PortugIF: Uma Ferramenta Interativa para o Ensino de Estruturas de Programação em Portugol

Igor L. S. C. Oliveira[1], **Carlos A. Silva**[1]

[1]Departamento de Informática – Instituto Federal de Minas Gerais (IFMG)
CEP 34.590-390 – Sabará, MG – Brasil

igorescritor@gmail.com , carlos.silva@ifmg.edu.br

Abstract. *This paper presents the development of PortugIF, an online educational tool designed to support beginners in understanding and constructing conditional and looping structures, and variable usage, using the Portugol pseudolanguage. The platform adopts user-centered design principles to ensure an intuitive and accessible learning environment. A user evaluation conducted with 21 participants demonstrated high acceptance rates (above 75%) in terms of usability, explanatory clarity, and educational value. The results highlight PortugIF's potential as a complementary resource for teaching introductory programming concepts.*

Resumo. *Este artigo apresenta o desenvolvimento da PortugIF, uma ferramenta educacional online voltada a estudantes iniciantes em programação. A plataforma tem como objetivo facilitar o entendimento e a construção de estruturas condicionais, de repetição e o uso de variáveis, por meio da pseudolinguagem Portugol. Baseada em princípios de usabilidade e design centrado no usuário, a solução foi avaliada por 21 participantes com diferentes níveis de familiaridade com programação. Os resultados indicaram uma taxa de aprovação superior a 75% em critérios como usabilidade, clareza das explicações e utilidade didática, evidenciando o potencial da PortugIF como recurso complementar no ensino introdutório de lógica de programação.*

1. Introdução

Entre as diversas áreas do conhecimento, o ensino de programação destaca-se por apresentar algumas das maiores dificuldades em seu processo de aprendizagem. Com altas taxas de evasão e reprovação, os desafios no processo de aprendizagem são diversas, porém podemos classificá-los em dois grupos básicos: aqueles relativos à compreensão e aplicação de conceitos abstratos inerentes à programação e as relativas à compreensão e utilização de elementos essenciais, como as estruturas de controle. As dificuldades no processo de compreensão e utilização dos diferentes conceitos programáticos se destaca como um dos grandes motivos para que o o ensino de programação continue a enfrentar desafios persistentes mesmo após décadas de pesquisas e desenvolvimentos na área [Gomes et al. 2008b, Ambrósio 2011].

Motivadas pelas barreiras no processo de aprendizado, diferentes ferramentas e metodologias foram sendo desenvolvidas com o passar dos anos para auxiliar alunos a terem uma melhor compreensão dos elementos e conceitos programáticos. Embora as taxas de reprovação e repetência em disciplinas de programação continuem elevadas, é

notável que tais ferramentas mostram impactos positivos no desempenho de estudantes [Gomes et al. 2008b]. Exemplos de algumas dessas ferramentas e metodologias incluem o AI(3P)A [Faeda et al. 2020], o SICAS e o PROGUIDE [Gomes et al. 2008a].

Contudo, é importante destacar que mesmo com o auxílio de ferramentas de apoio, certos pontos de dificuldade entre os alunos continuam frequentes, tais como dificuldades em noções de estruturas condicionais e de repetição: no estudo feito por [Oliveira et al. 2014], utilizando a ferramenta Scratch, "laços de repetição e estruturas condicionais" foram alguns dos conteúdos com menor aproveitamento, alcançando uma média de apenas 54,4%. Em comparação, os únicos conteúdos com uma média de aproveitamento similar ou menor são "interatividade com usuário e *design* (personalização) de personagens" e "eventos ativados através dos dispositivos de entrada (mouse e teclado)", com 60% e 30% de aproveitamento, respectivamente. Estes conteúdos, embora bastante importantes, não são considerados tão básicos ou essenciais quanto os demais como "números binários", "seguindo instruções" e "passagem de parâmetro e movimentação", todos com média de aproveitamento consideravelmente maior (75%, 81,2% e 80%, respectivamente), sendo que os dois primeiros nem sequer possuem comandos no Scratch. Assim sendo, entre os elementos que podem ser considerados essenciais, as estruturas de repetição e as estruturas condicionais ocupam as menores posições por uma margem significativa, mais de 20 pontos percentuais abaixo do elemento mais próximo — ilustrando, assim, a dificuldade apresentada pelos alunos quanto a essas estruturas.

O ensino de programação promove grande desenvolvimento para estudantes. Mesmo desconsiderando-se casos diretamente relacionados à computação, os resultados obtidos ao familiarizar indivíduos com as lógicas de programação são notáveis — e isso se mostra em diferentes níveis de educação. [Sáez-López et al. 2019], em seu artigo sobre os efeitos da programação no entendimento matemático e científico de alunos de escolas primárias, indica que "esses conceitos de programação, sequências, laços e condicionais permitiram que os estudantes desenvolvessem competências relativas à matemática, ciência e tecnologia, promovendo uma abordagem multidisciplinar". Uma noção similar pode ser vista nas pesquisas de [Rodrigues et al. 2015], que aponta em suas conclusões uma correlação moderada entre o desempenho de estudantes no ENEM e a sua experiência com programação, além de um desempenho melhor de alunos estudantes de programação em todos os eixos cognitivos, quando comparados aos que não possuem essa experiência. É importante destacar que tais observações não valem apenas para um contexto de discentes, mas também de docentes: [Rocha and Prado 2018] aponta que sua pesquisa "(...) mostrou o potencial da atividade de programação utilizada no contexto de formação continuada do professor de matemática, em termos de propiciar a apropriação das tecnologias digitais", além de que "esta forma de aprender (...) pode ser um caminho que oportunize aos professores integrar os Conhecimentos Tecnológico e Pedagógico do Conteúdo (em inglês, *Technological Pedagogical Content Knowledge — TPACK* — modelo teórico apresentado por [Mishra and Koehler 2006]), a fim de reconstruir a base do conhecimento profissional docente para a era digital".

As pesquisas supracitadas evidenciam a importância da programação no contexto de aprendizado em diversos níveis — tanto para crianças, jovens e até mesmo docentes — independente de estarem ou não diretamente envolvidos com a área da informática. Consequentemente, considerar os entraves que iniciantes enfrentam nessa área ganha des-

taque ainda maior, uma vez que os benefícios do seu estudo se estendem além da sua área principal.

Observando casos como a análise da ferramenta Scratch, é possível notar dificuldades consideráveis de novos estudantes em absorver certos conteúdos de programação corretamente. Soma-se a isso, no entanto, um fator ainda mais preocupante: a baixa taxa de aprovação de estudantes em disciplinas iniciais de programação. [Fernandes and Junior 2016] exemplificam essa realidade ao apontarem uma taxa de aprovação de apenas 58% entre os alunos do ensino superior nas disciplinas de linguagem de programação I e II, no período de 2010 a 2015, com uma taxa de evasão de 26% e uma taxa de 16% de reprovação no Campus Sombrio, do Instituto Federal Catarinense. [Coutinho et al. 2017] nos apresentam uma realidade ainda mais preocupante, com uma taxa de aprovação de apenas 49,6% dos alunos da disciplina de graduação em Sistemas e Mídias Digitais da Universidade do Ceará.

Entre os maiores problemas encontrados pelos estudantes, conforme relatado pelo estudo de [Oliveira et al. 2014], alguns dos principais pontos são as estruturas condicionais e de repetição. Há ampla evidência na literatura que corrobora quanto a essa dificuldade: [Cetin et al. 2020] cita diversos estudos abordando os desafios que estudantes vivenciam com estruturas de repetição, mencionando estudos que vão desde 1986 até 2018 relatando dificuldades nessa área. Os estudos de [Vieira et al. 2015] apontam resultados similares, com as estruturas de repetição sendo o terceiro conteúdo da grade inicial de programação no qual os alunos apresentam as maiores dificuldades, seguido em quarto lugar pelas estruturas condicionais: acima desses encontram-se apenas os vetores (segundo lugar) e matrizes (primeiro lugar). [Malik and Coldwell-Neilson 2017] reforça ainda mais essa noção, mencionando estruturas de repetição, funções e vetores como os tópicos mais desafiadores enfrentados pelos respondentes de sua pesquisa — estudantes de Ciências da Computação, Sistemas de Informação e Engenharia de Software.

2. Justificativa

Conforme apontado na introdução por autores como [Gomes et al. 2008b], ferramentas educacionais voltadas ao ensino de programação têm apresentado resultados positivos, atenuando problemas recorrentes observados entre estudantes iniciantes. Essas ferramentas surgem de formas diversas, abrangendo desde *softwares* que tornam mais acessível a compreensão de lógicas computacionais até metodologias de ensino baseadas em abordagens inovadoras. Entre os diversos exemplos de ferramentas como essas temos o jogo de RPG MAKER XP *Wu's Castle*, desenvolvido por [Eagle and Barnes 2008]; o sistema VR-OCKS, um jogo de realidade virtual desenvolvido para auxiliar no ensino de conceitos básicos de programação, desenvolvido por [Segura et al. 2020]; a utilização de robôs de baixíssimo custo para o auxílio do ensino de programação e das fundações de circuitos, desenvolvido por [Pérez and López 2019]; a criação do Jelliot, um programa para a visualização de códigos desenvolvido para programadores iniciantes por [Kirby et al. 2010]; e a ferramenta chamada "The Meadow", um jogo 3D desenvolvido para auxiliar no ensino de programação para estudantes de Animação Digital desenvolvido por [Anderson and McLoughlin 2007]. É importante destacar que ferramentas como essas não são apenas desenvolvimentos recentes: no ano 2000 foi desenvolvido o VINCE [Rowe and Thorburn 2000], um programa *debugger* que, conforme destacado pelos autores, "permite o rastreamento da execução de um programa ao nível de uma

operação única", capacidade que os próprios autores ressaltam como "uma visão mais detalhada das operações de um programa do que um debugger típico". Outro exemplo desses esforços, talvez ainda mais significativo é o de "Karel the Robot", desenvolvido por [Pattis 1994], um trabalho de grande destaque que auxiliava estudantes a partir de uma interface visual e interativa, e que serviu como base para outros trabalhos, como o de [Anderson and McLoughlin 2007].

