JOAO - Proposta de um Juiz Online como Ferramenta de Apoio ao Ensino e Aprendizagem de Programação para Crianças e Adolescentes

Socorro Vânia L. Alves[1], Sávio M. Gaia[1], Yure S. G. Duarte[1], Enoque Melo Alves[1]

[1]Instituto de Engenharia e Geociências – Programa de Computação – Universidade Federal do Oeste do Pará (UFOPA)
Caixa Postal 68040 – 470 – Santarém – PA – Brazil

{socorro.alves,enoque.alves}@ufopa.edu.br,{savio.gaia, yure.duarte}@discente.ufopa.edu.br

Abstract. *This article describes the proposal, development and validation of an online judge system, called JOAO, aimed at learning block-based programming, particularly for children and teenagers. Through a practical experiment, carried out by a group of users, it was verified that the approach adopted in the system left the learners very motivated, increased their level of engagement with the activities and facilitated the understanding of abstract programming concepts.*

Resumo. *Este artigo descreve a proposta, desenvolvimento e a validação de um sistema de juiz online, denominado JOAO, voltado para a aprendizagem de programação baseada em blocos, particularmente para crianças e adolescentes. Por meio de um experimento prático, realizados por um grupo de usuários, verificou-se que a abordagem adotada no sistema deixou os aprendizes muito motivados, aumentou o nível de engajamento deles com as atividades e facilitou a compreensão dos conceitos abstratos da programação.*

1. Introdução

Com o avanço das tecnologias digitais da informação e comunicação (TDIC), muitos sistemas computacionais têm sido utilizados nos mais distintos contextos educacionais a fim de potencializar os processos de ensino e aprendizagem, permitindo que os aprendizes se sintam mais motivados a pesquisar, a construir novos conhecimentos e a desenvolver a autonomia. Dentre tais sistemas, os juízes online de programação têm recebido bastante atenção na literatura nos últimos anos [Dwan et al. 2017, Watanobe et al. 2020, Pereira et al. 2021].

Juízes online são sistemas que disponibilizam exercícios de programação e permitem a construção de códigos-fonte, que são submetidos e corrigidos automaticamente utilizando casos de teste pré-cadastrados. Normalmente são utilizados como ferramentas de apoio pedagógico, pois reduzem a carga de trabalho do professor e oferecem *feedback* rápido aos aprendizes [Dwan et al. 2017].

Atualmente existem inúmeros exemplos de sistemas juízes online disponíveis para uso e com uma boa aceitação de professores e alunos. No entanto, a maioria requer que o aprendiz escreva as linhas de códigos em uma linguagem de programação usual do mercado (como Phyton, Java, entre outras), obedecendo toda a sintaxe desta. Esta

DOI: 10.5753/wie.2022.225698

demanda, comumente, exige muito mais tempo do programador e um estudo mais aprofundado dos elementos da linguagem até que se chegue a um aprendizado realmente significativo. Essa situação agrava-se ainda mais quando os aprendizes são crianças e adolescentes, pois esse público tem mais dificuldade em aprender os conceitos de programação, devido a carga forte de conceitos abstratos que envolvem todo o ambiente de aprendizado, como a própria linguagem de programação, a plataforma de desenvolvimento do código-fonte, até o próprio entendimento do funcionamento do computador.

Na perspectiva de encontrar soluções para dar maior expressividade e apoio ao ensino e aprendizagem de programação para um público mais jovem, uma abordagem pedagógica que vem crescentemente se revelando como promissora é o método da programação em blocos. Nessa abordagem, os blocos são a unidade fundamental do código-fonte e podem representar comandos, condições, repetição, objetos e muitos outros elementos que fazem parte da construção de um programa de computador. Muitos estudos revelam que é uma abordagem simples, porém, eficiente de introduzir conhecimentos que à primeira vista dos aprendizes parecem ser muito abstratos e complicados [Alves et al. 2017, Oliveira et al. 2021, Perin et al. 2021].

As interfaces dos juízes online existentes até o presente momento não permitem que o aprendiz crie seu código-fonte, para a solução do exercício disponibilizado pela ferramenta, utilizando o método da programação em blocos. São majoritariamente baseados na construção do código-fonte com base em linhas de comandos de uma linguagem de programação. Na perspectiva que a programação em blocos poderia auxiliar o aprendizado de programação de computadores, principalmente para um público mais jovem, este trabalho tem então como objetivo apresentar um novo sistema de juiz online, denominado JOAO (acrônimo para Juiz Online de Avaliação com Orientação), que utiliza como unidade básica para a criação do código-fonte a estrutura de blocos de comandos. O sistema além de permitir que o estudante receba *feedback* imediato sobre a corretude do código criado, torna a aprendizagem de programação mais visual, auxiliando na transmissão de um conhecimento puramente abstrato para algo mais sólido.

