UTILIZAÇÃO DE LINGUAGENS DE PROGRAMAÇÃO VISUAL NO ENSINO INTERDISCIPLINAR

Jonathan Donato Pippi[1]; Rogério Correa Turchetti[2]; Leila Maria Araújo Santos[3]; Ricardo Machado Ellensohn[4]

Resumo
O ensino e aprendizagem de algoritmos e lógica de programação, dentre vários benefícios, estimulam o raciocínio lógico e a criatividade dos estudantes, além de desenvolver habilidades relacionadas à resolução de problemas. Geralmente, os conteúdos relacionados ao ensino e aprendizagem de algoritmos e lógica de programação, não beneficiam a resolução de problemas. Para aumentar o interesse dos estudantes no ensino e aprendizagem de algoritmos e lógica de programação, uma alternativa possível é a adoção de ferramentas que utilizam elementos gráficos e figuras para desenvolver programas. Nesse sentido, as linguagens de programação visual são consideradas adequadas para serem utilizadas no início do processo de ensino e aprendizagem de algoritmos e lógica de programação, pois possibilitam aos estudantes o desenvolvimento do raciocínio lógico a partir do foco na resolução de problemas. Considerando este contexto, o presente trabalho busca avaliar as funcionalidades da linguagem de programação visual: Alice, App Inventor e Scratch para, além da aprendizagem da lógica de programação, também, explorar questões multidisciplinares viabilizando, na prática educativa, uma abordagem teoricamente sistematizada.
Palavras-chave: Algoritmos; Lógica de programação; Linguagem de programação visual.

Abstract
Teaching and learning algorithms and programming logic offers various benefits, including stimulating students' logical reasoning and creativity, as well as developing problem-solving skills. However, the content typically associated with teaching these subjects often fails to effectively foster problem-solving abilities. To boost student interest, one viable alternative is the adoption of tools that utilize graphical elements and visual figures for program development. In this regard, visual programming languages are considered suitable for the initial stages of instruction, as they enable students to develop logical reasoning by focusing on problem-solving. Against this backdrop, this study evaluates the functionalities of the visual programming languages Alice, App Inventor, and Scratch. The aim is not only to facilitate the learning of programming logic but also to explore multidisciplinary issues, thereby enabling a theoretically systematized approach within educational practice.
Keywords: Algorithms; Programming logic; Visual programming language.

[1] Mestre em Tecnologias Educacionais em Rede pela Universidade Federal de Santa Maria-UFSM; professor da Escola Estadual de Ensino Médio Professora Maria Rocha em Santa Maria-RS. E-mail: jonathan.pippi@gmail.com.
[2] Doutor em Informática pela Universidade Federal do Paraná-UFPR; professor associado da Universidade Federal de Santa Maria-UFSM, atuando nos cursos de graduação em computação (presencial e EaD) e no Mestrado Acadêmico em Educação Profissional e Tecnológica do CTISM. E-mail: turchetti@redes.ufsm.br.
[3] Doutora em Informática na Educação pela Universidade Federal do Rio Grande do Sul-UFRGS; professora titular da Universidade Federal de Santa Maria-UFSM. E-mail: leilamas@ctism.ufsm.br.
[4] Doutor em Química Orgânica pela Universidade de São Paulo – USP; professor associado na Universidade Federal do Pampa - UNIPAMPA, campus Caçapava do Sul/RS. E-mail: ricardoellensohn@unipampa.edu.br.

1 INTRODUÇÃO

O ensino e aprendizagem de algoritmos e lógica de programação, não apenas nos cursos que envolvem Computação, dentre vários benefícios, estimulam o raciocínio lógico e a criatividade dos estudantes, além de desenvolver habilidades relacionadas à resolução de problemas (Krohl; Dutra; De Matos, 2021).

Segundo Zorzo *et al.* (2017), o ensino e aprendizagem de lógica e programação pode ser considerado fundamental para que o estudante desenvolva a criatividade e o raciocínio lógico.

O ensino e aprendizagem de algoritmos e lógica de programação, mesmo indicando inúmeras vantagens, apresentam desafios que precisam ser avaliados, como a falta de interesse e motivação de alguns estudantes, por considerarem o conteúdo de difícil compreensão (Moreira *et al*., 2018).

Corroborando com isso, para Morais, Mendes Neto e Osório (2020), os conteúdos relacionados ao ensino e aprendizagem de algoritmos e lógica de programação, usualmente, não beneficiam a resolução de problemas, mas a sintaxe da linguagem computacional e o significado de determinados comandos.