Dado panorama educacional, considerando as contribuições de iniciativas anteriores e tendo em mente as barreiras e complexidades enfrentadas no aprendizado de programação, bem como os efeitos positivos de ferramentas como as anteriormente citadas em auxiliar o desempenho e a compreensão dos conceitos por alunos, deu-se origem à ideia da elaboração de uma ferramenta direcionada a novos estudantes com dificuldades na aprendizagem de programação. Inicialmente focada em abordar apenas estruturas condicionais, no intuito de auxiliar estudantes a transformar situações cotidianas em estruturas de código compreensíveis, a proposta desse trabalho foi eventualmente ampliada para incluir explicações e a criação de exemplos de estruturas como variáveis e estruturas de repetição — os laços *while* e *for* — e facilitar a compreensão dessas estruturas. Uma vez que é necessária a exibição de códigos gerados através da interatividade dos alunos com a plataforma, e considerando que esse código deve ser intuitivo, claro e acessível ao público-alvo, optou-se pela utilização da linguagem Portugol, definida por [Rubek 2002] como "(...) é uma pseudolinguagem que permite ao programador pensar no problema em si e não no equipamento que irá executar o algoritmo". Sendo amplamente utilizada em cursos de programação — principalmente em suas etapas iniciais — por apresentar comandos em português e priorizar a resolução do problema em detrimento de aspectos sintáticos complexos, essa linguagem foi considerada ideal para esse trabalho, uma vez que o foco está não em ensinar o aluno a programar em uma linguagem específica, mas em auxiliá-lo a entender a lógica de programação que poderá futuramente ser aplicada a qualquer linguagem de sua escolha.

3. Objetivos

O objetivo do presente trabalho consiste no desenvolvimento, teste e posterior disponibilização de uma plataforma virtual (*website*) focada em auxiliar alunos e interessados que estão tendo seu primeiro contato com a área da programação a compreenderem alguns dos elementos essenciais da área. Nessa plataforma, temos como foco três pontos, que podemos considerar como pilares da plataforma: o conceito e a utilização de variáveis, de estruturas condicionais (*if*) e de estruturas de repetição (laços *for* e *while*). Há de fazer notar que existe outra estrutura condicional possível não contemplada nessa plataforma, o *switch-case*. Uma vez que essa estrutura é menos utilizada, mais diferenciada quanto a sua elaboração e implementação, e que a plataforma se foca em apresentar de forma simples os conceitos principais das estruturas abordadas, optou-se por não incluir o *switch-case* para evitar confundir os usuários.

No interesse de apoiar os usuários dentro dos três pilares estabelecidos, o objetivo do site é permitir que o usuário crie expressões de forma natural, usando o português coloquial, enquanto guia o usuário a formular suas frases de maneira similar ao utilizado durante a elaboração das estruturas. Dessa forma a intenção é que o usuário simultaneamente entenda e pratique a elaboração dessas estruturas, entendendo a lógica por trás das mesmas de forma que consiga elaborá-las por conta própria no futuro.

Ao mesmo tempo em que a plataforma tem o interesse de auxiliar alunos, é importante ressaltar que ela evita ser uma "muleta" que restringe a evolução acadêmica e profissional dos mesmos. Em outras palavras, embora a plataforma tenha o objetivo de ajudar o estudante a entender como funcionam determinadas estruturas, a intenção é que seja o estudante a chegar a essa resposta, não que ela lhe seja dada.

Esse fundamento foi uma das principais razões em se optar por apresentar o código das estruturas na pseudolinguagem **Portugol**, dado a forma como ela contribui aos fins da plataforma de múltiplas formas. Os exemplos são diversos: maior facilidade de compreensão devido ao uso de comandos em português, facilitando o entendimento do seu funcionamento por alunos que não entendam ou tenham compreensão limitada do inglês; facilidade de conversão, uma vez que o Portugol utiliza uma estrutura muito semelhante à apresentada por várias das linguagens de programação mais utilizadas na atualidade, como *Java*, *Python* e *C#*; e o próprio fato de *demandar* uma conversão se apresenta como uma vantagem nesse caso: uma vez que o Portugol, sendo uma pseudolinguagem, não é de fato utilizada em aplicações comerciais ou mesmo na maioria das atividades acadêmicas, o estudante se vê forçado a analisar a resposta oferecida e adaptá-la às suas necessidades — incentivando, assim, a compreensão da lógica das estruturas.

3.1. Objetivos específicos

- Oferecer explicações sintetizadas, em linguagem clara e não técnica, sobre o funcionamento e a utilização de elementos da programação como variáveis, estruturas condicionais e estruturas de repetição, através de *tooltips* (pequenas caixas com explicações disponibilizadas a partir de interação com elementos da tela) e *drop-downs* (caixas de texto que originalmente apresentam apenas um título e exibem o texto por completo ao serem ativadas por clique);
- Elaborar estruturas de *inputs* que permitam aos usuários experimentar a criação dos elementos supracitados;
- Oferecer exemplos de código de acordo com as definições passadas por usuários nos *inputs*;
- Validar corretamente os *inputs* inseridos pelos usuários, retornando mensagens de erro claras e concisas conforme necessidade;
- Oferecer ferramentas para customizar ativação e duração das *tooltips* de acordo com preferências do usuário.

4. Fundamentação Teórica

Como apontado anteriormente, as barreiras educacionais relativas ao ensino de programação são claras, com o desenvolvimento e emprego de diversas ferramentas no objetivo de facilitar esse processo provendo resultados positivos. Consequentemente, o desenvolvimento desse tipo de ferramentas torna-se comum, sendo essas criadas para auxiliar o processo educacional de diversas formas, inclusive a partir de funcionalidades como a geração de código, similares às que o presente trabalho apresenta.

Dentre essas ferramentas, uma das mais atuais e de maior relevância é o *Github Copilot* [Github 2022]. Criada pelo Github, a ferramenta utiliza de uma inteligência artificial treinada por bilhões de linhas de código. Ela sugere linhas ou mesmo inteiras funções de código em tempo real, transformando a linguagem natural do usuário em código fun-

cional em diversas linguagens — em outras palavras, traduzindo o ”pensar”humano, o raciocínio comum de como enxergamos condições e consequências do nosso dia-a-dia, para linguagem programática. Embora seja a mais famosa, contudo, o *Copilot* não é a única ferramenta de seu tipo: a *Tabnine* [Tabnine 2022] possui um funcionamento similar ao do *Github Copilot*, oferecendo sugestões de código ao usuário e podendo inclusive sugerir uma linha de código enquanto o desenvolvedor escreve essa linha; já a ferramenta *Captain Stack* [Captain Stack 2022], um *plugin* disponível para *VSCode*, tem um funcionamento similar, embora diferenciado por usar a ferramenta de pesquisas Google ao invés de uma inteligência artificial, enviando uma pesquisa para o Google, extraindo as respostas do *StackOverflow* e *Github Gist* para então completar o código para o usuário; além de ferramentas ainda em desenvolvimento, como o *CodeWhisperer* da Amazon [Amazon 2022], que intenciona competir com o Copilot, executando as mesmas funções da ferramenta com uma inteligência artificial treinada a partir de repositórios de código aberto, repositórios internos da Amazon, documentação de API's e fóruns.

O que difere a atual proposta dessas várias ferramentas pode ser resumido a dois pontos-chave: o primeiro, bastante simples, trata-se do fato de que todas as ferramentas supracitadas são desenvolvidas em inglês, sem suporte atualmente para processar e elaborar sugestões com base em código ou comentário escrito em português. Embora seja importante que desenvolvedores possuam ao menos familiaridade com o idioma ao ponto de conseguirem escrever e entender código em inglês, deve-se considerar que muitos estudantes em seu primeiro contato com a programação ainda não dominam o idioma, tornando a utilização de ferramentas disponíveis apenas em inglês muito mais difícil. Dessa forma, uma ferramenta em português representaria um facilitador, eliminando toda uma faceta de dificuldade que os estudantes teriam em sua utilização.

O segundo ponto, também simples, é que essas ferramentas foram desenvolvidas pensando em desenvolvedores experientes, com foco em otimizar seu trabalho e poupar tempo ao evitar que tenha de digitar estruturas e blocos de código que a inteligência artificial pode gerar. Embora muito úteis para desenvolvedores, isso representa um problema para iniciantes; ainda que o código proporcionado seja completamente funcional e possa ser gerado por linguagem natural, ele é apenas gerado ou completado para o usuário, sem envolver o usuário em seu processo criativo. Em outras palavras, as ferramentas apresentadas auxiliam um desenvolvedor experiente que já sabe o que está fazendo, sem foco na parte didática. A ferramenta proposta nesse trabalho, a *PortugIF*, visa integrar o usuário no processo de criação de código, utilizando da seleção de termos-chave e da montagem de frases para guiar o usuário ao pensamento lógico necessário para desenvolver essas estruturas. Assim, embora ofereça o código pronto em Portugol, o produto que a *PortugIF* visa desenvolver é o próprio raciocínio lógico do aluno, o aprendizado didático-prático que facilitará a compreensão das estruturas por parte do mesmo.

Além da fundamentação prática, a proposta da *PortugIF* pode ser relacionada a teorias educacionais consolidadas. A Teoria da Aprendizagem Significativa, proposta por Ausubel, destaca que novos conceitos são mais facilmente assimilados quando se conectam a estruturas cognitivas já existentes no aluno [Ausubel et al. 1978]. Ao permitir a construção de estruturas de controle por meio de uma linguagem próxima do cotidiano, a ferramenta facilita essa assimilação. Complementarmente, os princípios da teoria sociocultural de Vygotsky evidenciam o papel das ferramentas como mediadoras no pro-

cesso de desenvolvimento cognitivo [Vygotsky 1978]. Assim, ao guiar o estudante na formulação lógica de comandos, a *PortugIF* atua como um instrumento de mediação entre o conhecimento prévio e os novos conteúdos de programação.

5. Metodologia

5.1. Ferramentas Utilizadas

Durante o processo de construção da plataforma, um ponto de grande importância foi a definição de que tipo de tecnologias utilizar na mesma. Inicialmente, considerou-se a tecnologia *JHipster* [JHipster 2022] foi considerada para gerar o código-base, interligando *front-end* e *back-end*, porém essa ideia foi descartada dada a consideração de que uma estrutura de *back-end* seria desnecessária, considerando a simplicidade que se aspira oferecer à plataforma. Após as devidas considerações, optou-se pelas ferramentas a seguir:

Figura 1. Logo do *framework Angular*.

Angular, versão **13.3.5**

Um *framework JavaScript* de código aberto, escrito em *TypeScript*, o Angular (Figura 1) representa uma das plataformas mais utilizadas para a construção de códigos de *front-end* na atualidade. Mantida pelo Google, sua principal função para esta plataforma está na construção de SPAs (*Single Page Applications*), ou seja, aplicações focadas em apresentar seu conteúdo para o usuário em uma única página, com trechos diferentes do conteúdo sendo separados em componentes e "alinhados" na página conforme a necessidade do desenvolvedor.

Figura 2. Logo da ferramenta *Git*.