Este artigo está organizado da seguinte forma: na Seção 2 são apresentados os fundamentos teóricos, a Seção 3 apresenta o sistema de juiz online JOAO, na Seção 4 é descrita a metodologia empregada na avaliação do sistema JOAO, na Seção 5 são apresentados os resultados, seguido das conclusões e trabalhos futuros, na Seção 6.

2. Fundamentação teórica

2.1. Aprendizagem e Ensino de Programação para crianças e adolescentes

Nos dias atuais, a importância e pervasividade dos computadores e aparelhos eletrônicos na sociedade têm aumentado muito. Crianças e adolescentes já estão acostumados a navegar na internet, usar *notebooks*, *tablets* e *smartphones* sem muitos problemas, algo que na geração passada não era comum. O conhecimento da programação destes dispositivos para a execução das mais diversas tarefas é fundamental e tem se tornado uma habilidade cada vez mais pertinente desde a infância. No entanto, programar é uma atividade complexa e que envolve diferentes conceitos e bastante prática.

No Brasil, o ensino de programação para crianças e adolescentes é um tema que vem sendo amplamente discutido nos últimos anos. Várias pesquisas e projetos apontam

seus benefícios tanto para o público quanto para a área tecnológica, como uma evolução no rendimento escolar, mais autonomia na hora de resolver problemas, tendência ao trabalho colaborativo e um aumento no interesse por cursos técnicos e superiores na área de informática [Mota et al. 2014, Rodrigues et al. 2017].

Entretanto, apesar dos muitos benefícios proporcionados à educação, ainda existem muitas barreiras que dificultam o ensino e a aprendizagem da programação de computadores. Almeida et al. (2002) observaram nos cursos que têm como objetivo o ensino da programação de computadores, um grande desinteresse dos alunos, o que é comumente associado a resistências e dificuldades no aprendizado. Os problemas surgem principalmente devido a uma carga forte de conceitos abstratos que envolvem todo o ambiente de aprendizado, e que torna o processo frustrante.

Além do motivo supracitado, outros estudos apontam diferentes dificuldades: incompreensão do enunciado dos problemas, dificuldades com a sintaxe da linguagem utilizada, baixa abstração para focar apenas no que é importante e pensamento lógico não estabelecido, entre outras. A obrigatoriedade do uso de uma linguagem de programação, que segue o modelo de aplicação em console, baseada em comandos que precisam estar sintaticamente bem construídos, dificulta o aprendizado, principalmente das crianças e adolescentes.

Além dos problemas relacionados à aprendizagem, existe também a dificuldade no ensino da programação, que exige do professor, além do domínio teórico, a utilização de métodos e técnicas variadas de ensino. Alves et al. (2017) discutem que a dificuldade do professor acompanhar efetivamente as atividades laboratoriais de programação, em decorrência do elevado número de alunos por turma, ou por pouco tempo para corrigir os exercícios enviados pelos alunos, são fatores que agravam o processo de ensino da programação. A falta de *feedback* rápido do professor agrava ainda mais todo processo.

Na tentative de apoiar o processo de ensino e aprendizagem de programação, principalmente para um público mais jovem, muitas abordagens e ferramentas têm sido propostas nos últimos anos, como a programação baseada em blocos, apresentada na próxima seção.

2.2. A Programação em Blocos

Ao contrário da programação tradicional, que consiste em construir um código com a digitação de comandos de uma linguagem, na programação em blocos o programador simplesmente precisa arrastar os blocos de construção (*building blocks*) de comandos para o ambiente de desenvolvimento, agrupando-os como se estivesse montando um quebra-cabeça, sem a necessidade de se preocupar com a escrita de comandos complexos, regras de sintaxe e alinhamento, o que torna o processo de codificar mais amigável e atrativo. Assim, quando uma sequência lógica de peças é montada, tem-se um programa para a realização de uma tarefa e/ou resolução de um problema.