Por estes motivos, bem como, para tentar minimizar esta situação, e aumentar o interesse dos estudantes no ensino e aprendizagem de algoritmos e lógica de programação, uma alternativa possível, para os estudantes superarem os desafios encontrados, é a adoção de ferramentas que utilizam elementos gráficos e figuras para desenvolver programas.

A CSTA (*Computer Science Teachers Association*), responsável pelo currículo internacional CSTA K-12 Computer Science Standards, que planeja os objetivos de aprendizagem para o ensino de computação, entende que o ensino e aprendizagem de algoritmos e lógica de programação, por meio das linguagens de programação visual, deveriam ser introduzidas desde a educação básica, pois despertam a autonomia e curiosidade do estudante (CSTA, 2016).

As linguagens de programação visual são consideradas adequadas para serem utilizadas no início do processo de ensino e aprendizagem de algoritmos e lógica de programação, pois possibilitam aos estudantes o desenvolvimento do raciocínio lógico a partir do foco na resolução de problemas, tornando a ação de programar menos complexa (Tóth; Lovászová, 2021).

A linguagens de programação visual, diferentes das linguagens de programação textuais, mostram o feedback das ações em tempo real, pois o desenvolvimento acontece imediatamente, de forma interativa e visual por meio de blocos de programas (Perin; Silva; Valentim, 2021).

Acreditamos que a disciplina de algoritmo e lógica de programação pode ser um elo para integrar diferentes áreas do conhecimento. Por exemplo, conceitos de matemática, de lógica e da língua inglesa podem ser explorados juntamente com ambientes de linguagem de programação visual.

Considerando este contexto, o presente trabalho busca responder a seguinte pergunta de pesquisa: “As linguagens de programação visual em blocos possuem potencialidade para serem utilizadas como ferramenta auxiliar para o ensino de lógica e programação?" Para responder tal questionamento, o presente trabalho procura avaliar as funcionalidades da linguagem de programação visual Alice 3D, App Inventor e Scratch, para, em segundo momento, explorar questões interdisciplinares. A ideia deste estudo é investigar seus benefícios para viabilizar, de maneira mais fácil, o ensino e aprendizagem de conteúdos interdisciplinares, de difícil compreensão e de alto nível de abstração.

Busca-se, por meio do estudo a respeito das possibilidades da utilização das linguagens de programação visual, considerada por muitos autores, um método eficaz, a viabilidade de utilizá-las em âmbito multidisciplinar. Portanto, a avaliação das ferramentas para programação visual utilizadas irá analisar aspectos pedagógicos e técnicos de cada software testado.

Este artigo está organizado da seguinte maneira: a seção 2 apresenta os trabalhos correlatos; a seção 3 aborda as linguagens de programação visual Alice 3D, App Inventor e Scratch; a seção 4 descreve os métodos utilizados na pesquisa; a seção 5 apresenta e discute os resultados; e, por fim, a seção 6 reúne as considerações finais.

2 TRABALHOS CORRELATOS

Os trabalhos correlatos apresentados nessa seção reforçam a justificativa de aplicar softwares de linguagens de programação visual para explorar aspectos interdisciplinares.

Dentre os trabalhos escolhidos, destaca-se a pesquisa de Silva e Coutinho (2022), que investiga a importância da utilização da linguagem de programação visual Scratch para o ensino de língua inglesa, para crianças, incluindo na formação destes conceitos básicos de língua inglesa e lógica de programação. Os resultados mostram que os participantes da pesquisa entendem a importância de utilizar linguagens de programação visual para tornar o processo ensino e aprendizagem mais divertido, além de compreenderem que, por meio da lógica de programação e da língua inglesa, é possível desenvolver habilidades e competências essenciais para os dias atuais, tanto em docentes quanto em estudantes.

Os questionamentos da pesquisa, respondidas por 71 pessoas, de diferentes áreas, como: educação, computação e engenharia, mostra que a ampla maioria (59,16%) dos entrevistados

acham extremamente interessante a ideia de ensinar lógica de programação e da língua inglesa desde as séries iniciais, enquanto 33,8% dos participantes consideram o tema muito interessante e apenas 7,04% dos entrevistados da pesquisa enxergam isso com pouco entusiasmo.