Git

Conhecida mundialmente e considerada um parâmetro para controle de versionamento, o Git (Figura 2) é um sistema de controle de versionamento gratuito e de código, utilizado por desenvolvedores de *software* ao longo de todo o mundo. Com essa ferramenta, o desenvolvedor consegue enviar uma cópia do projeto no qual está trabalhando para um repositório virtual. Envios ao repositório são feitos por meio de *commits*, versões de código, permitindo a criação de uma biblioteca de versões que não só possibilita o compartilhamento do código-fonte e a sua manutenção em ambiente virtual, mas também permite que o usuário navegue livremente por diferentes versões do código, ajudando também na resolução de *bugs* ou demais problemas que ocorreram ao longo do desenvolvimento.

Como mencionado, o Git permite que o desenvolvedor envie uma cópia de seu projeto para um repositório virtual. Entre as opções disponíveis, o GitHub (Figura 3) é

Figura 3. Logo da plataforma *GitHub*.

GitHub

uma das plataformas mais conhecidas, oferecendo recursos que facilitam o controle de versões e o trabalho colaborativo em equipe.

Além disso, o *GitHub* também conta com o recurso do *GitHub Pages*, tornando a ferramenta ainda mais atrativa pela possibilidade de hospedar de forma simples e gratuita a ferramenta para que usuários possam testá-la e validar seu funcionamento.

Figura 4. Logo do *framework Bootstrap*.

Bootstrap, versão **5.2.2**

O Bootstrap (Figura 4) trata-se de um *framework web* gratuito e de código aberto utilizado para auxiliar o desenvolvimento de aplicações *front-end* usando HTML, CSS e JavaScript. Trazendo uma enorme gama de estilizações pré-programadas e permitindo ampla customização por parte do desenvolvedor, a ferramenta é de grande auxílio na estilização, alinhamento e *design* geral de páginas da web, poupando tempo e esforço por parte dos desenvolvedores que a utilizam.

A escolha do Angular deveu-se à familiaridade e à experiência prévia do autor com a tecnologia. O Git, por sua vez, foi adotado por ser amplamente utilizado na indústria e por permitir o controle de versões e o armazenamento seguro do código, garantindo acesso mesmo em caso de falhas locais. Dentre as plataformas de hospedagem compatíveis, optou-se pelo GitHub, também por afinidade prévia do autor, que já o utilizara em outros projetos e o considerou adequado às demandas deste trabalho. Por fim, o Bootstrap foi empregado para facilitar a organização visual e a estilização dos elementos da interface, reduzindo o tempo necessário para ajustes manuais em CSS por meio de seus componentes pré-definidos

5.2. *Design* da Plataforma

A aplicação foi concebida como uma SPA de *design* simples e minimalista, com foco em fornecer ao usuário explicações claras sobre seu uso, sem armazenar dados pessoais ou registros de navegação. Sua estrutura segue o Princípio KISS (*Keep it Simple, Stupid!* — "Mantenha Simples, Estúpido!"), que valoriza a simplicidade como meio de promover maior aceitação e interatividade por parte dos usuários [Rahman 2022][Smith 2012][Kreutz et al. 2017][Yunanto et al. 2024]. Considerando que o público-alvo são iniciantes em programação, a adoção desse princípio é vista como essencial para facilitar a compreensão das estruturas lógicas apresentadas. Por isso, todo o desenvolvimento priorizou uma experiência direta: uma única página com

elementos bem definidos, permitindo ao usuário realizar rapidamente as consultas desejadas, ajustar preferências mínimas e retornar ao seu foco de estudos, sem distrações.

Ao longo da única página na qual a plataforma é disposta, foram criadas seções relevantes para a elaboração dos elementos abordados pela plataforma: as variáveis, estruturas condicionais e estruturas de repetição. Para cada um deles, foi pensada uma estrutura adequada que permitisse uma experiência de criação simples para os usuários, enquanto simultaneamente os informam sobre as condições do funcionamento dessas estruturas. Dentre elas, as estruturas de repetição foram abordadas de forma separada em *while* (Figura 5) e *for* (Figura 6), de forma a poder oferecer exemplos mais variáveis e ilustrar melhor as diferenças entre ambas, que são comumente utilizadas e cumprem propósitos similares de formas diferenciadas.

Figura 5. Seção de estruturas de repetição *while*.

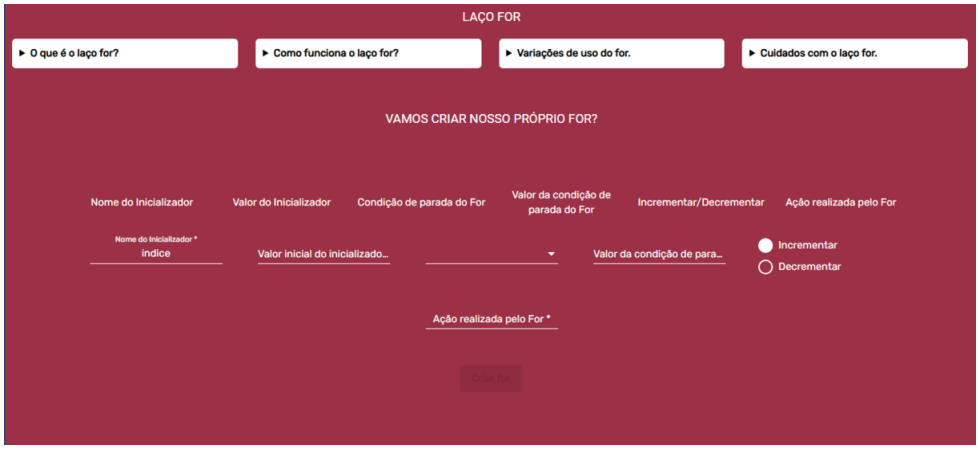

Figura 6. Seção de estruturas de repetição *for*.

Dado seu caráter educativo, a plataforma possui um volume considerável de informações sobre o funcionamento das diferentes estruturas abordadas. Para evitar sobrecarregar o usuário com informação demais, optou-se na plataforma por uma abordagem capaz de suprir informações ao usuário conforme demanda. Isso se deu por meio do uso de caixas de texto *dropdown* (Figura 7) acionadas pelo clique, assim como a

utilização de *tooltips* (Figura 8) apresentadas apenas após o usuário mover o *mouse* sobre pontos específicos — geralmente títulos ou *labels* — para oferecer mais informações sobre o que determinado elemento significa. Ao se adotar esses padrões, a intenção é de ganho duplo: primeiro em uma maior elegância e usabilidade da plataforma ao se evitar a poluição visual de grandes quantias de texto desnecessário na tela; e em segundo ao oferecer maior conforto ao usuário por meio da disponibilização de conteúdos conforme sua necessidade, permitindo que o estudante estabeleça seu próprio ritmo e acesse esses conteúdos conforme a sua vontade sem se sentir intimidado ou sobrecarregado pelo volume de informações.

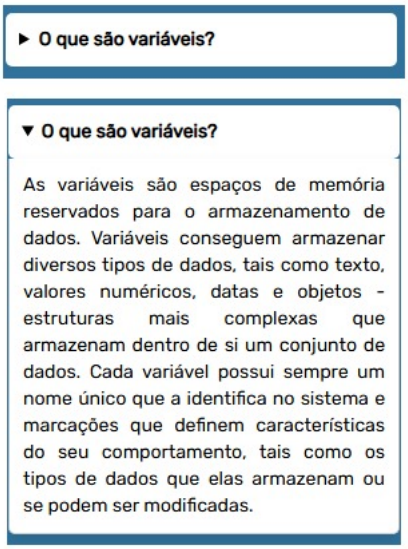

Figura 7. Exemplo de *dropdown* aberto e fechado.

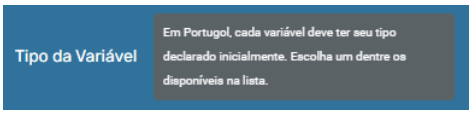

Figura 8. Exemplo de *tooltip* disponibilizado após passar o *mouse* sobre *label*.

Para estabelecer familiaridade entre o usuário e a plataforma e minimizar qualquer confusão que o mesmo possa ter ao acessá-la, foram feitos esforços para estabelecer padrões sempre que possível na forma como as diferentes estruturas são abordadas. Em termos gerais, para os elementos estabelecidos segue-se o seguinte padrão: primeiro são apresentados quatro *dropdowns* explicando o que cada elemento é e detalhes de seu funcionamento, sendo que cada *dropdown* conta com títulos diretos e intuitivos, tais como: "o que são *x*", "como *x* é utilizado" ou "variações do uso de x". Dentro dessas *dropdowns*, o texto disponibilizado segue uma escrita polida e casual, priorizando a clareza da comunicação, para apresentar as informações de forma que o usuário assimile tenha o mínimo de dificuldade com a interpretação do texto. Após isso, estruturas de formulário são apresentadas, utilizando campos de texto livre, seletores de opções e botões

de opções radial (*radio button*) para o usuário escolher uma entre duas opções. Esses campos são apresentados com *placeholders* e, em alguns casos, valores *default* conforme necessário, e seus títulos ou *labels* possuem *tooltips* destinadas a esclarecer ao usuário dúvidas que possam ter sobre como os campos funcionam. Após o preenchimento correto das informações necessárias nesse formulário, um botão para gerar a estrutura em questão (variável, *if*, *while* ou *for*) é disponibilizado, e um pequeno trecho em Portugol é exibido após o clique, ilustrando a estrutura conforme os valores definidos pelo usuário.

Na plataforma, apenas dois dos quatro elementos principais apresentam uma estrutura mais complexa do que essa. A primeira é a estrutura condicional, a qual dispõe de uma tabela que permite maior customização por motivos abordados na próxima seção. A outra é a estrutura de variáveis: dada a relativa simplicidade de uma variável e o quão essencial é esse elemento para a programação, optou-se por implementar uma estrutura que permite ao usuário não só gerar uma variável, como também validar seu valor, enviando um *input* que será verificado pela plataforma e retornará uma mensagem ao usuário dizendo se o valor do *input* é ou não igual ao inserido na variável (Figura 9).