Segundo Batista et al. (2017), a codificação em blocos é altamente eficaz em introduzir jovens em disciplinas introdutórias de programação em cursos da área de tecnologia da informação. A programação em blocos é uma forma mais fácil de ensinar os conceitos básicos de programação, como entradas/saídas, estruturas sequenciais, condicionais e de decisão. A linguagem impede os usuários de cometerem erros de sintaxe, devido os blocos serem programados para encaixarem-se somente onde houver sentido sintaticamente.

Os esforços dos grupos de pesquisadores do MIT (*Massachusetts Institute of Technology*), em desenvolvimento de linguagens e ambientes de programação e aprendizagem para crianças e jovens deram origem ao Scratch (https://scratch.mit.edu), uma linguagem de programação inspirada na linguagem LOGO [Papert 1985]. A partir do Scratch, surgiram outras linguagens de programação visual e interfaces que facilitam e dão suporte ao desenvolvimento de programas seguindo o paradigma de blocos programáveis, como o ArduBlock, o App Inventor, Swift Playgrounds e a Blockly (https://developers.google.com/blockly), uma biblioteca, desenvolvida pelo Google, que permite criar editores e linguagens de programação visual baseadas em blocos.

2.3. Juiz Online

O aprendizado de programação baseia-se na prática e repetição constante de exercícios [Galvão et al. 2016]. Nesse processo é importante que o estudante receba *feedback* imediato, a fim de localizar seus erros, compreender a origem destes e remediá-los. Assim, evita-se que os estudantes fiquem desmotivados na tentativa de aprender os princípios da programação [Pelz et al., 2012]. Neste sentido, uma solução que vem sendo muito utilizada nos últimos anos para dar apoio ao ensino e aprendizagem de programação, tem sido sistemas de juízes online [Galvão et al. 2016, Bez et al. 2014, Pereira et al. 2021].

Um juiz online pode ser definido como um sistema que recebe código-fonte desenvolvido pelo usuário como solução de um determinado exercício de programação, compila, executa e avalia a corretude deste código de forma automatizada e em tempo real [Yera e Martínez 2017]. Esta análise de corretude normalmente é realizada mediante a execução do código e a comparação da saída retornada com uma saída esperada, ou seja, utilizando casos de teste pré cadastrados.

Entre as vantagens do uso de juízes online, pode-se citar: redução do tempo de *feedback* e correção dos códigos por parte dos professores, possibilidade do professor acompanhar as atividades desenvolvidas pelo estudante, avaliações mais consistentes, e ampliação dos tipos e do número de problemas resolvidos pelos aprendizes [Yera e Martínez 2017, Galvão et al. 2016, Zhao et al. 2018].

Atualmente existem inúmeros exemplos de sistemas juízes online disponíveis para uso e com ampla aceitação por parte de professores e aprendizes. Dentre os mais conhecidos destacam-se: o sistema URI Online Judge (www.urionlinejudge.com.br), os sistemas Code Chef (www.codechef.com) e o BOCA (*BOCA Online Contest Administrator*), usados em competições de programação; o HackerRank (www.hackerrank.com), uma ferramenta gamificada que tem desafios de programação; o sistema CodeForce (https://codeforces.com/), uma plataforma que apoia o desenvolvimento do pensamento computacional e habilidades de codificação para iniciantes.; e o CodeBench (http://codebench.icomp.ufam.edu.br/), desenvolvido na UFAM (Universidade Federal do Amazonas).

3. Apresentação do Juiz Online JOAO

O Juiz Online de Avaliação com Orientação (JOAO) tem como público alvo crianças e adolescentes, mas especificamente alunos do Ensino Fundamental II (6º ao 9º ano), por isso sua interface foi projetada utilizando-se critérios de usabilidade, como clareza, consistência, familiaridade, objetividade e atratividade.

No desenvolvimento da interface do sistema foi utilizada a biblioteca *ReactJS*, devido sua característica de criar páginas dinâmicas e flexíveis. Na parte lógica, a plataforma foi desenvolvida utilizando *NodeJS,* juntamente com o framework *loopback 4*, por ser uma tecnologia considerada altamente extensível, focada em produtividade e possuir compatibilidade com a maioria dos bancos de dados. O banco de dados selecionado para o armazenamento dos exercícios de programação foi o MongoDB, devido as suas características de escalabilidade e facilidade de implementação.

