos de 2009 a 2024) e da Maratona de Programação (de 2013 a 2023). É importante destacar que a organização e a precisão do catálogo ainda não

[4] https://archive.org/details/@bocaproblemsarchive

alcançaram um nível ideal e, por isso, seria necessário ajustar o código para alcançar um maior refinamento.

- **C2-RF3**. A personalização de metadados da maratona não demandou o uso de bibliotecas.
- **C2-RF4**. A implementação das funcionalidades de persistência e *backup* foi realizada através do banco de dados IndexedDB e da biblioteca Dexie.js (LAMMES, 2024).
- **C2-RF5**. A biblioteca pdfmake (PAMPUCH; M., 2024) foi utilizada para a implementação do recurso de geração de arquivos PDF.
- **C2-RF6**. A biblioteca JSZip (KNIGHTLEY, 2024) foi utilizada para a implementação do recurso de geração de arquivos ZIP.
- **C2-RNF1**. React (META OPEN SOURCE, 2024) e Bootstrap (OTTO; THORNTON, 2024) foram usados para o desenvolvimento da interface.

A aplicação se encontra disponível *online*[5]. A interface do BOCA Problems Builder é apresentada na Figura 5. Ademais, um vídeo de demonstração da aplicação está disponível no repositório do projeto[6].

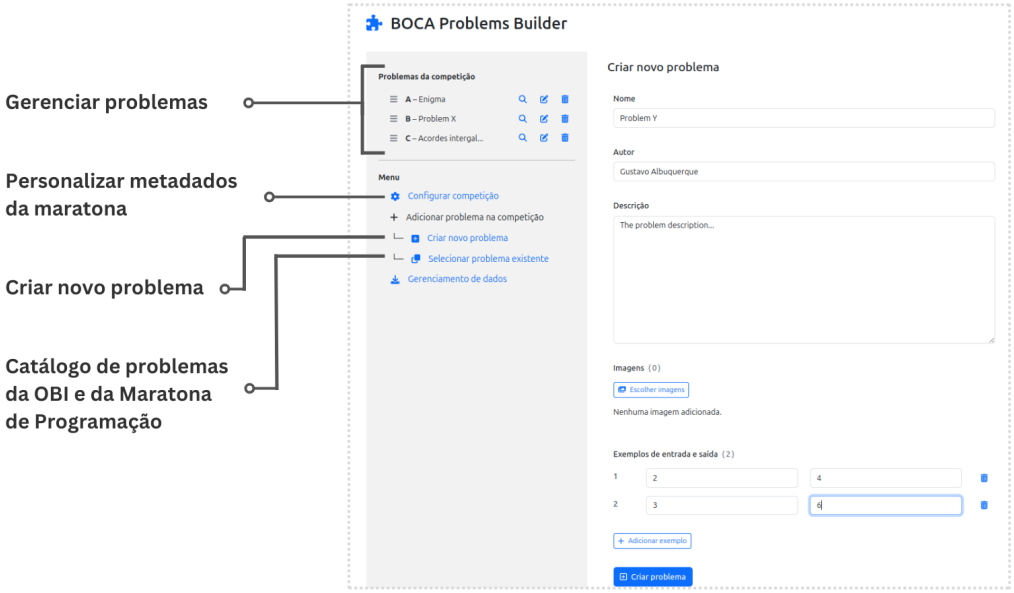


Figura 5 — A interface do BOCA Problems Builder. Fonte: os autores.

5.3 AVALIAÇÃO SOMATIVA

Esta subseção se concentra em apresentar a metodologia, resultados e discussão do teste de usabilidade conduzido para validar o BOCA Problems Builder. Os arquivos relacionados à avaliação estão disponíveis *online*[7].

[5] https://boca-problems-builder.netlify.app/
[6] https://github.com/gusalbukrk/boca-problems-builder
[7] https://drive.google.com/drive/folders/1ZT-X4F3hoWYuydMbsfJKBgvf1OiJnItl?usp=sharing

Conforme é comum em testes de usabilidade, o objetivo principal da avaliação foi identificar problemas de usabilidade para possibilitar melhorias incrementais. Além disso, um objetivo secundário desse teste consistiu em avaliar se os usuários percebem que a aplicação atende ao seu propósito de facilitar a criação de pacotes de problemas.