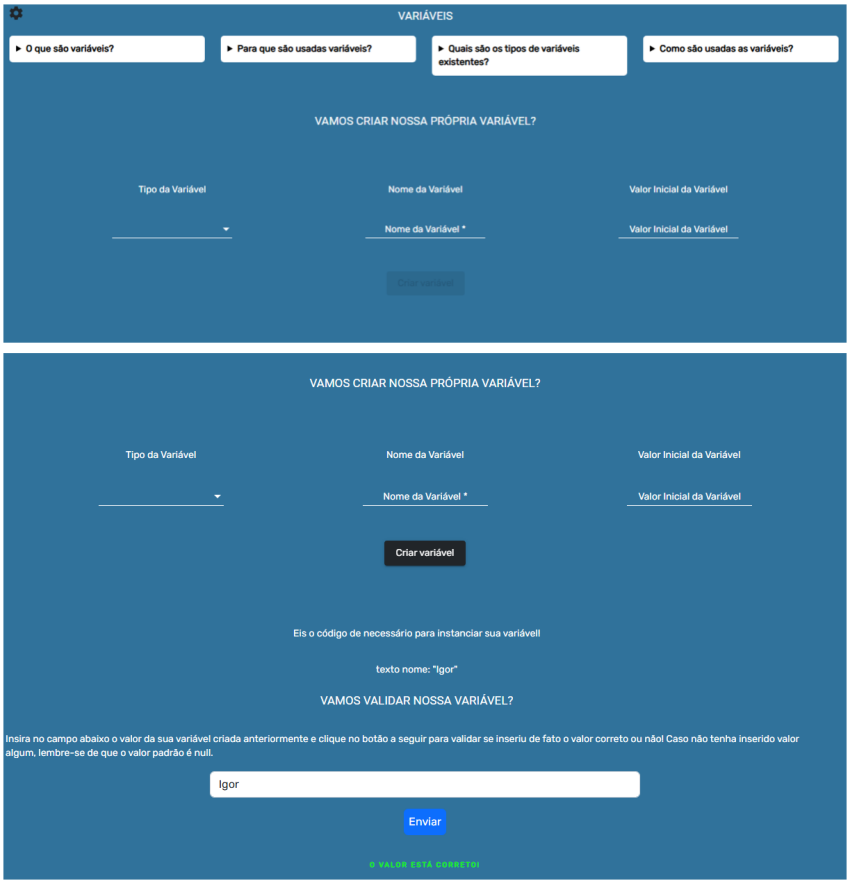

Figura 9. Estrutura de variáveis inicial e com a validação de valor da variável acionada.

Nesse contexto, é relevante reforçar também a característica da plataforma como um auxiliar no aprendizado dos usuários para que melhor entendam como as estruturas funcionam, mas frisar que a mesma não se trata de uma forma com o interesse ou objetivo de oferecer "respostas prontas" ao usuário. A construção de trechos de código de variáveis, *if*, *while* ou *for* é dada para indicar ao usuário como os elementos são organizados e disponibilizados nessas estruturas, mas não é do interesse da plataforma oferecer códigos que de fato realizem uma determinada ação para o usuário. Se, por exemplo, o usuário deseja um código que imprima o resultado da soma de 2 + 2 caso uma determinada condição seja atingida (ou seja, por meio de uma condicional *if*), é de interesse da plataforma tão apenas elucidar para o mesmo a estrutura que essa condicional teria para alcançar seu objetivo, mas não fornecer o código que de fato faria a impressão dessa soma. É por esse motivo que, em trechos onde o usuário elucida o que será feito ("ação realizada pelo *for*", por exemplo), a plataforma se limita a posicionar um "escreva" com o texto da ação dentro de si. Cabe ao usuário então implementar sua própria lógica para realizar a ação em questão, substituindo o "escreva" — que atua, efetivamente, como um simples *placeholder* — por ela.

Essa decisão, tal como a de oferecer conteúdo ao usuário sob demanda, sustenta-se sobre duas justificativas: primeiramente, existem infinitas possibilidades de ação que o usuário pode executar por meio de código, e compreender e representar todas elas nessa plataforma seria exageradamente complexo e fora de linha para com o propósito dela, avançando muito além do escopo do trabalho. A segunda e maior motivação, contudo, é conservar um nível de independência por parte dos usuários. Ao oferecer códigos completos que executem de fato tudo que o usuário deseja, este muitas vezes pode criar uma relação de dependência com a ferramenta que o leve a deixar que ela crie códigos em seu lugar, coibindo o desenvolvimento pessoal do usuário. Como ferramenta auxiliar, é de interesse da plataforma que o usuário a use para sanar dúvidas, entender conceitos e orientar-se sobre o funcionamento de estruturas, mas tudo isso enquanto ele se mantém o principal responsável pelo código, implementando suas próprias soluções de forma que exercite sua capacidade de pensamento lógico e mantenha sua evolução profissional.

Dessa forma, a plataforma deve ser vista como uma ferramenta facilitadora para aqueles que possuem dificuldade em compreender a lógica ou organização de uma das estruturas supracitadas, mas não como uma ferramenta para disponibilizar, aos usuários, trechos de código que façam o que ele deseja, com base em suas instruções, como o Copilot.

5.3. Desenvolvimento

Como dito anteriormente, em um momento inicial a proposta do presente trabalho era tanto mais limitada. Planejado originalmente como um trabalho para a criação de uma plataforma destinada a auxiliar o usuário a criar códigos de *if* em Portugol, o desenho inicial da plataforma foi mantido nesse escopo até a conclusão desse módulo. Ao finalizá-lo, contudo, optou-se pela expansão do escopo para uma plataforma capaz não apenas de auxiliar na criação de estruturas condicionais, mas também de laços *while*, *for* e variáveis em geral. A estrutura *if* (Figura 10), no entanto, manteve um *design* mais complexo e completo do que as demais, propositalmente, uma vez que estudos apontam as estruturas condicionais como uma das que apresenta maior obstáculo para novos estudantes, além da necessidade de compreender bem como funciona uma estrutura condicional para que

se possa aplicar esses conceitos em estruturas de repetição. Dessa forma, diferentemente das demais estruturas da plataforma, a estrutura *if* é organizada ao longo de uma tabela, permitindo uma customização mais minuciosa e favorecendo a melhor compreensão por parte do usuário.

Figura 10. Estrutura *if* completa, com e sem a geração do código.

Durante o desenvolvimento, buscou-se sempre preservar a simplicidade, especialmente no que diz respeito à facilidade de uso e à compreensão, por parte do usuário, das ações executadas e do comportamento do sistema. A adoção do modelo SPA seguiu essa lógica: ao concentrar todos os recursos em uma única página contínua, pretende-se facilitar o fluxo de leitura e o aprendizado dos conceitos abordados. Esse mesmo princípio motivou a simplificação ou exclusão de certos elementos, a fim de evitar confusões decorrentes de sobrecarga informacional. Considerando que o excesso de opções poderia gerar mais dúvidas do que soluções, optou-se por restringir o escopo às quatro estruturas fundamentais já mencionadas. Elementos como o *switch-case* foram deixados de fora por apresentarem sintaxe mais específica e aplicação limitada, dando lugar a construções mais frequentes e acessíveis aos iniciantes.

Essa mesma justificativa se aplica a variações como o *for-each*, que, por se tratar

de um desdobramento da estrutura *for*, pode causar confusão para usuários que estejam tendo seu primeiro contato com o conceito. Optou-se também por limitar certas características de uso avançado, como incrementos personalizados no *for*, mantendo-o restrito a incrementos e decrementos unitários. Manipulações mais complexas podem gerar dúvidas e erros capazes de frustrar, desmotivar ou inibir o aprendizado. Sendo a PortugIF voltada a um público iniciante, o foco foi mantido em apresentar conceitos fundamentais de forma acessível, conduzindo o usuário a um raciocínio lógico que favoreça o aprendizado e desperte o interesse por aprofundamento. Assim, a introdução de elementos avançados nessa etapa se torna contraproducente. Ainda que certos conceitos sejam mencionados — para familiarização e incentivo à pesquisa —, não são tratados diretamente no escopo da plataforma.

Uma vez estabelecidos os elementos principais da página na forma das estruturas de ensino padronizadamente, o próximo passo no desenvolvimento foi o estabelecimento de uma *sidebar* para configurações. Uma vez que a plataforma foi desenvolvida para que usuários inexperientes em programação pudessem recorrer ao seu apoio, é importante considerar a possibilidade de dificuldades na leitura e compreensão dos textos explicativos disponibilizados por meio de *tooltips*. As configurações disponíveis na *sidebar* — visíveis na Figura 11 — permitem que o usuário defina o tempo de duração das *tooltips*, ou mesmo mude o seu acionamento para que se tornem visíveis ou desapareçam conforme o clicar do *mouse*, ao invés de acionadas ao mover o *mouse* sobre a sua região, conforme a preferência do usuário. Sendo disponibilizadas apenas dentro da *sidebar*, essas configurações também permanecem ocultas até que o usuário decida acessá-las, evitando poluição visual que possa acarretar distrações.

Figura 11. Componente *sidebar* aberto para configurações.

Com as estruturas de programação definidas e a adição de configurações para customização dos *tooltips*, a principal etapa do desenvolvimento da plataforma em si deu-se por concluída. A partir desse ponto foram focados esforços em melhorar a visibilidade e usabilidade da plataforma; isso se traduz na criação de um elemento misto que funciona simultaneamente como *header* e *navbar*, visto na Figura 12, permitindo ao usuário se mover rapidamente por cada uma das seções do código rapidamente. Investiu-se tempo, também, na melhoria do *design* e *layout* de diversos elementos da página, não limitado a elementos auxiliares como *sidebar* e *footer*, mas envolvendo toda a página na busca de um *design* que mantenha a simplicidade e apresente os elementos de forma elegante e

coerente.

Figura 12. *Header* da plataforma.

Entre esses ajustes, o principal foi a paleta de cores da plataforma. Originalmente foi considerado manter a plataforma em uma paleta simples de fundo branco e texto preto para corresponder à ideia de simplicidade norteadora da *PortugIF*, opções de conjuntos de cores foram exploradas e, eventualmente, optou-se pela utilização de um tom de verde para o *header* — presente por toda a extensão da plataforma — e por uma alternação entre tons de azul e vermelho para o "corpo" da plataforma em si, com tons de brancos para o texto. A escolha por essa opção de cores foi motivada por três fatores, o primeiro sendo a vinculação do projeto à imagem e nome do IFMG, levando a optar por um tom de verde similar ao utilizado na identidade visual do instituto (Figura 13).

Figura 13. Paleta de cores utilizada e logo do IFMG.

O segundo fator para essa seleção foi, em meio a pesquisas, a identificação de uma constante presença e associação das cores verde, vermelho, azul e amarelo em empresas de tecnologia, tal como a *Google* ou a *Microsoft* (Figura 14) [Breindel 2024].