O projeto da ferramenta foi desenvolvido com uma arquitetura desacoplada, isto é, a parte da interface visual *(front-end)* é totalmente independente da lógica *(back-end)*, o que impôs o uso de uma arquitetura do tipo REST (Figura 1). Nessa arquitetura, o *front-end* se comunica com a API no formato JSON, que detém todas as informações sobre os exercícios, usuários e os códigos de resolução dos problemas. Já o *back-end* é composto de dois componentes independentes: a API que é responsável por disponibilizar os exercícios e o juiz que é responsável por determinar o resultado dos códigos submetidos.

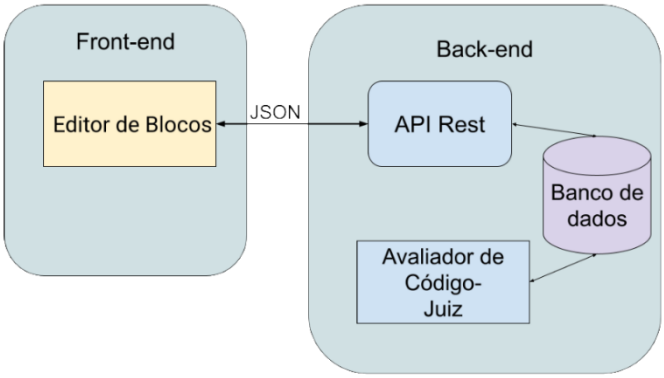

Figura 1. Modelo conceitual da arquitetura do sistema.

O JOAO utiliza a abordagem de autenticação estática para o usuário ter acesso aos recursos do sistema. Embora qualquer pessoa possa criar uma conta através de email e senha, nem todas tem permissão para criar ou editar recursos do sistema, isto ocorre porque o JOAO adota um mecanismo de acesso denominado de controle de acesso baseado em funções (*Role-based access control* ou RBAC) que envolve definir permissões e privilégios para permitir o acesso a usuários autorizados. Com base neste mecanismo, o JOAO disponibiliza dois tipos de usuário, um administrador e o usuário comum, que tem a responsabilidade de manter seus dados pessoais atualizados e submeter soluções aos problemas propostos pela ferramenta. O administrador, além de possuir as mesmas permissões que um usuário comum, é responsável por manter o banco de exercícios, seja editando, adicionando ou excluindo itens.

Na Figura 2 é apresentado o ambiente de desenvolvimento (do inglês, *Integrated Development Environment* - IDE) do JOAO, nele o código-fonte da solução do exercício disponibilizado é desenvolvido com blocos gráficos de comandos e avaliado após a submissão. Quando um novo exercício é registrado no sistema, este passa a integrar o banco de questões e fica imediatamente disponível. Os exercícios são categorizados em níveis de dificuldades (fácil, intermediário e difícil) que podem ser selecionados de acordo com a preferência e perfil do usuário.

Figura 2. Tela com o ambiente de desenvolvimento.

Após a seleção do exercício, é apresentada a tela do editor de código em blocos. Conforme ilustrado na Figura 2, os blocos de comandos foram divididos em categorias, que usam formas e cores específicas para ajudar a diferenciar sua função, por exemplo, na categoria “Texto” temos blocos de métodos e funções para manipular textos, já a categoria “Entrada e Saída” agrega blocos de comandos de entrada e saída de dados. O exercício a ser resolvido é exibido na parte superior da interface, então o aprendiz monta o código da solução do problema a partir dos blocos de comandos disponíveis, que devem ser arrastados para a área do ambiente de desenvolvimento. Na Figura 2 ainda é possível observar o código montado no ambiente de desenvolvimento.

A estrutura do editor de blocos de comandos foi criada com a biblioteca Blockly. Segundo a documentação do Google, "Blockly é uma biblioteca que adiciona um editor de código visual a aplicativos da Web e móveis”. A biblioteca permite criar um ambiente personalizado, permitindo a adição de novos blocos de comando e a personalização dos já existentes. Blockly possui ainda um recurso que gera código nas linguagens Python, JavaScript, Dart e Php, a partir dos blocos montados em um ambiente de desenvolvimento, ou de uma estrutura em Json ou XML. Também é possível fazer o contrário, gerar a estrutura a partir do código em uma das linguagens citadas. Na Figura 3, mais à direita, é possível visualizar no JOAO uma aba acima do ambiente de desenvolvimento com código em JavaScript, com a possibilidade do usuário ainda alterar para outras linguagens que são geradas pela biblioteca.