Outro resultado relevante, encontrado no trabalho de Silva e Coutinho (2022), diz respeito à complexidade do ensino de algoritmos e lógica de programação e da língua inglesa para crianças. Para a maioria (49,3%) dos participantes da pesquisa, o ensino de algoritmos e lógica de programação e da língua inglesa para crianças é razoavelmente complexo, enquanto, para 38,02% dos entrevistados, esta tarefa pode ser considerada muito complexa. Apenas 7,04% dos participantes acreditam que é fácil ensinar algoritmos e lógica de programação e língua inglesa para crianças e, quatro entrevistados consideram a esta proposta extremamente complexa.

Outra pesquisa selecionada, foi o trabalho de Costa *et al*. (2022), que investigou, junto a 100 estudantes, o potencial da linguagem de programação visual Scratch para facilitar o processo de ensino e aprendizagem de lógica de programação. Os resultados obtidos podem ser considerados significativos, pois a partir da utilização do Scratch, os estudantes buscaram, por meio de uma postura autônoma, a resolução de problemas durante a fase da programação, além de demonstrarem um comportamento mais colaborativo e crítico durante a realização das atividades.

Pensando na avaliação de softwares educativos, o trabalho de Nascimento (2016) propõe avaliar três softwares educativos (*The Huxley*, URI *Online Judge* e Uva) para o ensino e aprendizagem de programação. Utilizando o método Reeve, que lista dez critérios sobre interface e quatorze critérios pedagógicos, os quais são avaliados por meio de uma representação gráfica, os autores buscam entender as características dos softwares educacionais, a partir de diversos aspectos. Os resultados mostram que, no quesito interface, a maioria dos critérios foram atendidos em todos os softwares educacionais analisados. Semelhante à avaliação da interface, os critérios pedagógicos, em sua maioria, foram contemplados, o que demonstra a importância de avaliar softwares educacionais para servir de apoio para o ensino e aprendizagem de programação, devido aos altos índices de evasão e retenção (Nascimento, 2016).

As avaliações de software no aspecto pedagógico e educacional têm sido muito exploradas ao longo dos anos, como a pesquisa de Reategui e Finco (2010), que serviu, como base, para discussão deste trabalho.

3 LINGUAGEM DE PROGRAMAÇÃO VISUAL

Diferente das linguagens de programação baseadas em texto, que privilegiam estruturas lineares e modulares, as linguagens de programação visual apresentam blocos de diferentes cores e formas, que se encaixam, semelhante à um puzzle, tornando o ensino e aprendizagem, não apenas de conteúdos relacionados a computação, mais eficiente e divertido, diminuindo significativamente as dificuldades intrínsecas de algoritmos e lógica de programação (Tóth; Lovászová, 2021).

O ensino e aprendizagem de algoritmos e lógica de programação, por meio de linguagens de programação visual, pode auxiliar o desenvolvimento do raciocínio lógico, bem como a criatividade e a autonomia do estudante (Perin; Silva; Valentim, 2021).

Por meio de funcionalidades acessíveis e atraentes, como ícones, botões e símbolos, as linguagens de programação visual possuem características específicas, que indicam comandos e funções exibidos de maneira a auxiliar os estudantes sem experiência em programação (Begosso; Begosso; Aragão, 2020).

Corroborando com isso, Glushkoval (2016) acrescenta que as linguagens de programação visual, por meio de blocos de comandos com cores distintas, deixam o ensino e aprendizagem de algoritmos e lógica de programação descomplicados, aumentando o interesse dos estudantes em programar.

Para Perin, Silva e Valentim (2021), com base em uma experiência mais intuitiva, as linguagens de programação visual têm potencial para tornar o ensino e aprendizagem de algoritmos e lógica de programação mais atrativo.

Diferentes linguagens de programação visual estão disponíveis para serem utilizadas de apoio, não apenas para o ensino e aprendizagem de algoritmos e lógica de programação, mas de conteúdos que podem ser trabalhados de forma multidisciplinar (Tóth; Lovászová, 2021).

Nesse sentido, a próxima seção trata, respectivamente, do ambiente Alice, App Inventor e Scratch - linguagens de programação visual escolhidas para este trabalho.

3.1 Alice 3D

A linguagem de programação visual Alice 3D deixa o ensino e aprendizagem de programação mais atraente, pois diminui significativamente a complexidade de programar, oferecendo ao estudante uma abordagem de programação visual e intuitiva (Dann; Cooper; Pausch, 2005).

A IDE *(Integrated Development Environment*) Alice fornece um ambiente de programação interativo para criar animações, histórias ou jogos, que disponibiliza recursos de arrastar e soltar blocos, ideal desenvolver o raciocínio lógico dos estudantes (Idrees; Aslam, 2022).