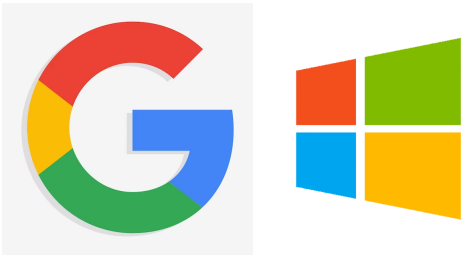

Figura 14. Logos das empresas Google e Microsoft.

Já o terceiro fator é uma referência ao modelo de cores *RGB* que não só é amplamente utilizado na área tecnológica e na programação, mas cujas três cores principais possuem também boa correlação, sendo complementares segundo a teoria das cores [Britannica 2024][Baird 2015][Aidar 2024].

Em adição às alterações das cores para melhor *design*, foram realizadas diversas alterações menores para melhorar a visualização da plataforma. Essas vão desde a mudança da fonte usada, até ajustes no CSS e na organização do HTML da plataforma para resolver problemas de quebra de *overflow* da página, e melhorar a visualização de texto dentro dos *dropdowns* criados.

Ao fim, foi adicionada uma última seção de *Perguntas Frequentes* à plataforma para oferecer respostas objetivas a perguntas que o usuário pode ter ao acessar a plataforma, tal como a qual público ela é direcionada ou o porquê da opção pela linguagem Portugol (Figura 15).

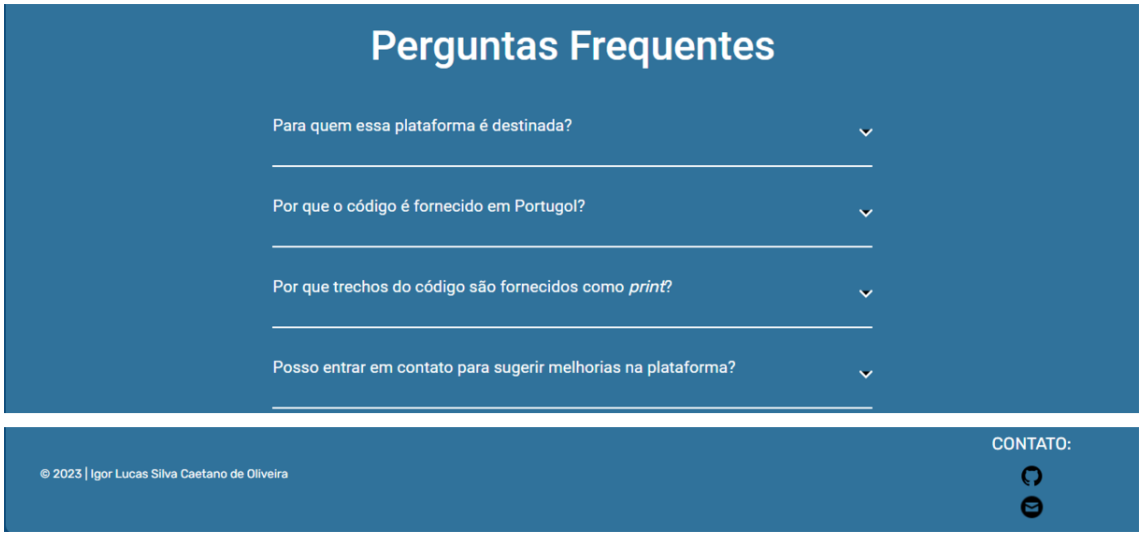

Figura 15. Seção de *Perguntas Frequentes* e *footer*.

5.4. Metodologia de Coleta de Dados

Sendo essa uma plataforma direcionada ao público, uma das etapas de maior importância para o seu desenvolvimento envolve a disponibilização da plataforma para acesso de usuários e coleta de *feedbacks* sobre o funcionamento da mesma. Para esse processo foi escolhido fazer uso da plataforma *Google Forms* devido a uma série de fatores: os recursos que a plataforma oferece, a facilidade de divulgação dos seus formulários, a familiaridade e confiabilidade associadas à marca Google e a gratuidade da ferramenta sendo alguns dos principais motivadores.

No interesse de motivar usuários a responderem o formulário, optou-se por uma estrutura focada primariamente em seleções e escalas. Perguntas de seleção única, múltiplas seleções (Figura 16) ou escalas lineares na estrutura da Escala de Likert (Figura 17) são predominantes no formulário, com opções de "Outros" nas perguntas de seleções para permitir um retorno mais customizado do usuário, caso assim queira, e a Escala de Likert sendo utilizada para avaliar o quanto o usuário concorda ou não com diversas afirmações relativas à plataforma.

Essas estruturas representam quase todos os mecanismos de coleta de dados do formulário. Salvo por uma exceção na forma de uma pergunta que pede como resposta

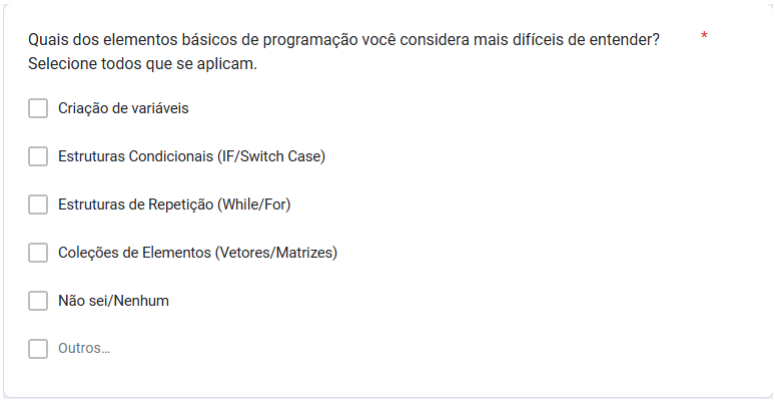

Figura 16. Exemplo de estrutura de múltiplas escolhas.

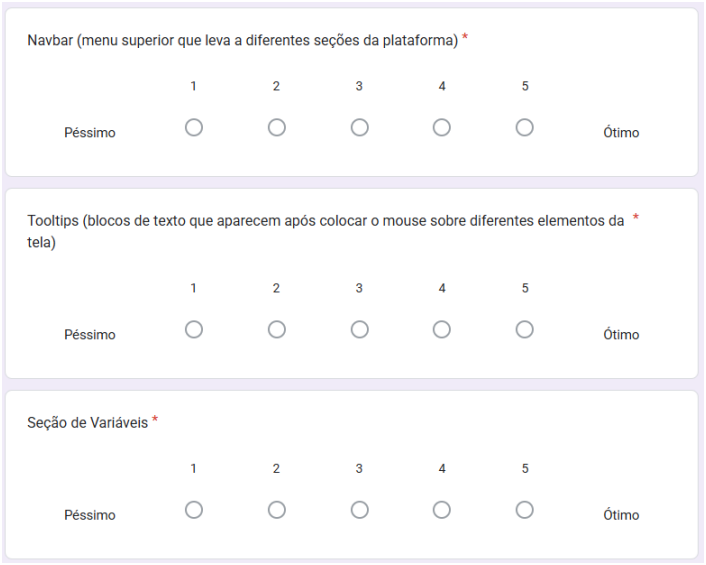

Figura 17. Exemplo de escalas lineares usando a estrutura da Escala de Likert.

um texto curto e um par de campos opcionais no final do questionário, todas as outras perguntas foram estabelecidas de forma objetiva e fazem uso de uma das estruturas acima para simultaneamente direcionar o *feedback* recebido a um nível mais fácil de analisar como também agilizar o processo de resposta do questionário.

A adoção de tantas medidas para simplificar e agilizar o questionário encontra motivação na notória dificuldade em se obter respostas para questionários *online*. Embora a aplicação de questionários ou formulários seja uma forma de reduzir tempo na coleta de dados e atingir um público muito maior do que o normalmente acessível, diversos estudos como os de [Bastos et al. 2023] e [Faleiros et al. 2016] citam a baixa taxa de respostas que esses questionários recebem. Dessa forma, na tentativa de minimizar esse problema, o questionário foi estabelecido pensando na agilidade e simplicidade: ele consiste em apenas sete seções, sendo duas delas informando o usuário de como serão utilizados os dados e coletando sua permissão para fornecimento deles, três com questões de múltiplas escolhas ou escala linear, uma com campos para observações finais e a última consistindo

em um agradecimento ao usuário. O formulário conta com apenas vinte e duas questões no total, sendo duas delas opcionais. Em testes realizados pelo autor, foi constatado um tempo de resposta de cerca de 3 minutos para o formulário; conforme pode se ver na Figura 18 esse tempo é divulgado com destaque logo no início do formulário na intenção incentivar os participantes a seguirem até o fim e, dessa maneira, buscar se obter o máximo de respostas possível.

Figura 18. Trecho introdutório informando o tempo de resposta estimado do formulário.

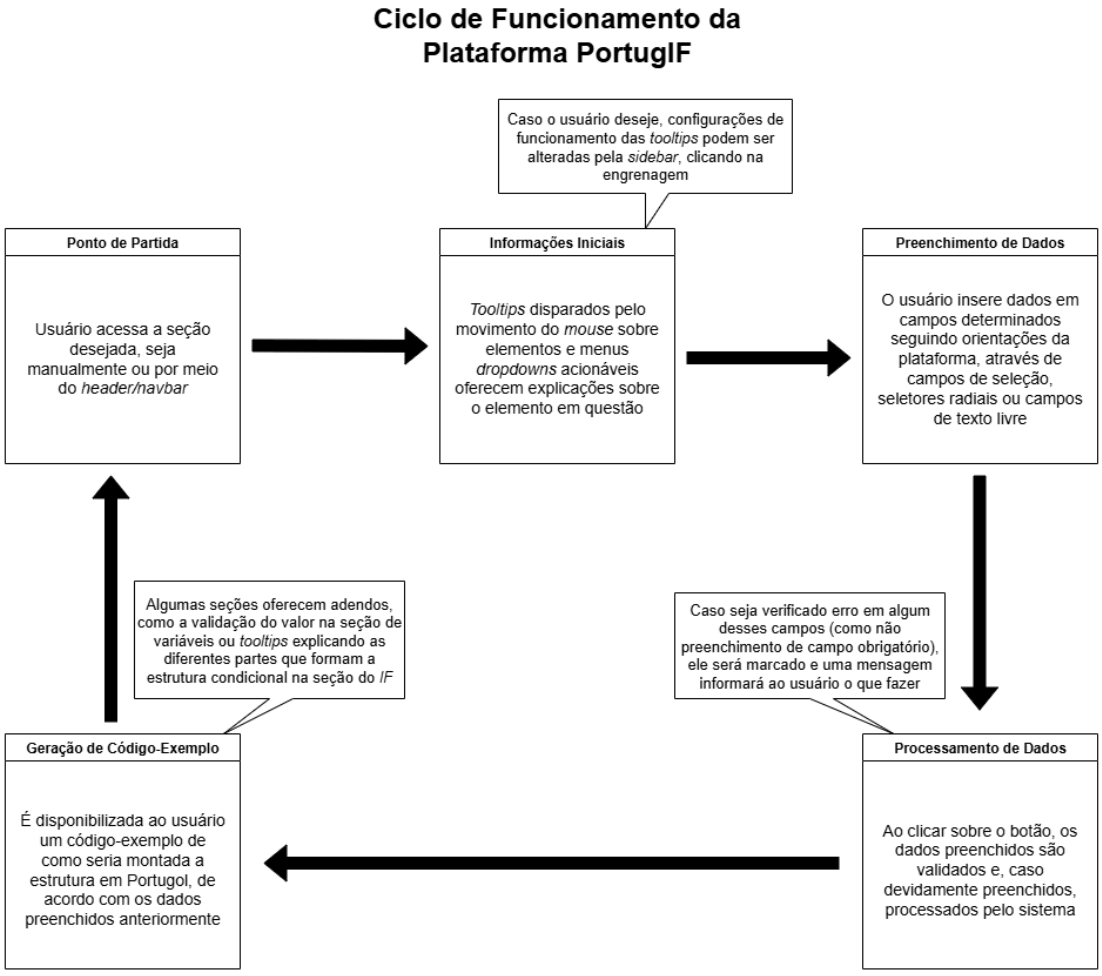

Figura 19. Diagrama ilustrativo do ciclo de funcionamento da plataforma.