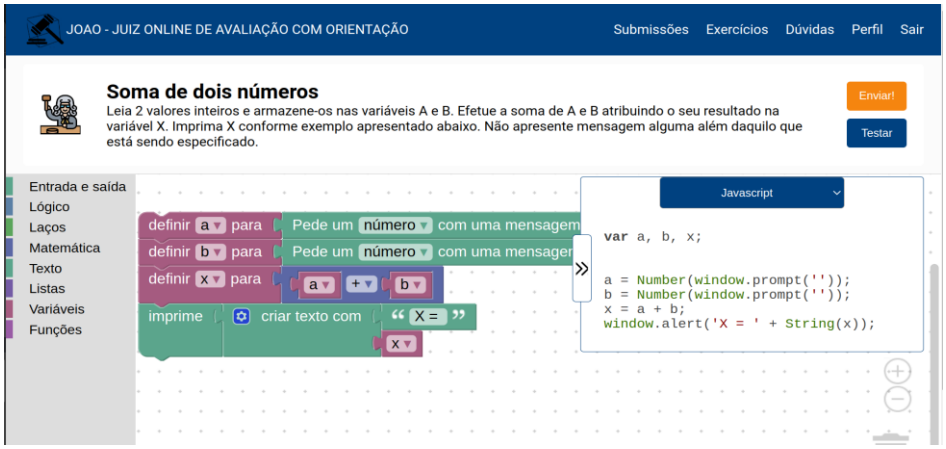

Figura 3. Tela com o editor de código em blocos

Depois do código ser submetido à avaliação do juiz online, o aprendiz é imediatamente informado se este é uma solução válida para o problema que tentou solucionar. Conforme é visualizado na Figura 4, quando o código não é aceito o sistema informa, em porcentagem, o quanto ele está correto, baseado na quantidade de casos de teste em que foi aprovado, além disso é dada orientação ao estudante sobre o erro e como corrigi-lo em tal código, emitindo dicas personalizadas. Os resultados são classificados seguindo o critério de conformidade de igualdade de saídas, isto é, se a saída do código-fonte avaliado for a mesma definida nos casos testes do problema, então a solução é aceita, senão é rejeitada.

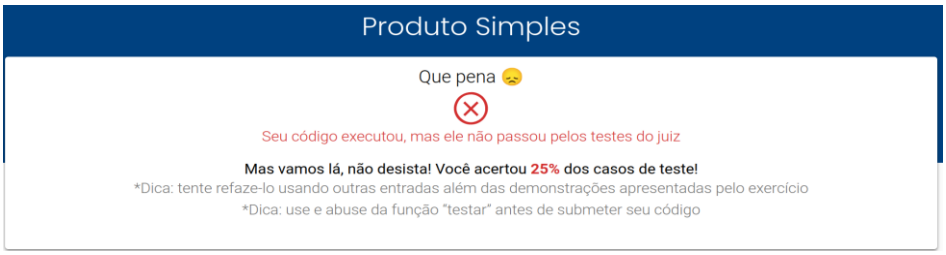

Figura 4. Tela de resposta do Juiz com resposta para uma exceção

O aprendiz pode revisar seu código e submetê-lo novamente tantas vezes quantas desejar. No Quadro 1 são apresentadas as possíveis respostas fornecidas após a avaliação de um código ser submetido ao sistema.

Quadro 1. Respostas possíveis do juiz JOAO

Resposta	Descrição
Accepted	O código gerou as saídas esperadas para todos os casos de teste do problema
Presentation Error	a apresentação da saída do código julgado se difere em um ou mais casos de teste da apresentação esperada;
Runtime Error	houveram erros durante a execução do código em avaliação
Time limit exceeded	o código em avaliação levou um tempo maior que o configurado para o problema para executar

Após avaliar o código e este passar pelos testes, o aprendiz é direcionado para o próximo exercício daquele nível de dificuldade e um novo ciclo de problema-solução com blocos de comandos inicia-se. Quando todos os exercícios de um dado nível são solucionados com sucesso, o juiz automaticamente direciona o aprendiz para o nível seguinte de dificuldade. Outra funcionalidade presente na ferramenta é o envio de email que favorece a interação entre aprendiz e professor. O aprendiz, por exemplo, pode enviar um email para o professor para esclarecer dúvidas sobre o enunciado ou fazer argumentações sobre a solução adotada.