Metodologia. Durante a fase de testes, os participantes foram encarregados de completar 14 tarefas referentes ao seguinte cenário "Você está organizando uma maratona de programação que consistirá em três problemas e você vai usar o BOCA Problems Builder para gerar os pacotes de problemas para esta competição". As tarefas eram diversas e coletivamente abrangiam todas as funcionalidades disponíveis na aplicação. Os métodos empregados para coleta de dados foram o protocolo *think aloud* e um questionário pós-teste. Os comentários dos participantes considerados pertinentes foram anotados pelo facilitador e, posteriormente, documentados em uma coluna designada dentro da planilha contendo as respostas do questionário. Após a conclusão da fase de testes, um questionário de seis seções foi administrado para coletar informações sobre os participantes e suas opiniões sobre o BOCA Problems Builder. A seguir, apresenta-se uma breve descrição de cada uma das seções desse questionário. A seção inicial possuía as declarações de imparcialidade e consentimento, solicitando aos participantes a confirmação de sua capacidade de fornecer *feedback* isento e a autorização para o uso dos dados em conformidade com os princípios éticos da pesquisa acadêmica. A segunda seção focou na coleta de dados demográficos e de experiências anteriores. Em seguida, entre as seções 3 e 5, foram aplicados, respectivamente, os seguintes questionários padronizados: SUS, para avaliação da usabilidade; TAM, para avaliação da utilidade percebida e da facilidade de uso; e VisAWI-Pos, para avaliação da qualidade do *design* visual. É importante ressaltar que o questionário SUS utilizado foi baseado na tradução e adaptação transcultural para o português brasileiro proposta por Lourenço, Carmona e Lopes (2022). Em contraste, os questionários TAM e VisAWI-Pos foram traduzidos pelos autores devido à falta de traduções validadas. Por fim, a sexta seção solicitou a classificação, em termos de utilidade, de oito funcionalidades propostas para uma possível versão futura e incluiu um campo aberto para que os participantes relatassem eventuais problemas identificados ou sugerissem novas funcionalidades.

Participantes. A amostra de participantes foi composta por oito educadores de programação de computadores. Embora os usuários mais adequados fossem aqueles com experiência prévia na preparação de pacotes de problemas para maratonas administradas através do BOCA, nenhum indivíduo com tal histórico estava disponível para o estudo. Para mitigar essa discrepância, uma breve intervenção pré-teste foi ministrada para os participantes. Essa apresentação incluiu informações introdutórias sobre a programação competitiva, o sistema de gerenciamento de maratonas BOCA e o processo de criação de pacotes de problemas para competições administradas através do BOCA.

Execução. O teste foi conduzido em agosto de 2024 em um dos Laboratórios de Informática do Instituto Federal Goiano — Campus Avançado Catalão. Os participantes foram testados individualmente, com cada sessão tendo uma duração de 30 a 50 minutos — incluindo a apresentação e o preenchimento do questionário.

Resultados. Os participantes da avaliação eram compostos por dois homens e seis mulheres, apresentando uma idade média de 38,5 anos e uma mediana de 38 anos. Todos os participantes possuíam formação em áreas relacionadas à Ciência da Computação e relataram lecionar disciplinas de programação para alunos do ensino médio e/ou de graduação. Entretanto, apenas três participantes possuíam experiência na organização de maratonas de programação.

Na Figura 6, são apresentas as pontuações médias dos questionários padronizados SUS, TAM-PU, TAM-PEU e VisAWI-Pos, os quais foram empregados para avaliar, respectivamente, a usabilidade, a utilidade percebida, a facilidade de uso e a qualidade do *design* visual. As

pontuações TAM e VisAWI-Pos foram normalizadas para uma escala de 0-100 através da normalização min-max para facilitar a apresentação dos dados. De particular interesse são as pontuações médias de 90,6, 91,1, 88,0 e 82,4.