A figura 19 mostra uma visão resumida e simplificada de um caso de uso da plataforma, partindo do acesso feito pelo usuário e indo até a disponibilização de um código-exemplo. A partir dela é possível se ter uma ideia do ”caminho”percorrido pelo usuário: primeiro o usuário acessa a seção que deseja, podendo fazer isso manualmente ou usando da ferramenta do *Navbar* para isso, se assim preferir; depois o usuário recebe informações sobre a funcionalidade que está visualizando por meio dos *tooltips* e *dropdowns* disponíveis ali; em seguida, guiado pela organização dos elementos e as explicações dispostas, o usuário preenche os diferentes *inputs* com os dados que deseja; uma vez que todos os *inputs* sejam preenchidos, o botão para criação da estrutura é disponibilizado ao usuário; por fim, ao clicar nesse botão, os elementos preenchidos são validados e o código-exemplo é gerado e fornecido ao usuário. Esse ciclo pode ser repetir quantas vezes o usuário quiser, permitindo a criação de múltiplos exemplos de uma mesma estrutura ou de estruturas diferentes.

6. Resultados

O formulário desenvolvido para a coleta de dados sobre a plataforma foi enviado para o público no dia 12/02/2025, com divulgação tanto por meios pessoais do autor quanto pela

equipe do IFMG — *Campus* Sabará, após solicitação de divulgação da parte do autor e orientador. Como pode-se visualizar na figura 18, este formulário trás em sua introdução o *link* de acesso à plataforma, *link* este também presente no e-mail de divulgação da pesquisa e na distribuição própria feita pelo autor, possibilitando que respondentes acessem a plataforma e testem todas as suas funcionalidades à vontade antes de responderem. Entre essa data e o período de 05/05/2025 foram coletadas um total de 21 respostas diferentes ao questionário. Nesse ponto, cabe ressaltar que, embora esse número seja considerado relativamente pequeno e idealmente uma amostra maior tivesse sido coletada, a incapacidade de se obter mais respondentes fizeram com que se optasse por seguir com a análise contando apenas com essas 21 respostas.

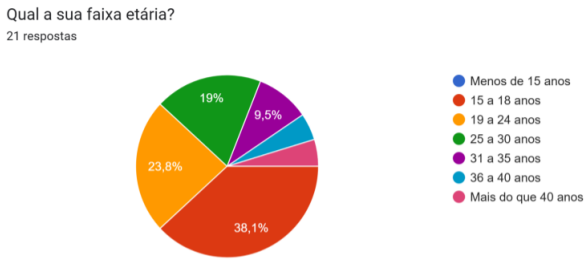

Figura 20. Distribuição da faixa etária dos respondentes.

A maioria dos participantes do questionário pertence a faixas etárias mais jovens: 38,1% têm entre 15 e 18 anos e outros 23,8% estão na faixa de 19 a 24 anos (Figura 20). Isso significa que mais de 60% dos respondentes têm menos de 25 anos. Embora essa informação não implique necessariamente inexperiência com programação — como será discutido adiante —, sugere que o público-alvo da plataforma ainda está em fase de formação e aprendizado, o que reforça a relevância da proposta.

Figura 21. Distribuição da familiaridade dos respondentes com a programação.

Com relação à familiaridade, temos uma noção um pouco diferente, porém ainda próxima da anterior; tal como podemos observar na Figura 21, 38,1% dos respondentes indicam ter estudado programação a nível técnico, enquanto 23,8% afirmam trabalhar com programação em nível profissional e 19% indicam nunca ter estudado ou trabalhado com programação. Embora possamos considerar que aqueles que trabalham na área já estão consideravelmente familiarizados com ela, cabe destacar aqui que quase 60% dos

respondentes possuem o nível de experiência com a área mais desejado entre os respondentes, sendo ou indivíduos que não conhecem a área (e, consequentemente, teriam maiores dificuldades), ou indivíduos cujo nível de conhecimento sugere uma aproximação maior com os conceitos abordados pela plataforma.

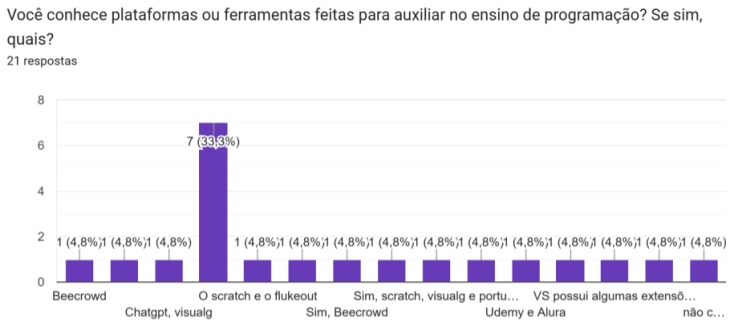

Figura 22. Conhecimento dos respondentes de plataformas auxiliares.

Em relação a conhecerem demais plataformas auxiliares, a maioria dos respondentes indica conhecer pelo menos uma plataforma diferente (Figura 22). Embora um terço dos respondentes afirmem desconhecer outras plataformas, muitos também responderam indicando plataformas como o Scratch, Beecrowd, Chat GPT, VisualG, entre outras. Uma vez que essa resposta foi dada em campo aberto, com a possibilidade dos usuários inserirem as plataformas que conhecem, em uma análise mais minuciosa, foi possível verificar que houveram 3 menções ao Beecrowd, duas menções ao Scratch e VisualG, e apenas uma menção ao Chat GPT — sendo que não necessariamente são menções exclusivas, ou seja, existem casos de respondentes que mencionaram tanto o Scratch quanto o VisualG em suas respostas. Da mesma forma, nem todas as plataformas mencionadas se caracterizam da mesma forma; enquanto o Scratch é uma plataforma no interesse de auxiliar no entendimento da programação, plataformas como a Udemy e Alura — mencionadas em uma das respostas — são plataformas de cursos *online* sobre diversos assuntos, incluindo cursos de programação. Assim, destaca-se que essas plataformas, embora possam ser usadas como auxiliares, existem com a proposta de serem plataformas educativas, não ferramentas auxiliares.

Talvez o resultado mais inesperado entre os respondentes seja que mais de 50% dos participantes indicaram a dificuldade de aprender programação como um nível 3 de cinco — ou seja, mediana — como pode ser observado na Figura 23. Embora parte disso possa se atribuir ao fato de a maioria dos respondentes ter alguma experiência com programação, mesmo entre os que relataram nunca ter estudado a área foram obtidas uma classificação 2, uma classificação 3 e duas classificações 4. Embora as indicações tendam claramente para uma classificação da programação como uma área “difícil”de se aprender ao invés de como uma área “fácil”, a expectativa do autor era de uma classificação geral 4 ou 5, tornando o resultado obtido uma surpresa.

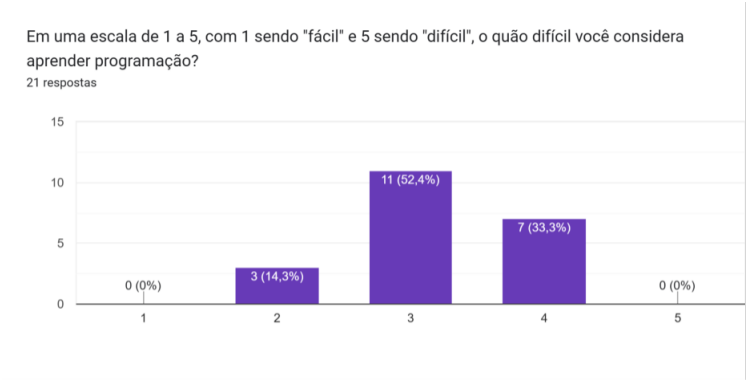

Figura 23. Dificuldade percebida na aprendizagem de programação.

Figura 24. Tópicos de maior dificuldade em programação.

Por outro lado, se a dificuldade relatada pelos respondentes foi abaixo do esperado, a dificuldade indicada quanto aos elementos de programação seguiu o padrão esperado. A grande maioria dos respondentes indicou a criação de coleções de elementos, tais como vetores ou matrizes, como os mais difíceis, seguidas pelos trabalhos com estruturas de repetição, estruturas condicionais e, por último, criação de variáveis, conforme indicado pela Figura 24.