4. Metodologia de avaliação do sistema JOAO com usuários reais

Com o objetivo de verificar os resultados gerados pelo uso do sistema JOAO, quanto ao ensino e aprendizagem dos conteúdos de programação e também quanto ao uso geral da própria ferramenta, referente ao seu funcionamento e usabilidade, foi realizada uma experimentação prática da ferramenta com um grupo de 34 adolescentes, na faixa etária de 11 a 16 anos, alunos de um projeto de extensão intitulado “Pensamento Computacional

e Programação para crianças e adolescentes do Residencial Salvação: Criando Perspectivas para o futuro", desenvolvido na Universidade Federal do Oeste do Pará (UFOPA). O projeto ainda está em andamento no ano de 2022, tem a duração de 1 ano, e visa desenvolver habilidades e competências relacionadas ao Pensamento Computacional em adolescentes em situação de vulnerabilidade social do Bairro Residencial Salvação, despertando o raciocínio lógico-matemático, a criatividade, a capacidade de resolução de problemas e a organização do pensamento, através da Computação Desplugada e da Programação de Computadores.

As aulas do projeto foram divididas em 4 módulos: programação desplugada, computação plugada com a programação em blocos, modelagem 3D e robótica. O Sistema JOAO foi introduzido no segundo módulo (o plugado) e foi apresentado aos alunos como uma ferramenta que os apoiaria no desenvolvimento e avaliação automática dos códigos de programação com blocos. A experimentação ocorreu entre os dias 11/04/2022 e 24/06/2022, com todos os encontros no formato presencial, realizados nos laboratórios do curso de Ciência da Computação da UFOPA.

Os 34 alunos do projeto foram convidados a participarem do experimento e ao final deste exporem suas percepções quanto ao uso do Sistema JOAO. A coleta de dados foi realizada através da aplicação de um questionário online, que consistia de perguntas abertas para capturar livremente o entendimento e a dificuldade de realizar os exercícios propostos e a opinião acerca da ferramenta. A partir dos dados coletados, realizou-se uma análise qualitativa sobre os comentários dos participantes por meio do Método de Explicitação do Discurso Subjacente (MEDS) [Nicolaci-da-Costa 2007]. Os resultados são apresentados na Seção 5 a seguir.

5. Análise dos Resultados Obtidos

Nesta seção, são apresentados os resultados qualitativos (percepção dos alunos) a respeito da ferramenta de juiz online JOAO adotada no ensino de programação. Os alunos participantes do experimento serão referenciados como aluno A1, aluno A2, e assim sucessivamente.

A partir da análise qualitativa das respostas dos alunos sobre o questionário aplicado ao fim do módulo plugado do projeto de extensão, foram identificadas as categorias apresentadas na Tabela 1. As respostas foram sistematicamente comparadas em busca de recorrências similares e contabilizadas na coluna ocorrências da tabela.

Table 1. Categorias resultantes da análise qualitativa dos dados

Num	Categoria	Ocorrências
1	Feedback rápido	23
2	Facilidade na criação do código fonte	19
3	Qualidade dos enunciados dos exercícios propostos	16
4	Quantidade de exercícios propostos	9
5	Forma de correção do código	17
6	Facilidade de interação com os tutores	13
7	Estudo autônomo	7

A seguir, são transcritas e comentadas algumas respostas consideradas mais relevantes em determinadas categorias. Tais respostas são destacadas em itálico e reproduzidas da mesma forma como foram escritas pelos alunos.

O *feedback* rápido (categoria 1) proporcionado pela ferramenta teve bastante relevância na motivação dos alunos. A maioria deles explicitaram que ter uma resposta rápida sobre o código desenvolvido ajuda muito a perceber o seu real entendimento sobre o conteúdo trabalhado pelo professor em sala, além da identificação dos seus erros recorrentes, conforme ilustra a citação do aluno A13: "*Foi muito bom praticar na ferramenta, não ficamos só com o exemplo dado pelo professor, dá de resolver outros problemas parecidos, e ter uma resposta rápida sobre o código que montei, se errava a ferramenta mostrava o que errei, quanto acertei e me dava dicas de como corrigir os erros. Não precisava o professor ver e corrigir*".