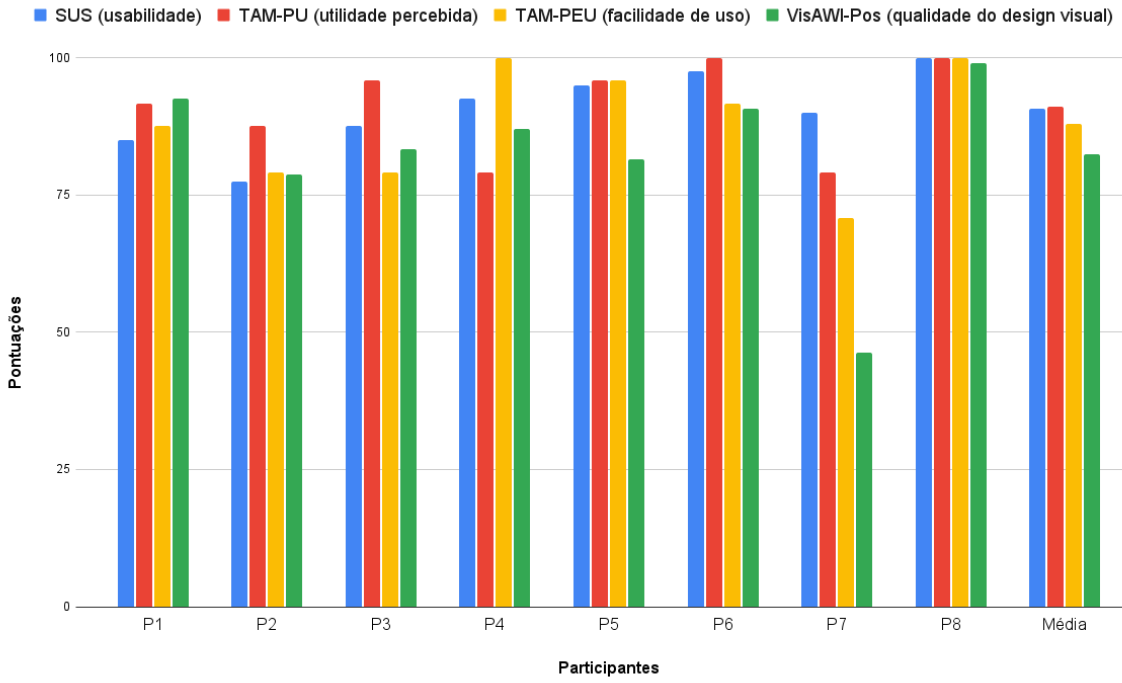


Figura 6 — As pontuações SUS, TAM-PU, TAM-PEU e VisAWI-Pos de todos os oito participantes e as médias. Fontes: os autores.

Na sexta e última seção do questionário, os participantes foram solicitados a avaliar a utilidade de oito possíveis funcionalidades futuras por meio de uma escala de 5 pontos. Essas funcionalidades foram priorizados pelos participantes na seguinte ordem: mecanismo para filtragem de problemas por nível de dificuldade (pontuação média de 5), personalização das instruções impressas no livreto de problemas (4,75), roteamento *client-side* para melhorar a navegação (4,25), suporte ao idioma inglês (4,125), integração de um editor de texto com recursos de formatação (4,125), adição de problemas de programação de outras fontes (4,125), melhorias gerais no *design* visual (3,875) e suporte ao modo escuro (3,75). Além disso, todos os respondentes, exceto um, destacaram a ausência de uma indicação visual após o envio do formulário. Esse *feedback* emergiu como uma das observações mais significativas coletadas durante o teste, em virtude de sua prevalência e da forte ênfase que os participantes atribuíram à necessidade de tal recurso.

Discussão. Os resultados foram positivos para o BOCA Problems Builder. As pontuações SUS, TAM-PEU e VisAWI-Pos indicam que a aplicação exibe um nível satisfatório de usabilidade, facilidade de uso e qualidade de *design* visual. Especificamente em relação à pontuação SUS, que representa o grau de usabilidade geral do sistema, é notável que a média obtida esteja dentro da nota A+ na escala de classificação curva de Sauro-Lewis (LEWIS, 2018). Conforme discutido na revisão da literatura, usabilidade e qualidade do *design* visual são fatores críticos para a UX. Portanto, o forte desempenho nesses aspectos sugere que a aplicação ofereceu uma UX positiva. Ademais, a pontuação TAM-PU demonstra que os participantes percebem a aplicação como útil, um fator essencial para a aceitação de uma ferramenta tecnológica. Em relação aos resultados da classificação de potenciais funcionalidades futuras, é significativo que

todas as funcionalidades receberam classificações relativamente altas, o que evidencia a pertinência dessas funcionalidades. Por fim, a ausência de um indicador visual após submissões de formulários bem-sucedidas foi uma falha de *design* significativa, dada a ubiquidade desse recurso em aplicações modernas.