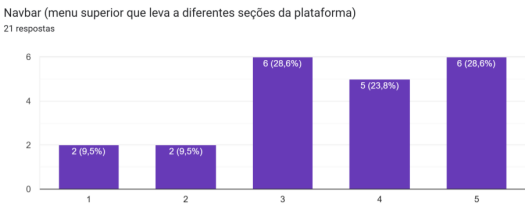

Figura 25. Avaliação dos respondentes sobre Navbar

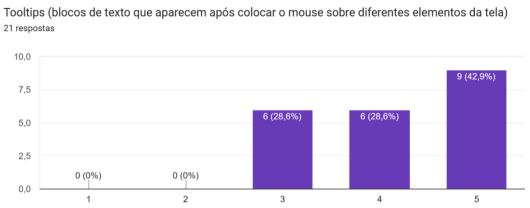

Figura 26. Avaliação dos respondentes sobre Tooltip

Implementadas durante o trabalho de desenvolvimento da plataforma, estruturas como o *Navbar* e os *Tooltips* foram pensadas para aprimorar a experiência dos usuários, tanto em permitir um acesso mais fácil a diferentes seções da plataforma quanto a auxi-

liar em uma melhor compreensão de seus elementos. Assim sendo, foi solicitado que os usuários avaliassem essas funcionalidades da plataforma no relativo ao quão bem cumprem suas funções e ao quão intuitiva é o seu uso. No relativo a isso, observa-se um alto nível de aprovação dos componentes auxiliares — como a *navbar*, que facilita a navegação entre os diferentes blocos da interface, e os *tooltips*, que oferecem explicações adicionais sobre os elementos exibidos na tela (Figuras 25 e 26). Vale destacar que, na versão inicial publicada, a *navbar* apresentava um erro: embora funcionasse corretamente em ambiente local, peculiaridades do *GitHub Pages* faziam com que os links resultassem em páginas de erro 404. O problema só foi identificado após os primeiros relatos dos usuários e, a partir daí, foi corrigido e a versão online foi devidamente atualizada.

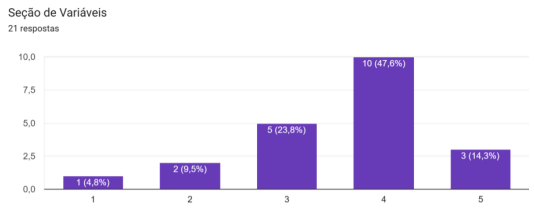

Figura 27. Avaliação dos respondentes sobre Variáveis.

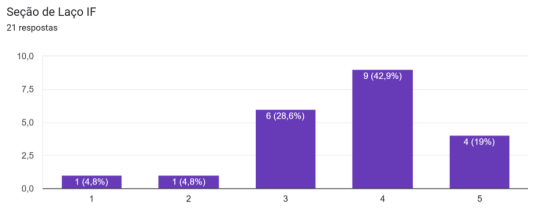

Figura 28. Avaliação dos respondentes sobre estruturas If.

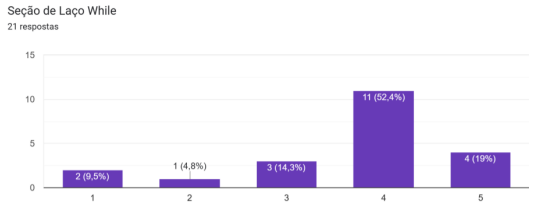

Figura 29. Avaliação dos respondentes sobre estruturas While.

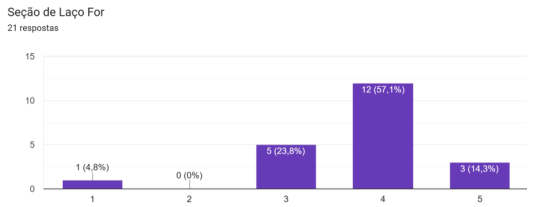

Figura 30. Avaliação dos respondentes sobre estruturas For.

De forma similar ao observado sobre as estruturas auxiliares da plataforma, os componentes principais — isto é, as seções que tratavam sobre explicações de diferentes elementos de programação — tiveram uma alta taxa de aprovação no relativo a cumprirem suas funções e serem intuitivos de utilizar, com dentre elas a de maior aprovação sendo a seção que trata sobre o *for* (Figura 30) enquanto, talvez de forma um pouco paradoxal, a seção que trata o *while* seja aquela com a maior reprovação (Figura 29). Isso dito, todas as seções tiveram altíssima taxa de aprovação, com não mais do que três respondentes (aproximadamente 13% dos respondentes) as classificando de forma negativa e no mínimo treze respondentes (aproximadamente 60%) classificando as seções de forma positiva (Figuras 27, 28, 29 e 30).

No relativo à classificações sobre como o conteúdo é apresentado pela plataforma e como busca alcançar seu objetivo, embora as avaliações observadas continuassem sendo em sua maior parte positivas, observou-se também um número um pouco maior de avaliações negativas. Embora quase metade dos respondentes classifique a plataforma

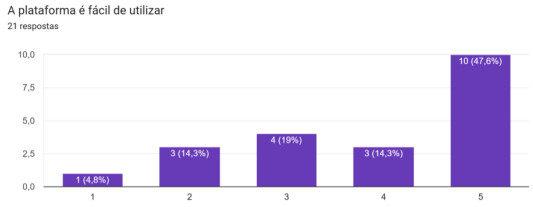

Figura 31. Avaliação dos respondentes sobre a usabilidade da plataforma.

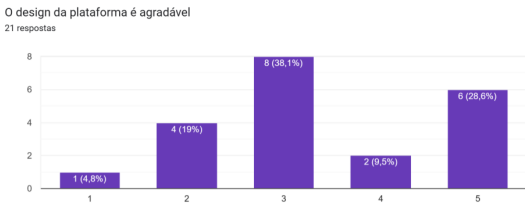

Figura 32. Avaliação dos respondentes sobre o design da plataforma.

como fácil de utilizar (Figura 31), uma quantia significativa (aproximadamente 20%) a considerou de uso difícil. Dentre os pontos avaliados nesse nível, o mais negativo deles é, sem dúvida, o *design* da plataforma, com quase 25% de avaliações negativas e majoritariamente avaliações medianas, conforme indicado pela Figura 32. Embora parte disso possa se atribuir ao meio de acesso à plataforma — ela foi pensada para ser acessada por meio de *desktops* e, consequentemente, usuários que a acessaram por *tablets* ou *smartphones* fizeram uso de uma versão não otimizada da mesma — uma quantia considerável de críticas foi levantada quanto à paleta de cores utilizada na plataforma, com usuários apontando problemas na leitura do texto graças a ela.

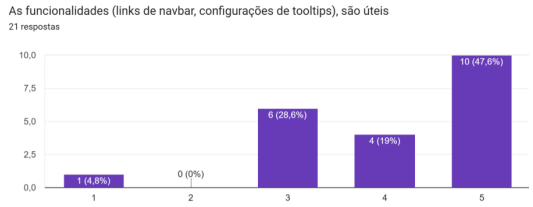

Figura 33. Avaliação dos respondentes sobre as funcionalidades da plataforma.

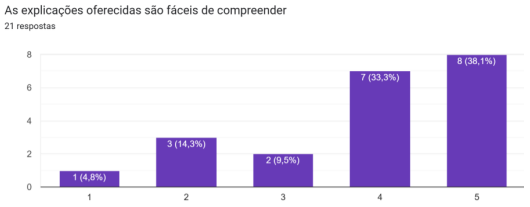

Figura 34. Avaliação dos respondentes sobre as explicações da plataforma.

As classificações de utilidade das funcionalidades da plataforma e das explicações nos *dropdowns* (Figuras 33 e 34) seguem um padrão semelhante: as funcionalidades receberam alto nível de aprovação, o que confirma os resultados já observados na seção anterior. As explicações também foram bem avaliadas, embora com uma taxa de reprovação ligeiramente maior. Apesar da adoção deliberada de uma linguagem mais informal para facilitar a compreensão, uma das sugestões recebidas foi a de ampliar ainda mais esse estilo, o que reforça sua importância na assimilação dos conceitos apresentados.

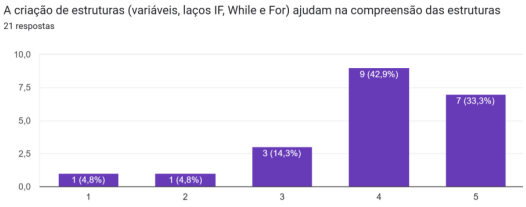

Figura 35. Avaliação dos respondentes sobre a criação de estruturas.

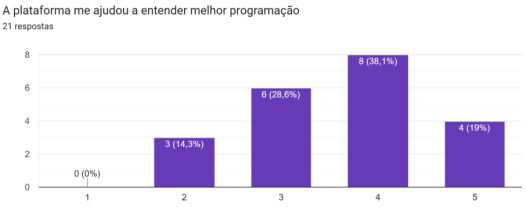

Figura 36. Avaliação dos respondentes quanto a plataforma como auxiliar no ensino.

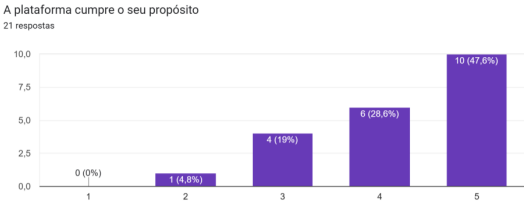

Figura 37. Avaliação dos respondentes quanto a plataforma atingir seu propósito.

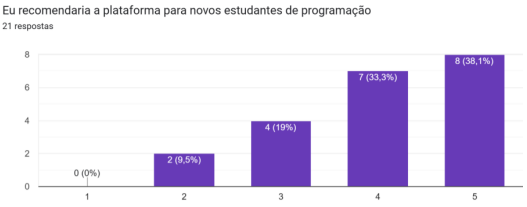

Figura 38. Avaliação dos respondentes quanto à possibilidade de recomendação.

Ademais, nota-se que a grande maioria dos respondentes avalia positivamente a funcionalidade de criação de estruturas da plataforma (Figura 35) e entende que a mesma cumpre o seu papel, auxiliando na compreensão da programação e sendo uma plataforma que recomendariam para novos estudantes, conforme indicado pelas Figuras 36, 37 e 38. Essas afirmações ressaltam o potencial uso da plataforma como um auxiliar no processo educativo; embora uma análise revele que os respondentes sem experiência prévia com programação aprovam da iniciativa da plataforma, os dados revelam também que a grande maioria dos indivíduos já inseridos na área a enxergam como uma ferramenta útil nesse processo.

A Tabela 1 a seguir apresenta um resumo dos principais índices de aprovação coletados por meio do formulário de avaliação, destacando o desempenho dos diferentes componentes da PortugIF conforme a percepção dos usuários.

Tabela 1. Síntese das avaliações dos respondentes sobre a PortugIF

Item Avaliado	**Aprovação (%)**
Compreensão das estruturas IF, FOR, WHILE e variáveis	60–81%
Tooltips e navegação geral da plataforma	$>$75%
Recomendação da ferramenta para outros alunos	$>$80%
Design (paleta de cores, estética visual)	$\sim$75%, com 25% negativa

7. Discussão e Conclusão

De forma geral, uma análise do *feedback* proporcionado pelos participantes da pesquisa revela tanto pontos fortes da plataforma como algumas das suas vulnerabilidades. Tanto como conceito quanto como plataforma, a PortugIF apresenta uma utilidade considerável a ser explorada. Embora os respondentes não tenham elencado tantas dificuldades nos elementos abordados quanto inicialmente esperado e muitos deles tenham apresentado outras ferramentas que utilizam como auxiliares na programação, a maioria dessas são ferramentas de propósitos distintos — sejam elas plataformas de ensino como Udemy, Allura e Khan Academy, ou ferramentas de IA cujo uso difere do almejado nessa plataforma pelos motivos elencados na seção de Justificativa.