Para Vegso (2015), o ambiente de programação tridimensional, da IDE Alice, auxilia o desenvolvimento da lógica de programação, pois disponibiliza componentes gráficos interativos e oferece o feedback das ações em tempo real.

Além de auxiliar no ensino e aprendizagem de algoritmos e lógica de programação, a linguagem de programação visual Alice oferece aos estudantes o conceito de orientação a objetos, por meio de seu ambiente de programação em blocos descomplicado e interativo (Idrees; Aslam, 2022).

3.2 App Inventor

A linguagem de programação visual App Inventor foi pensada a partir da necessidade de tornar o ensino e aprendizagem de algoritmos e lógica de programação mais fácil, visando provocar mudanças positivas nas disciplinas que envolvem programação (Xie; Abelson, 2016).

O funcionamento do App Inventor ocorre a partir do desenho da interface do aplicativo, que acontece na tela Design e da programação dos componentes, na tela Block Editor, possibilitando ao usuário usar sua criatividade para desenvolver diferentes aplicações para dispositivos móveis (Hardesty, 2010).

Para Hardesty (2010), sem a necessidade de escrever linhas de código e decorar sintaxes, típicas de linguagens de programação textual, o App Inventor possibilita que o estudante, por meio do encaixe de blocos codificados, entenda mais facilmente conteúdos relacionados à programação.

O App Inventor pode ser considerado uma linguagem de programação visual, baseada em blocos lógicos, que possibilita aos usuários o desenvolvimento de aplicativos móveis para dispositivos Android (Xie; Abelson, 2016).

3.3 Scratch

O Scratch é uma linguagem de programação visual, que possibilita aos usuários, criar animações, histórias interativas e até jogos, por meio da conexão de blocos lógicos, sem a necessidade de digitar comandos (Majed, 2014).

Para Maloney *et al*. (2010), os blocos de instruções coloridos e intuitivos, presentes na linguagem de programação visual Scratch, anulam os erros de sintaxe e diminuem a dificuldade da criação de scripts, pois indicam apenas os encaixem possíveis para determinado código.

Os blocos lógicos disponíveis no Scratch são interativos e seu formato lembra peças de quebra-cabeça, tornando o ensino e aprendizagem de algoritmos e lógica de programação descomplicado e divertido (Resnick *et al*., 2009).

Utilizando a linguagem de programação visual Scratch, os estudantes desenvolvem o raciocínio lógico de maneira mais rápida, pois este tipo de abordagem favorece a resolução de problemas (Maloney *et al*., 2010).

4 MÉTODO

A presente pesquisa caracteriza-se pela aplicação de um conjunto de procedimentos científicos, seguindo estratégias metodológicas de investigação com o objetivo de buscar respostas para os problemas tratados neste trabalho, tendo como base a metodologia científica proposta por Gil (2008).

Quanto aos objetivos, esta pesquisa pode ser considerada explicativa, pois busca, por meio de uma revisão da literatura, investigar o tema da programação visual no ensino interdisciplinar.

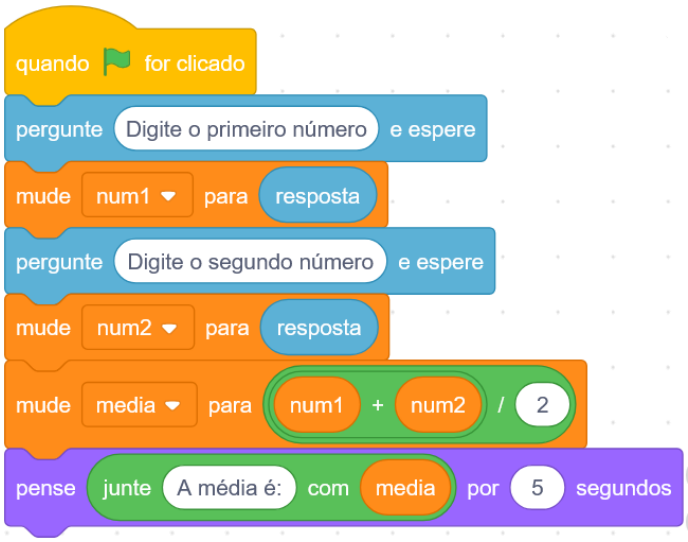
No que se refere à abordagem da pesquisa, pretende-se utilizar a abordagem qualitativa-quantitativa, também conhecida como abordagem mista, uma vez que os dados obtidos tanto na revisão da literatura quanto no estudo de caso serão analisados de forma quantitativa, por meio de questões fechadas, e de forma qualitativa, a partir de análises detalhadas das respostas às questões abertas.