Ameaça à validade. A ausência de participantes com experiência na criação de pacotes de problemas provavelmente comprometeu a qualidade e a profundidade das respostas, uma vez que os participantes não possuíam a experiência necessária para uma análise mais fundamentada. Entretanto, a breve intervenção realizada antes do teste mitigou parcialmente essa limitação.

6 CONCLUSÃO

Neste trabalho, foram apresentadas duas contribuições para o domínio dos *softwares* destinados à programação competitiva: o BOCADE e o BOCA Problems Builder. O BOCADE é uma extensão do VS Code que oferece um ambiente de desenvolvimento especializado para uso durante as maratonas de programação. Por sua vez, o BOCA Problems Builder é uma aplicação *web* para facilitar e agilizar a criação de pacotes de problemas para competições. O propósito desses *softwares* é aprimorar, respectivamente, a UX dos participantes e organizadores de maratonas de programação administradas através do BOCA. Com o intuito de validar os *softwares* desenvolvidos e testar a hipótese do estudo, foram realizados testes de usabilidade. Embora essas avaliações possuam certas limitações, os seus resultados, de modo geral, fornecem evidências suficientes para concluir que ambos os *softwares* cumprem o seu objetivo de aprimorar a UX de seus respectivos usuários.

No que diz respeito a trabalhos futuros, a implementação de uma seção de ajuda seria pertinente para o BOCADE, conforme mencionado na discussão dos resultados do teste somativo. Com a exceção desse recurso, não há outras sugestões de melhorias ou novas funcionalidades para esse software, isso porque todas as funcionalidades planejadas foram implementadas e nenhum recurso adicional ou problema de usabilidade foi identificado na avaliação somativa. Em contrapartida, nem todas as funcionalidades identificadas na fase de elicitação de requisitos do BOCA Problems Builder foram implementadas, devido à priorização de outras consideradas mais relevantes. Os participantes do teste de usabilidade atribuíram pontuações elevadas de utilidade a cada uma dessas possíveis funcionalidades, o que evidencia sua pertinência. Portanto, esforços futuros poderiam se concentrar na implementação desses recursos. Além disso, o extenso catálogo de centenas de problemas de programação, compilado durante o desenvolvimento do BOCA Problems Builder, pode servir como base para pesquisas futuras. Possíveis vias para exploração incluem: análise de dados para identificar tendências e padrões na dificuldade dos problemas e distribuição de tópicos, integração do catálogo em plataformas de aprendizagem e implementação de técnicas de aprendizagem de máquina para geração ou resolução automatizada de problemas de programação.

7 REFERÊNCIAS

ADITHYA, Vikram Chidanand. **Security sandboxing for PCˆ 2**. 2011. Tese (Doutorado) – California State University, Sacramento.

AGRAWAL, Divyanshu. **Competetive Programming Helper**. 2020. Disponível em: <https://github.com/agrawal-d/cph>. Acesso em: 18 ago. 2024.

AGRAWAL, Vibhor. **compile-run**. 2020. Disponível em: <https://github.com/vibhor1997a/compile-run/>. Acesso em: 18 ago. 2024.

ALGAR TELECOM. **Roteiro para configuração de MiniMaratona de programação**. 2012. Disponível em: <https://docplayer.com.br/36306096-Minimaratonas-de-programacao.html>. Acesso em: 30 mar. 2024.

ALVES, Edson. **ejtools**. 2019. Disponível em: <https://gitlab.com/ejudge/ejtools>. Acesso em: 18 ago. 2024.

AUDRITO, Giorgio; CIOBANU, Madalina; LAURA, Luigi. Giochi di Fibonacci: Competitive Programming for Young Students. **Olympiads in Informatics**, p. 19–31, 2023.

CAMPOS, Cassio P de. **BOCA**. 2024. Disponível em: <https://github.com/cassiopc/boca>. Acesso em: 18 ago. 2024.

CAMPOS, Cassio P de; FERREIRA, Carlos E. BOCA: um sistema de apoio a competições de programação. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. WORKSHOP de Educação em Computação. [S.l.: s.n.], 2004.