As avaliações positivas sobre os elementos da plataforma, tanto auxiliares como principais, é um grande encorajador quanto aos potenciais usos da mesma em ambiente educativo. Apresentando consistentemente mais de 60% de aprovação em seus elementos, cerca de 20% de avaliações neutras e uma reprovação por volta de 20%, a plataforma teve sucesso em se mostrar útil aos usuários, algo que fica muito claro a partir da visão do número de respondentes que afirmam que ela os ajudou no entendimento de programação e que afirmam que recomendariam a plataforma para novos estudantes.

Deve-se, obviamente, se atentar também às ponderações negativas quanto a plataforma. Embora o problema relativo ao *navbar* foi consertado após apontado, uma vez que se trata de uma funcionalidade auxiliar que não estava funcionando como deveria em primeiro momento, demais elementos criticados ao longo da respostas permaneceram inalterados para não se afetar o retorno dos respondentes. Dentre eles o que tem maior destaque é, evidentemente, o *design* da plataforma no relação à escolha da paleta de cores com a qual a plataforma foi montada. Em trabalhos posteriores esse seria certamente um dos principais pontos a serem atacados, embora de forma algum o único; foram elencadas pelos respondentes em suas observações sugestões e críticas quanto a diferentes pontos da plataforma, tal como o uso de linguagem mais informal, críticas quanto a formatação em *smartphones* e sugestões quanto a separação das diferentes seções em páginas diferentes, ao invés de diferentes "partes" de um SPA. Além disso, como visto em relação aos temas considerados mais difíceis, há uma grande quantia de dificuldade constatada em relação ao uso de estruturas como Vetores ou Matrizes, o que indica um possível interesse na implementação de meios auxiliares na compreensão dessas estruturas.

Por fim, embora ainda haja aspectos passíveis de aprimoramento, a plataforma demonstrou um resultado satisfatório, agradando a maior parte dos seus usuários e cumprindo em seu protótipo o objetivo principal ao qual se propôs: o de ser uma potencial ferramenta auxiliar para o ensino de programação, permitindo, assim, que aqueles novos na área ou com dificuldades na aprendizagem compreendam melhor o funcionamento dos diferentes elementos básicos através de exemplos e explicações do seu uso.

Sendo assim, a PortugIF se mostrou como uma ferramenta viável para o ensino introdutório de programação, cumprindo seu objetivo de auxiliar estudantes com barreiras na compreensão de estruturas básicas. Trabalhos futuros incluem melhorias no *design*, responsividade para dispositivos móveis e ampliação dos conteúdos abordados, como vetores e matrizes, fortalecendo ainda mais seu potencial educacional.

Referências

Aidar, L. (2024). Teoria das cores. https://www.significados.com.br/teoria-das-cores/. Online; acessado em 24/07/2024.

Amazon (2022). Amazon codewhisperer. https://aws.amazon.com/pt/codewhisperer/. Online; acessado em 13/09/2022.

Ambrósio, A. P. (2011). Programação de computadores: compreender as dificuldades de aprendizagem dos alunos. *Revista Galego-Portuguesa de Psicoloxía e Educación*, 19(1):185–197.

Anderson, E. F. and McLoughlin, L. (2007). Critters in the classroom: a 3d computer-game-like tool for teaching programming to computer animation students. In *ACM SIGGRAPH 2007 educators program*, pages 1–8.

Ausubel, D. P., Novak, J. D., Hanesian, H., et al. (1978). Educational psychology: A cognitive view.

Baird, C. S. (2015). Why are red, yellow, and blue the primary colors in painting but computer screens use red, green, and blue? https://www.wtamu.edu/~cbaird/sq/2015/01/22/why-are-red-yellow-and-blue-the-primary-colors-in-painting-but-computer-screens-use-red-green-and-blue/. Online; acessado em 24/07/2024.

Bastos, J. E. d. S., Sousa, J. M. d. J., Silva, P. M. N., and de Aquino, R. L. (2023). O uso do questionário como ferramenta metodológica: potencialidades e desafios. *Brazilian Journal of Implantology and Health Sciences*, 5(3):623–636.

Breindel, D. (2024). Technology brand colors: Red, bright and blue. https://www.desantisbreindel.com/insights/b2b-tech-brand-colors/. Online; acessado em 24/07/2024.

Britannica (2024). Rgb color model. https://www.britannica.com/science/RGB-colour-model. Online; acessado em 24/07/2024.

Captain Stack (2022). Captain stack — code suggestion for vscode. https://github.com/hieunc229/copilot-clone. Online; acessado em 13/09/2022.

Cetin, I. et al. (2020). Teaching loops concept through visualization construction. *Informatics in Education-An International Journal*, 19(4):589–609.

Coutinho, E. F., Lima, E. T., and Santos, C. C. (2017). Um panorama sobre o desempenho de uma disciplina inicial de programaçao em um curso de graduaçao. *Revista Tecnologias na Educaç ao*, 19(9):1–15.

Eagle, M. and Barnes, T. (2008). Wu's castle: teaching arrays and loops in a game. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 245–249.

Faeda, L. M., Baffa, M. F., and Pereira, J. S. (2020). Ai (3p) a: Uma metodologia para o ensino de lógica de programação utilizando jogos eletrônicos. *XIX Simpósio Brasileiro de Jogos e Entretenimento Digital-SBGames, Recife*.

Faleiros, F., Käppler, C., Pontes, F. A. R., Silva, S. S. d. C., Goes, F. d. S. N. d., and Cucick, C. D. (2016). Uso de questionário online e divulgação virtual como estratégia de coleta de dados em estudos científicos. *Texto & Contexto-Enfermagem*, 25:e3880014.

Fernandes, V. d. S. and Junior, V. F. (2016). Evasão e reprovação nas disciplinas de lógica e programação: Informações preliminares no campus sombrio, do instituto federal catarinense. In *5º SICT-SUL-Simpósio de Integração Científica e Tecnológica do Sul Catarinense*.

Github (2022). Your ai pair programmer. `https://www.github.com/features/copilot`. Online; acessado em 13/09/2022.

Gomes, A., Areias, C., Henriques, J., and Mendes, A. J. (2008a). Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista portuguesa de pedagogia*, pages 161–179.

Gomes, A., Henriques, J., and Mendes, A. (2008b). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educação, Formação & Tecnologias-ISSN 1646-933X*, 1(1):93–103.

JHipster (2022). Jhipster - full stack platform for the modern developer! `https://www.jhipster.tech`. Online; acessado em 17/10/2022.

Kirby, S., Toland, B., and Deegan, C. (2010). Program visualisation tool for teaching programming in c. In *Proceedings of the International Conference on Education, Training and Informatics (ICETI'10)*, pages 457–461.

Kreutz, D., Yu, J., Esteves-Verissimo, P., Magalhães, C., and Ramos, F. (2017). The kiss principle in software-defined networking: An architecture for keeping it simple and secure. *arXiv preprint arXiv:1702.04294*.

Malik, S. I. and Coldwell-Neilson, J. (2017). A model for teaching an introductory programming course using adri. *Education and Information Technologies*, 22:1089–1120.

Mishra, P. and Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record*, 108(6):1017–1054.

Oliveira, M., Souza, A., Ferreira, A., and Barreiros, E. (2014). Ensino de lógica de programação no ensino fundamental utilizando o scratch: um relato de experiência. In *Anais do XXII Workshop sobre Educação em Computação*, pages 239–248. SBC.

Pattis, R. E. (1994). *Karel the robot: a gentle introduction to the art of programming*. John Wiley & Sons.

Pérez, E. S. and López, F. J. (2019). An ultra-low cost line follower robot as educational tool for teaching programming and circuit's foundations. *Computer Applications in Engineering Education*, 27(2):288–302.

Rahman, S. H. (2022). Keep it simple, stupid — the kiss principle guide to developers. `https://medium.com/sliitwif/keep-it-simple-stupid-the-kiss-principle-guide-to-developers-d6ad83145955`. Online; acessado em 16/05/2024.

Rocha, A. K. d. O. and Prado, M. E. (2018). A programação computacional desenvolvida na perspectiva do tpack no contexto da formação continuada do professor de matemática. *Jornal Internacional de Estudos em Educação Matemática*, 11(3):202–209.

Rodrigues, R. d. S., Andrade, W., Guerrero, D., and Sampaio, L. (2015). Análise dos efeitos do pensamento computacional nas habilidades de estudantes no ensino básico: um estudo sob a perspectiva da programação de computadores. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 26, page 121.

Rowe, G. and Thorburn, G. (2000). Vince—an on-line tutorial tool for teaching introductory programming. *British Journal of Educational Technology*, 31(4):359–369.

Rubek, M. (2002). Apostila de t´ecnicas de programação. Technical report, Pontif´icia Universidade Cat´olica do Paran´a, Curitiba, PR.

Sáez-López, J.-M., Sevillano-García, M.-L., and Vazquez-Cano, E. (2019). The effect of programming on primary school students' mathematical and scientific understanding: educational use of mbot. *Educational Technology Research and Development*, 67:1405–1425.

Segura, R. J., del Pino, F. J., Ogáyar, C. J., and Rueda, A. J. (2020). Vr-ocks: A virtual reality game for learning the basic concepts of programming. *Computer Applications in Engineering Education*, 28(1):31–41.

Smith, B. (2012). How to kiss: The art of keeping it simple, stupid. *X Double Dot, LLC*, pages 1–6.

Tabnine (2022). Ai assistant for software developers. `https://www.tabnine.com`. Online; acessado em 13/09/2022.

Vieira, C. E. C., Junior, J. A. T. d. L., de Vieira, P. P., et al. (2015). Dificuldades no processo de aprendizagem de algoritmos: uma análise dos resultados na disciplina de al1 do curso de sistemas de informação da faeterj–campus paracambi. *Cadernos UniFOA*, 10(27):5–15.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*, volume 86. Harvard university press.

Yunanto, A. A., Putri, F. F., Permatasari, D. I., Hardiansyah, F. F., Sa'adah, U., Aziz, A. S., et al. (2024). Design and implementation the prayer reminder application using kiss principle based on user centered design. *Procedia Computer Science*, 234:1484–1491.