Por fim, para a avaliação das ferramentas analisadas neste trabalho, serão utilizadas as diretrizes de avaliação apresentadas no Quadro 1 e detalhadas na próxima seção. Essas diretrizes contemplam aspectos técnicos, como desempenho e portabilidade, além de aspectos relacionados à interface, como estética e interatividade.

5 RESULTADOS E DISCUSSÕES

Este trabalho aplica as diretrizes para avaliação técnica de objetos de aprendizagem desenvolvidas pelos autores Reategui e Finco (2010), para avaliar as linguagens de programação visual, descritas na seção 3 deste trabalho. Para auxiliar na avaliação técnica das ferramentas, foi desenvolvido uma aplicação para calcular a média aritmética de dois números.

Figura 1 – Cálculo da média



Fonte: Elaborada pelos Autores (2024)

O quadro 1 foi realizado com base na pesquisa dos autores acima, com adaptações do próprio autor.

Quadro 1 – Diretrizes para avaliação técnica e pedagógica

		Critérios	Questões
Aspectos técnicos	**REQUISITOS**	Desempenho (1)	(1.1) É isento de erros? (1.2) No caso de problemas inesperados, o objeto continua sua execução, permitindo ao usuário completar sua tarefa? (1.3) É robusta?
		Portabilidade (2)	(2.1) O objeto de aprendizagem pode ser utilizado em computadores com configurações diversas, das mais simples até as mais sofisticadas? (2.2) O objeto pode ser utilizado em computadores com diferentes sistemas operacionais (ex. Linux, Windows, MacOS)?
		Emprego de imagens e animação (3)	(3.1) As imagens são empregadas para ilustrar conceitos e explicações ao invés de apenas decorar as páginas? (3.2) O número de imagens apresentados em cada página é adequado, considerando-se que a presença excessiva de imagens pode gerar sobrecarga cognitiva - terminando por prejudicar os processos de aprendizagem?
		Apresentação de informações (4)	(4.1) Há contraste suficiente entre fontes e fundo de tela, facilitando a leitura dos textos? (4.2) As fontes utilizadas apresentam tamanho adequado, ou permitem que sejam aumentadas/diminuídas de acordo com a necessidade de cada usuário?

	INTERFACE	Orientação e navegação (5)	(5.1) A todo o momento é possível saber em que ponto nos encontramos no objeto de aprendizagem, através de seus rótulos e títulos? (5.2) Os links para acessar outras páginas e funções do objeto de aprendizagem são facilmente reconhecíveis, através do uso de convenções universais (ex. links sublinhados ou em negrito, botões facilmente identificáveis)? (5.3) Os ícones que dão acesso a outras páginas e funções do objeto são facilmente compreensíveis?
		Interatividade (6)	(6.1) Os recursos interativos empregados vão além da seleção de links e botões para avançar ou recuar na apresentação dos conteúdos? (6.2) Os recursos interativos exploram a possibilidade do usuário alterar configurações do sistema de modo a obter respostas diferentes de acordo com suas ações?
		Estética (7)	(7.1) O objeto de aprendizagem emprega recursos gráficos que melhoram o aspecto estético da interface, tornando mais aprazível
		Funcionalidades (8)	(8.1) O ambiente disponibiliza recursos e funções para auxiliar a programação?
		Armazenamento persistente (9)	(9.1) O ambiente possui suporte nativo para armazenamento persistente de dados? (9.1) O ambiente possui suporte para armazenamento persistente de dados, através de uma API (*Application Programming Interface*)?

Fonte: Adaptado de Reategui e Finco (2010)

As linguagens de programação visual Alice, App Inventor e Scratch desempenham um papel fundamental para apoiar o ensino e aprendizagem, não apenas em conteúdos relacionados à computação, mas pensando em requisitos de de software, especificamente no desempenho, as linguagens de programação visual, em condições adversas, não estão isentas de erros, em decorrência de problemas inesperados ou pelo uso intensivo. De acordo com a experiência dos autores, quanto maior o número de funcionalidades utilizadas simultaneamente, as ferramentas Alice e Scratch apresentaram problemas, travando ou reiniciando o ambiente de programação. Já o App Inventor, canal de comunicação entre cliente e servidor. Quanto à robustez, na aplicação, notou-se que a IDE Alice utiliza mais memória e processador do computador, comparado ao App Inventor e Scratch, devido ao maior número de funcionalidades gráficas.