CARMO NOGUEIRA, Tiago do; SOUZA CAMPOS, Eudes de; FERREIRA, Deller James. Cognition Developing of Computer Higher Education Students Through Gamification in the Algorithm Teaching-Learning Process. In: SBC. ANAIS do XXVI Workshop sobre Educação em Computação. [S.l.: s.n.], 2018.

CHEAH, Chin Soon. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. **Contemporary Educational Technology**, Bastas, v. 12, n. 2, ep272, 2020.

CHEAH, W. H. et al. Mobile technology in medicine: development and validation of an adapted system usability scale (SUS) questionnaire and modified technology acceptance model (TAM) to evaluate user experience and acceptability of a mobile application in MRI safety screening. **Indian Journal of Radiology and Imaging**, v. 33, n. 1, p. 36–45, 2023.

CIMON, Lucas; THOMA, Martin; PEVELER, Matthew. **pypdf**. 2024. Disponível em: <https://github.com/py-pdf/pypdf>. Acesso em: 18 ago. 2024.

COORE, Daniel; FOKUM, Daniel. Facilitating course assessment with a competitive programming platform. In: PROCEEDINGS of the 50th ACM Technical Symposium on Computer Science Education. [S.l.: s.n.], 2019. P. 449–455.

CRUZ, Allan Kássio Beckman Soares da et al. Utilizaçao da plataforma beecrowd de maratona de programaçao como estratégia para o ensino de algoritmos. In: SBC. ANAIS Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital. [S.l.: s.n.], 2022. P. 754–764.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, p. 319–340, 1989.

DOMJUDGE. **DOMjudge – About the project**. 2024. Disponível em: <https://www.domjudge.org/about>. Acesso em: 18 ago. 2024.

FLÓREZ, Francisco Buitrago et al. Changing a generation's way of thinking: Teaching computational thinking through programming. **Review of Educational Research**, Sage Publications Sage CA: Los Angeles, CA, v. 87, n. 4, p. 834–860, 2017.

GANORKAR, Chinmay Sanjay. PC2 web team interface. **ScholarWorks**, 2017.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de pesquisa**. [S.l.]: Plageder, 2009.

GLEZ-PEÑA, Daniel et al. Web scraping technologies in an API world. **Briefings in bioinformatics**, Oxford University Press, v. 15, n. 5, p. 788–797, 2014.

GOLD, Zachary; LATONERO, Mark. Robots welcome: Ethical and legal considerations for web crawling and scraping. **Wash. JL Tech. & Arts**, HeinOnline, v. 13, p. 275, 2017.

HAN, Suhun. **Googletrans**. 2024. Disponível em: <https://github.com/ssut/py-googletrans>. Acesso em: 18 ago. 2024.

ICPC FOUNDATION. **The ICPC International Collegiate Programming Contest**. [S.l.: s.n.], 2024. Disponível em: <https://icpc.global>. Acesso em: 18 ago. 2024.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). **Ergonomics of human-system interaction—Part 11: Usability: Definitions and concepts (ISO 9241-11: 2018)**. [S.l.: s.n.], 2018.

JSDOM. **jsdom**. [S.l.: s.n.], 2024. Disponível em: <https://github.com/jsdom/jsdom>. Acesso em: 18 ago. 2024.

KNIGHTLEY, Stuart. **JSZip**. 2024. Disponível em: <https://github.com/Stuk/jszip>. Acesso em: 18 ago. 2024.

KURTANOVIĆ, Zijad; MAALEJ, Walid. Automatically classifying functional and non-functional requirements using supervised machine learning. In: IEEE. 2017 IEEE 25th International Requirements Engineering Conference (RE). [S.l.: s.n.], 2017. P. 490–495.

LAMMES, Simon. **Dexie.js**. 2024. Disponível em: <https://github.com/dexie/Dexie.js>. Acesso em: 18 ago. 2024.

LEWIS, James R. The System Usability Scale: Past, Present, and Future. **International Journal of Human–Computer Interaction**, Springer, v. 34, n. 7, p. 577–590, 2018.

LIMA, Adriano Luiz de Souza; BENITTI, Fabiane Barreto Vavassori. UsabilityZero: Can a Bad User Experience Teach Well?. **Informatics in Education**, ERIC, v. 20, n. 1, p. 69–84, 2021.