Com relação a facilidade de montar o código fonte para a solução do exercício proposto (categoria 2), os alunos relataram vários motivos que os deixaram motivados com essa abordagem de aprender programação: não tiveram dificuldades de entender por ser visual, os blocos estavam bem posicionados nas categorias segundo suas funções, e a facilidade de um bloco se encaixar somente quando há sentido ajudou na construção do código e auxiliou até a repensar a lógica. O aluno A26 afirmou: "*É bem simples de usar, é só clicar e arrastar. Mesmo quando fiquei na dúvida onde estava o bloco do commando certo, rapidinho achava, o uso das cores e da forma também ajuda muito, na hora de montar o código de solução".* Outras citações também ilustram a facilidade no desenvolvimento do código com blocos, aluno A22: "*A ferramenta é bem simples, posso escolher o nível de dificuldade do exercício, e depois montar a pilha de blocos, submeter o meu código para correção e ter uma resposta rápida. Tudo muito fácil. Fiquei animado, acho que estou aprendendo*".

Com respeito a qualidade (categoria 3) e quantidade dos exercícios (categoria 4), os alunos destacaram que os enunciados dos exercícios estão bem claros, porém expressaram que o número de questões em cada nível de dificuldade poderia ser ampliado para diverficar os problemas abordados no nível de dificuldade, como observado na citação do aluno A17: "Achei a maioria dos exercícios fáceis de entender, só que tinham poucos para treinar, cheguei a resolver todos de um nível numa aula, queria mais".

Com relação a forma de correção do Sistema JOAO sobre o código fonte submetido (categoria 5), os alunos destacaram muitos pontos positivos. Na percepção deles mesmo quando há erro, a característica do sistema informar em porcentagem quanto está correto ajuda a mensurar o grau de entendimento já obtido e as fragilidades no aprendizado de programação que ainda precisam ser vencidas. As dicas personalizadas de como corrigir os erros também auxiliam muito nas próximas tentativas de contrução do código da solução.

Por fim, a funcionalidade presente no sistema do aluno poder enviar emails para o professor, expressando dúvidas, fazendo sugestões ou argumentações foi bastante elogiada pelos participantes. Segundo eles, a presença desse canal de comunicação facillitou a interação com os tutores (categoria 6), principalmente quando os exercícios eram resolvidos a distância, visto que os encontros presenciais para a apresentação de conteúdos ocorria somente uma vez por semana. Ao mesmo tempo, a possibilidade de estudo autônomo (categoria 7) potencializou a aprendizagem, pois o aluno se tornava protagonista do seu próprio conhecimento. As citações a seguir Aluno A27 e Aluno A11, exemplificam respectivamente cada categoria citada: "*Essa opção de enviar email ao professor é útil, duas vezes quando eu estava em casa precisei da ajuda dele, aí escrevi um email e logo me respondeu, não tinha entendido na aula o uso dos Laços*"; e "*Eu não*

tenho computador em casa e esse sistema possibilita que eu estude sozinho, na hora que sentir vontade".

6. Conclusões e Trabalhos Futuros

Juízes online, que provêm feedback automático sobre a corretude de programas submetidos, são ferramentas fundamentais para motivar o aprendizado e a autonomia dos alunos e apoiar o professor no processo de ensino dos conteúdos. O processo de codificação baseado em blocos quando integrado a essas ferramentas, permite que aprendizes programem sem precisar decorar comandos de uma linguagem de programação, o que o torna um modelo ideal para crianças ou pessoas que estão aprendendo a habilidade de programação.

Neste trabalho, apresentou-se o JOAO, um juiz online desenvolvido como ferramenta de apoio ao ensino de programação, utilizando blocos visuais de comandos. Verificou-se que essa abordagem deixou os aprendizes muito motivados, aumentou o nível de engajamento deles com as atividades e facilitou a compreensão dos conceitos abstratos da programação. O desempenho dos alunos na resolução dos exercícios propostos pela ferramenta e com os problemas propostos nos encontros presenciais pelos professores foi muito satisfatório. A percepção dos alunos participantes do experimento em relação à ferramenta foi, em geral, muito positiva e está motivando a continuidade e evolução do sistema.

Como trabalhos futuros, pretende-se tornar a ferramenta colaborativa, permitindo que mais de um usuário interaja na resolução de um problema, permitindo a discussão de ideias e o compartilhamento de conhecimentos. Para tanto, será necessário desenvolver um mecanismo mais efetivo de controle de usuários. Além disso, pretende-se criar um *dashboard* que possa apresentar de modo contínuo e atualizado informações sobre estados dos alunos e os progressos em programação.

Referencias

Almeida, E., Costa, E., Silva, K., Paes, R., Almeida, A. and Braga, J. (2002). AMBAP: Um Ambiente de Apoio ao Aprendizado de Programação. Workshop de Educação em Computação, Congresso anual da SBC, Florianópolis.