Ainda em requisitos de software, pensando em portabilidade, a aplicação desenvolvida para apoiar a avaliação técnica funcionou, tanto em computadores limitados, quanto nos mais potentes, nas linguagens de programação visual App Inventor e Scratch. Já a IDE Alice, devido às suas aplicações, que utilizam gráficos 3D, precisou ser instalada em um computador mais potente. Pensando em diferentes sistemas operacionais, as três linguagens, conforme testes, podem ser utilizadas no Linux, Windows e MacOS.

Na perspectiva de interface de software, pensando no emprego de imagens proposto por Reategui e Finco (2010), é possível afirmar que as linguagens de programação visual, escolhidas para esse trabalho, atendem esse requisito. Na IDE Alice e Scratch, quando o usuário seleciona a imagem, recebe instruções intuitivas das ações que podem ser realizadas para animar o objeto, por exemplo: ir para a direita, para a esquerda, para frente ou para trás. No App Inventor, as imagens escolhidas mostram combinações e blocos lógicos específicos para serem executados.

Pensando na apresentação de informações, na realização da aplicação, notou-se que as três ferramentas, testadas neste trabalho, não apresentam contraste suficiente entre fontes e fundo de tela, bem como não permitem que sejam aumentadas/diminuídas de acordo com a necessidade, prejudicando usuários de baixa visão. Cabe ressaltar que na Conferência Scratch 2023, um dos temas abordados foi a importância do desenvolvimento de ferramentas de acessibilidade, para privilegiar pessoas de baixa visão, e promover a equidade (SCRATCH, 2023).

Durante o desenvolvimento da aplicação, tanto na IDE Alice, App Inventor e Scratch, sempre foi possível perceber em que ponto da aplicação estávamos. A navegação no App Inventor e Scratch é mais simplificado, porém na IDE Alice, devido ao elevado número de funcionalidades, a navegação exige do usuário mais domínio da ferramenta, o que a torna mais complexa. Somado a isso, as convenções universais foram facilmente identificáveis durante o desenvolvimento da aplicação, nos três ambientes testados. Em relação aos ícones que dão acesso a outras páginas e funções, no App Inventor e Scratch, estão facilmente compreensíveis, enquanto na IDE Alice, devido a falta de tradução da linguagem para o português, pode causar dificuldade aos usuários que não tem familiaridade com a língua inglesa, pois as propriedades das funções estão disponíveis apenas neste idioma.

Tanto a IDE Alice, App inventor e Scratch, em termos de interatividade, disponibilizam recursos interativos, que vão além da seleção de links e botões para avançar ou recuar na apresentação dos conteúdos, além de oferecer a possibilidade de criar jogos interativos que possibilitam ao usuário alterar configurações do sistema de modo a obter respostas diferentes de acordo com suas ações. Durante o desenvolvimento da aplicação, notou-se que as três linguagens de programação visual disponibilizaram blocos lógicos interativos para complementar a aplicação.

Referente à estética, para os autores, a ferramenta mais aprazível e que melhor emprega os recursos, é a linguagem de programação visual Alice. A interface da IDE Alice disponibiliza cenários gráficos e animação 3D, que possibilitam a criação de aplicações robustas.

Pensando nas funcionalidades, todos os ambientes de programação vistos neste trabalho, possuem inúmeros recursos disponíveis para auxiliar os usuários que não têm conhecimento de programação. A IDE Alice possui diversas funcionalidades 3D e funções que possibilitam desenvolvimento de aplicações, inclusive mundos virtuais. O ambiente App Inventor tem potencial para ser utilizado na criação de aplicativos, pois possui componentes que permitem inserir funções como: mídias, bluetooth, tradutor, cloudDB, GPS, LEGO® MINDSTORMS®, além da possibilidade de integração com as redes sociais. Já o Scratch, a maioria de suas funcionalidades estão relacionadas a efeitos gráficos, com o intuito de animar os personagens.

Por fim, a IDE Alice não possui suporte para armazenamento persistente de dados, enquanto no ambiente de programação Scratch, é possível desenvolver aplicações que utilizam banco de dados, apenas por meio de uma API. Já a linguagem de programação visual App Inventor possui suporte nativo para armazenamento persistente de dados, como os componentes CloudDB e TinyDB, e através da API Firebase, plataforma de desenvolvimento de aplicativos gerenciada pelo Google.

A partir disso, foi elaborada uma planilha para avaliar os ambientes de programação, baseada nas diretrizes para avaliação técnica e pedagógica, idealizadas por Reategui e Finco (2010) e modificadas pelos autores. A planilha de avaliação pode ser observada no Quadro 2.