LIMA, Danilo Teixeira; MOURA, Flavio Rafael Trindade et al. Ux-tracking: Web and multimodal tool for user experience evaluation. In: SBC. ANAIS Estendidos do XXVIII Simpósio Brasileiro de Sistemas Multimídia e Web. [S.l.: s.n.], 2022. P. 107–110.

LOURENÇO, Douglas Fabiano; CARMONA, Elenice Valentim; LOPES, Maria Helena Baena de Moraes. Tradução e adaptação transcultural da System Usability Scale para o português do Brasil. **Aquichan**, v. 22, n. 2, 2022.

MAGGIOLO, Stefano; MASCELLANI, Giovanni. Introducing CMS: a contest management system. **Olympiads in Informatics**, Vilnius University, v. 6, p. 86–99, 2012.

MAJUMDAR, Anshuman. MacACM: Encouraging competitive programming via mentorship and outreach. **XRDS: Crossroads, The ACM Magazine for Students**, ACM New York, NY, USA, v. 24, n. 1, p. 14–15, 2017.

META OPEN SOURCE. **React**. 2024. Disponível em: <https://react.dev/>. Acesso em: 18 ago. 2024.

MICROSOFT. **TypeScript**. 2024. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 18 ago. 2024.

_____. **Webview UI Toolkit for Visual Studio Code**. 2024. Disponível em: <https://github.com/microsoft/vscode-webview-ui-toolkit/>. Acesso em: 18 ago. 2024.

MORAIS, Wall Berg; RIBAS, Bruno César. Maratona-Linux: um ambiente para a Maratona de Programação. **Anais do Computer on the Beach**, p. 416–426, 2019.

MORENO, Julian; PINEDA, Andres F. Competitive programming and gamification as strategy to engage students in computer science courses. **Revista ESPACIOS**, v. 39, n. 35, 2018.

MURTY, Rohan Narayana; DADLANI, Sandeep; DAS, Rajath B. How Much Time and Energy Do We Waste Toggling Between Applications? **Harvard Business Review**, 2022. Disponível em: <https://hbr.org/2022/08/how-much-time-and-energy-do-we-waste-toggling-between-applications>. Acesso em: 18 ago. 2024.

NUNES, Daniel. **ds-contest-tools**. 2024. Disponível em: <https://github.com/danielsaad/ds-contest-tools>. Acesso em: 18 ago. 2024.

OTTO, Mark; THORNTON, Jacob. **Bootstrap**. 2024. Disponível em: <https://github.com/twbs>. Acesso em: 18 ago. 2024.

PAMPUCH, Bartek; M., Libor. **pdfmake**. 2024. Disponível em: <https://github.com/bpampuch/pdfmake>. Acesso em: 18 ago. 2024.

PERRIG, S. A. et al. Development and validation of a positive-item version of the visual aesthetics of websites inventory: The visawi-pos. **Indian Journal of Radiology and Imaging**, p. 1–25, 2023.

PHAM, Minh Tuan; NGUYEN, Tan Bao. The DOMJudge based online judge system with plagiarism detection. In: IEEE. 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF). [S.l.: s.n.], 2019. P. 1–6.

PIEKARSKI, A. E. T. et al. Programação competitiva em um projeto de extensão para o ensino técnico em informática. **Revista Conexão UEPG**, v. 19, n. 1, p. 1–14, 2023.

PIEKARSKI, Ana Elisa et al. A metodologia das maratonas de programação em um projeto de extensão: um relato de experiência. In: ANAIS dos Workshops do Congresso Brasileiro de Informática na Educação. [S.l.: s.n.], 2015. v. 4, p. 1246.

RIIHIAHO, Sirpa. Usability testing. **The Wiley handbook of human computer interaction**, Wiley Online Library, v. 1, p. 255–275, 2018.

RODRIGUES, Luiz; ISOTANI, Seiji. When personalized gamification meets computing education: A multidimensional approach to motivate students to learn. In: SBC. ANAIS do XXXVI Concurso de Teses e Dissertações. [S.l.: s.n.], 2023. P. 50–59.