Alves, E.; Alves, S. V. L. and Cabral, L.; Junior, R. (2017). Módulo de Programação baseada em blocos para a plataforma Jabuti Edu com Blockly. XXIII Congresso Internacional de Informática na Educação (TISE). Anais Nuevas Ideas em Informática Educativa, Fortaleza-Ceará, v. 13, p.641-647. ISBN: 978-956-19-1043-0.

Batista, E. J. S., Silva, L., Leite, C. and Lima, A. (2017). Poredu: um ambiente de programação em blocos. Anais dos Workshops do Congresso Brasileiro de Informática na Educação; Anais dos Workshops do CBIE.

Bez, J. L., Tonin, N. A. and Rodegheri, P. R. (2014). Uri online judge academic: A tool for algorithms and programming classes. In 2014 9th International Conference on Computer Science & Education, pages 149–152. IEEE.

Dwan, F., Oliveira, E. and Fernandes, D. (2017). Predicão de zona de aprendizagem de alunos de introdução à programação em ambientes de correção automática de código. In Simpósio Brasileiro de Informática na Educação (SBIE), volume 28, page 1507.

Galvão, L., Fernandes, D. and Gadelha, B. (2016). Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. In Brazilian Symposium on Computers in Education (SBIE), volume 27, page 140.

Mota, F. P.; Ribeiro, N. F. A.; Emmendorfer, L.; Butze, P.; Machado, K. S. and Adamatti, D. F. (2014). Desenvolvendo o Raciocínio Lógico no Ensino Médio: uma proposta utilizando a ferramenta Scratch. In: Anais do 25º. Simpósio Brasileiro de Informática na Educação. 3º. Congresso Brasileiro de Informática na Educação. Nov. 2014. Dourados, MS.

Nicolaci-da-Costa, A. M. (2007) O Campo da Pesquisa Qualitativa e o Método da Explicitação do Discurso Subjacente (MEDS). In: Psicologia: Reflexão e Crítica. vol.20 no.1. ISSN: 0102-7972. RS, Porto Alegre.

Oliveira, S., Pereira, M. and Teixeira, F. (2021). MIT App Inventor como Ambiente de Ensino de Algoritmos e Programação. In Anais do XXIX Workshop sobre Educação em Computação, (pp. 61-70). Porto Alegre: SBC. doi:10.5753/wei.2021.15897.

Papert, S. (1985). Logo: computadores e educação. Editora Brasiliense, São Paulo.

Pelz, F. D.; Jesus, E. A.; Raabe, A. L. (2012). Um Mecanismo para Correção Automática de Exercícios Práticos de Programação Introdutória. In Anais do XXIII Simpósio Brasileiro de Informática na Educação (Vol. 23, No. 1).

Pereira, F. D., Fonseca, S. C., Oliveira, E. H., Cristea, A. I., Bellhauser, H., Rodrigues, L., Oliveira, D. B., Isotani, S. and Carvalho, L. S. (2021). Explaining individual and collective programming students' behaviour by interpreting a black-box predictive model. IEEE Access.

Perin, A., Silva, D. and Valentim, N. (2021). Um benchmark de ferramentas de programação em blocos que podem ser utilizadas nas salas de aula do Ensino Médio. In Anais do XXXII Simpósio Brasileiro de Informática na Educação, (pp. 1162-1173). Porto Alegre: SBC. doi:10.5753/sbie.2021.217765.

Rodrigues, L. C. Queiroga, A. P. G. de; Oliveira, M. V. and More, A. T. (2017). Projeto de Extensão: curso de introdução à programação para crianças do ensino fundamental. VI Congresso Brasileiro de Informática na Educação. Anais do XXIII Workshop de Informática na Escola.

Watanobe, Y., Chowdhury, I., Cortez, R. and Vazhenin, A. (2020). Next-generation programming learning platform: Architecture and challenges. SHS Web of Conferences, 77:01004.

Yera, R. and Martínez, L. (2017). A recommendation approach for programming online judges supported by data preprocessing techniques. Applied Intelligence, 47(2):277–290.

Zhao, W. X., Zhang, W., He, Y., Xie, X. and Wen, J.-R. (2018). Automatically learning topics and difficulty levels of problems in online judge systems. ACM Transactions on Information Systems (TOIS), 36(3):1–33.