Foi pensado em uma estrela, para as linguagens de programação que não atendem os critérios estabelecidos; duas estrelas para aqueles que atendem parcialmente e três estrelas para linguagens de programação que atendem totalmente os critérios estabelecidos para avaliação.

Quadro 2 – Planilha da avaliação

Critérios	Alice	App Inventor	Scratch
1	★★★	★★	★★
2	★★	★★★	★★★
3	★★	★★	★★
4	★★	★★	★★
5	★★	★★★	★★★
6	★★★	★★★	★★★
7	★★★	★★	★★
8	★★★	★★	★★

9	★	★★★	★★

Fonte: Elaborado pelos Autores (2024)

Por meio da planilha, foi possível perceber que a IDE Alice e o App Inventor foram as linguagens de programação visual que mais atendem totalmente os critérios estabelecidos para avaliação, entretanto a IDE Alice foi a única que não atendeu um critério proposto nesta pesquisa.

6 CONSIDERAÇÕES FINAIS

Os conteúdos relacionados ao ensino e aprendizagem de algoritmos e lógica de programação estimulam o raciocínio lógico e auxiliam na resolução de problemas. Mesmo apresentando desafios, o ensino e aprendizagem de algoritmos e lógica de programação é fundamental para os estudantes desenvolverem competências relacionadas ao século XXI. Uma alternativa possível, para diminuir estes desafios, é utilizar linguagens de programação visual, como Alice 3D, App Inventor e Scratch, para apoiar o ensino e aprendizagem de algoritmos e lógica de programação. O objetivo desta pesquisa é investigar os benefícios das linguagens de programação visual, para viabilizar, de maneira mais fácil, o ensino e aprendizagem de conteúdos interdisciplinares.

De acordo com nossa avaliação, a IDE Alice e App Inventor são as linguagens de programação visual que apresentam maior número de critérios que atendem totalmente os critérios estabelecidos para avaliação.

O Scratch é indicado à professores do Ensino Médio, para aqueles que desejam ensinar tecnologia e introdução à lógica de programação, uma vez que o ambiente é mais simples, com reduzido número de funcionalidades. Com o intuito de desenvolver uma aplicação robusta, com cenários gráficos e animação 3D, sugere-se a IDE Alice. O App Inventor é indicado para o desenvolvimento de aplicativos para dispositivos móveis com sistema operacional Android.

Para trabalhos futuros, pretende-se investigar o critério de acessibilidade nas linguagens de programação visual abordadas nesta pesquisa.

REFERÊNCIAS

BEGOSSO, L. C.; BEGOSSO, L. R.; ARAGÃO, N. An analysis of block-based programming environments for CS1. **In: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 2020,** Uppsala. Proceedings [...]. [S. l.]: IEEE, 2020. p. 1-5. Disponível em: https://doi.org/10.1109/FIE44824.2020.9273982 . Acesso em: 29 jul. 2026.

COSTA, N. *et al*. O uso da plataforma Scratch como ferramenta facilitadora durante o ensino de lógica de programação para alunos do ensino médio. **Brazilian Journal of Development**, Curitiba, v. 8, n. 8, p. 59279-59293, 2022. Disponível em: https://doi.org/10.34117/bjdv8n8-286 . Acesso em: 29 jul. 2026.

CSTA. **K-12 computer science framework**. [S. l.]: Computer Science Teachers Association, 2016. Disponível em: https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf . Acesso em: 29 jul. 2026.

DANN, W., COOPER, S., & PAUSCH, R. **Learning to program with Alice**. New York: Prentice Hall. 2005.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Atlas, 2008.

GLUSHKOVA, T. Application of block programming and game-based learning to enhance interest in computer science. **Journal of Innovations and Sustainability**, [S. l.], n. 1, p. 21-32, 2016. Disponível em: https://doi.org/10.51599/is.2016.02.01.21. Acesso em: 29 jul. 2026.

HARDESTY, L. **The MIT roots of Google's new software**. Cambridge, MA: MIT News, 2010. Disponível em: http://news.mit.edu/2010/android-abelson-0819. Acesso em: 29 jul. 2026.

IDREES, M.; ASLAM, F. **A comprehensive survey and analysis of diverse visual programming languages. VFAST Transactions on Software Engineering**, [S. l.], 2022. Disponível em: https://vfast.org/journals/index.php/VTSE/article/view/1009 . Acesso em: 29 jul. 2026.