RÜHSEN, Tim; SHAH, Darshit; SCRIVANO, Giuseppe. **GNU Wget2**. 2024. Disponível em: <https://www.gnu.org/software/wget/>. Acesso em: 18 ago. 2024.

SANTOS, Luiz Gustavo Albuquerque dos. **BOCA: funcionalidades**. 2024. Disponível em: <https://github.com/gusalbukrk/boca/tree/main/funcionalidades>. Acesso em: 18 ago. 2024.

SARSA, Sami et al. Automatic generation of programming exercises and code explanations using large language models. In: PROCEEDINGS of the 2022 ACM Conference on International Computing Education Research-Volume 1. [S.l.: s.n.], 2022. P. 27–43.

SENTANCE, Sue; CSIZMADIA, Andrew. Computing in the curriculum: Challenges and strategies from a teacher's perspective. **Education and information technologies**, Springer, v. 22, p. 469–495, 2017.

SILVA, Thais R de Moura Braga et al. Maratonando! Inspirando e capacitando programadores com diversidade de gênero e variedade de competições. In: SBC. ANAIS do XVII Women in Information Technology. [S.l.: s.n.], 2023. P. 346–351.

SINGER-VINE, Jerermy. **pdfplumber**. 2024. Disponível em: <https://github.com/jsvine/pdfplumber>. Acesso em: 18 ago. 2024.

STACK OVERFLOW. **Stack Overflow Developer Survey**. 2023. Disponível em: <https://survey.stackoverflow.co/2023/#section-most-popular-technologies-integrated-development-environment>. Acesso em: 18 ago. 2024.

TENÓRIO, Thyago; BITTENCOURT, Ig Ibert. A gamified peer assessment model for on-line learning environments: An experiment with MeuTutor. In: SBC. ANAIS do XXIX Concurso de Teses e Dissertações. [S.l.: s.n.], 2016. P. 381–386.

TOMOKI, Maruyama. **vscode-pdf**. 2023. Disponível em: <https://github.com/tomoki1207/vscode-pdfviewer>. Acesso em: 18 ago. 2024.

UZAYR, Sufyan bin. **Mastering Visual Studio Code: A Beginner's Guide**. [S.l.]: CRC Press, 2022.

VAITHILINGAM, Priyan; GUO, Philip J. Bespoke: Interactively synthesizing custom GUIs from command-line applications by demonstration. In: PROCEEDINGS of the 32nd annual ACM symposium on user interface software and technology. [S.l.: s.n.], 2019. P. 563–576.

VERMA, Rishabh. **Visual Studio Extensibility Development**. [S.l.]: Springer, 2020.

XIA, Belle Selene. A pedagogical review of programming education research: what have we learned. **International Journal of Online Pedagogy and Course Design (IJOPCD)**, IGI Global, v. 7, n. 1, p. 33–42, 2017.

YUEN, Kevin KF; LIU, Dennis YW; LEONG, Hong Va. Competitive programming in computational thinking and problem solving education. **Computer Applications in Engineering Education**, Wiley Online Library, v. 31, n. 4, p. 850–866, 2023.

ZAVALA, Laura; MENDOZA, Benito. On the use of semantic-based aid to automatically generate programming exercises. In: PROCEEDINGS of the 49th ACM technical symposium on computer science education. [S.l.: s.n.], 2018. P. 14–19.

ZHAN, Zehui et al. The effectiveness of gamification in programming education: Evidence from a meta-analysis. **Computers and Education: Artificial Intelligence**, Elsevier, v. 3, p. 100096, 2022.

Documento Digitalizado Público

Entrega Final TCC - MELHORANDO A EXPERIÊNCIA DO USUÁRIO EM MARATONAS DE PROGRAMAÇÃO POR MEIO DE SOFTWARES COMPLEMENTARES AO BOCA

Assunto: Entrega Final TCC - MELHORANDO A EXPERIÊNCIA DO USUÁRIO EM MARATONAS DE PROGRAMAÇÃO POR MEIO DE SOFTWARES COMPLEMENTARES AO BOCA
Assinado por: Fabiola Ribeiro
Tipo do Documento: Documentos Externos
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Original

Documento assinado eletronicamente por:

- **Fabiola Goncalves Coelho Ribeiro**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 02/01/2025 00:11:52.

Este documento foi armazenado no SUAP em 02/01/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 688926
Código de Autenticação: ee306aaa91