KROHL, D. R; DUTRA, T. C.; DE MATOS, C. P. A interdisciplinaridade como proposta para o ensino de lógica de programação no ensino fundamental II. 2021. **Revista Conexão UEPG**. v. 17, n. 1, p. 1-14. Disponível em: https://doi.org/10.5212/Rev.Conexao.v.17.16902.23. Acesso em: 29 jul. 2026.

MAJED, M. **Title of English-language original: Learn to Program with Scratch**. Nonatec Editora Ltda, 2014.

MORAIS, C. G. B.; MENDES NETO, F. M.; OSÓRIO, A. J. M. Difficulties and challenges in the learning process of algorithms and programming in higher education: a systematic literature review. **Research, Society and Development**, Vargem Grande Paulista, v. 9, n. 10, e9429109287, 2020. Disponível em: https://doi.org/10.33448/rsd-v9i10.9287. Acesso em: 29 jul. 2026.

MOREIRA, G. L. *et al*. Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros: um estudo exploratório. **In**: ENCONTRO DE COMPUTAÇÃO DO OESTE POTIGUAR (ECOP), 3., 2018, Pau dos Ferros. **Anais** [...]. Pau dos Ferros: UFERSA, 2018. v. 2, n. 1. Disponível em: https://periodicos.ufersa.edu.br/index.php/ecop/article/view/7907. Acesso em: 29 jul. 2026.

NASCIMENTO, J. P. M. **Um estudo sobre avaliação de software para o auxílio no ensino de programação**. 2016. Trabalho de Conclusão de Curso (Licenciatura em Ciência da

Computação) – Universidade Federal da Paraíba, Rio Tinto, 2016. Disponível em: https://repositorio.ufpb.br/jspui/handle/123456789/3320 . Acesso em: 29 jul. 2026.

PERIN, A. P. J.; SILVA, D. E.; VALENTIM, N. M. C. Um benchmark de ferramentas de programação em blocos que podem ser utilizadas nas salas de aula do Ensino Médio. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 32., 2021, Online. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 1162-1173. Disponível em: https://doi.org/10.5753/sbie.2021.217765. Acesso em: 29 jul. 2026.

REATEGUI, E.; FINCO, M. Proposta de diretrizes para avaliação de objetos de aprendizagem considerando aspectos pedagógicos e técnicos. **RENOTE: Revista Novas Tecnologias na Educação**, Porto Alegre, v. 8, n. 3, 2010. Disponível em: https://doi.org/10.22456/1679-1916.18066. Acesso em: 29 jul. 2026.

RESNICK, M. *et al*. Scratch: programming for all. **Communications of the ACM**, New York, v. 52, n. 11, p. 60-67, 2009. Disponível em: https://doi.org/10.1145/1592761.1592779. Acesso em: 29 jul. 2026.

SCRATCH. **Web Content Accessibility Guidelines (WCAG) 2.2**. [S. l.]: Scratch Foundation, 2023.

SILVA, N. C.; COUTINHO, E. F. Scratch como ferramenta lúdica para o ensino de língua inglesa. **Research, Society and Development**, Vargem Grande Paulista, v. 11, n. 9, e19511931201, 2022. Disponível em: https://doi.org/10.33448/rsd-v11i9.31201. Acesso em: 29 jul. 2026.

TÓTH, T.; LOVÁSZOVÁ, G. Mediation of knowledge transfer in the transition from visual to textual programming. **Informatics in Education**, Vilnius, v. 20, n. 3, p. 489-511, 2021. Disponível em: https://doi.org/10.15388/infedu.2021.20. Acesso em: 29 jul. 2026.

VEGSO, J. Interest in CS as a major drops among incoming freshmen. **Computing Research News**. 17(3), 236-248. 2015. Disponível em: https://cra.org/CRN/articles/may05/vegso. Acesso em: 29 jul. 2026.

XIE, B.; ABELSON, H. Skill progression in MIT App Inventor. *In*: IEEE SYMPOSIUM ON VISUAL LANGUAGES AND HUMAN-CENTRIC COMPUTING, 2016, Cambridge. **Proceedings** [...]. [S. l.]: IEEE, 2016. p. 213-217. Disponível em: https://doi.org/10.1109/VLHCC.2016.7739687. Acesso em: 29 jul. 2026.

ZORZO, A. F. **et al**. **Referenciais de Formação para os Cursos de Graduação em Computação**. Porto Alegre. Sociedade Brasileira de Computação. 2017. ISBN 978-85-7669-424-3. Disponível em: https://doi.org/10.5753/sbc.ref.2017.134. Acesso em: 29 jul. 2026.