USO DO APLICATIVO MIT *APP INVENTOR* NA APRENDIZAGEM DE PROGRAMAÇÃO: UMA REVISÃO SISTEMÁTICA DA LITERATURA ENTRE 2011 e 2020

THE USE OF MIT APP INVENTOR IN PROGRAMMING LEARNING: A SYSTEMATIC REVIEW OF LITERATURE BETWEEN 2011 and 2020

USO DE LA APLICACIÓN MIT APP INVENTOR EN EL APRENDIZAJE DE PROGRAMACIÓN: UNA REVISIÓN SISTEMÁTICA DE LA LITERATURA ENTRE 2011 Y 2020

Rosana Gomes Costa
Universidade de Lisboa.
E:mail costagomes2045@gmail.com

João Manuel Nunes Piedade
Professor do Instituto de Educação da Universidade de Lisboa e investigador da Unidade de I&D, Unidade de Investigação e Desenvolvimento Educação e Formação (UIDEF).
E-mail: jmpiedade@ie.ulisboa.pt.
Orcid: 0000-0002-4118-397X

RESUMO

A aprendizagem da programação envolve a compreensão de um conjunto de teorias, práticas e o conhecimento sobre paradigmas, sintaxes e semânticas das linguagens, competências de raciocínio lógico, matemático e abstrato e algorítmico. Estes aspetos tornam a aprendizagem difícil para os alunos inciantes, sendo uma disciplina com elevadas taxas de reprovação e evasão. Vários estudos têm apontado os ambientes de programação visual e por blocos como ferramentas que podem ajudar a ultrapassar algumas das barreiras inicias. Este artigo apresenta uma Revisão Sistemática da Literatura sobre a utilização do ambiente de programação por blocos *MIT App Inventor* no ensino e aprendizagem de programação. Para tal, desenvolveu-se uma pesquisa sistemática de artigos publicados, entre 2011 e 2020, em seis das principais bases de dados relacionadas com as ciências da computação e educação. Os resultados dos 10 artigos analisados salientaram as potencialidades do ambiente para a aprendizagem de conceitos de programação e revelaram, igualmente, a falta de estudos experimentais que analisem o efeito da sua utilização na aprendizagem e na motivação dos alunos.

Palavras-chave: Ambiente de programação por blocos. Aprendizagem de programação. MIT *app inventor*. Revisão sistemática.

ABSTRACT

Learning to program involves understanding a set of theories and practices, knowledge about language paradigms, syntax and semantics, logical, mathematical, and abstract and algorithmic reasoning skills. These aspects make learning difficult for beginning students, being a subject with high failure and dropout rates. Several studies have pointed out the visual and block programming environments as tools that can help overcome some of the initial barriers. This article presents a Systematic Literature Review on the use of the MIT *App Inventor* block programming environment in teaching and learning to program. To this end, a systematic search of articles published between 2011 and 2020 was carried out in six of the main databases related to computer science and education. The results of the 10 articles analyzed highlighted the potential of the environment for learning programming concepts and also revealed the lack of experimental studies that analyze the effect of its use on students' learning and motivation.

Keywords: Block-based programming. Learning programming. MIT app inventor. Systematic review.

RESUMEN

Aprender a programar implica comprender un conjunto de teorías y prácticas, conocimiento sobre paradigmas del lenguaje, sintaxis y semántica, habilidades de razonamiento lógico, matemático y abstracto y algorítmico. Estos aspectos dificultan el aprendizaje de los estudiantes principiantes, al ser una asignatura con altos índices de reprobación y deserción. Varios estudios han señalado los entornos de programación visual y de bloques como herramientas que pueden ayudar a superar algunas de las barreras iniciales. Este artículo presenta una revisión sistemática de la literatura sobre el uso del entorno de programación de bloques MIT App Inventor para enseñar y aprender a programar. Se realizó una búsqueda sistemática de artículos publicados entre 2011 y 2020 en seis de las principales bases de datos relacionadas con la informática y la educación. Los resultados de los 10 artículos analizados destacaron el potencial del entorno para el aprendizaje de conceptos de programación y también revelaron la falta de estudios experimentales que analicen el efecto de su uso en el aprendizaje y la motivación de los estudiantes.

Palabras clave: Aprendiendo a programar. Programación basada en bloques. MIT app inventor. Revisión sistemática.

INTRODUÇÃO

A aprendizagem da programação envolve a compreensão de um conjunto de teorias e práticas, bem como o conhecimento sobre paradigmas, sintaxes e semânticas das linguagens, competências de raciocínio lógico, matemático e abstrato e de pensamento algorítmico (PIEDADE *et al*., 2019). Estes aspetos tornam a aprendizagem um processo complexo e difícil para muitos estudantes (MARTINS; MENDES; FIGUEIREDO, 2013; JENKINS, 2002). De acordo com Robins *et al.* (2003), citado por Cheng (2019, p. 362) "[...] one of the major challenges facing novice programming pupils was that they knew the syntax and semantics of each single statement in the programming language, but they did not understand how to combine the statements to create a valid computer program [...]". No mesmo sentido, Deters *et al.* (2008) afirmam que as disciplinas de programação são consideradas desafiadoras pelos alunos, pois são aquelas em que a reprovação e a desistência está muito presente.

As principais dificuldades evidenciadas em alguns estudos, apontam para a falta de habilidades dos alunos relativamente à resolução de problemas, ao pensamento algorítmico e matemático, ao raciocínio lógico, ao tempo dedicado aos estudos e apontam, igualmente, a idade imatura dos alunos nos cursos introdutórios (FERREIRA *et al*., 2017; SILVA *et al*., 2018). Salcedo e Idobro (2011) referem, ainda, o fato de os estudantes recorrerem em demasia à memoriação como forma de estudo, o que na opinião dos autores é insuficiente na aprendizagem da programação que requer muita prática e um comportamento dinâmico. O processo de aprendizagem de programação é lento, gradual,

progressivo e requer um treino intensivo e elevados níveis de concentração (HOLANDA *et al.*, 2018; SILVA *et al.*, 2018). De fato, programar é muito mais do que codificar um conjunto de intruções ou pedaços de código, e a primeira dificuldade dos alunos é compreender aquilo que estão escrevendo e qual o papel de cada instrução na construção do programa final (BOSSE; GEROSA, 2017).

Além das relativas dificuldades sinalizadas aos alunos, encontramos em alguns trabalhos evidências a dificuldades relacionadas aos métodos didáticos e pedagógicos, bem como associadas às crenças e competências dos professores (GOMES *et al.*, 2008; MOONS; BRACKER, 2013).

Aos professores de programação coloca-se, por vezes, um dilema sobre qual o ambiente de programação utilizar em suas aulas. A escolha de um ambiente mais profissional (e.g. textual) ou um ambiente pedagogicamente mais acessível para os alunos, como, por exemplo, os ambientes de programação visual e por blocos. Os ambientes profissionais são poderosas ferramentas (NAVARRETE, 2013), e mais cedo ou mais tarde, os alunos terão contato com elas, mas são igualmente complexas para alguns alunos (MARTINS; MENDES; FIGUEIREDO, 2013). Maya *et al.* (2015) ressaltam que muitos professores acreditam que apenas as experiências de programação recorrendo a linguagens textuais, tais como Java ou C++, permitem aprender a programar.

Pensando em formas de ultrapassar as dificuldades evidenciadas pelos alunos, alguns estudos vêm apontando a utilização de ambientes de programação visual e por blocos como ferramentas com potencialidades pedagógicas para ajudar a minimizar algumas das dificuldades. Esses ambientes permitem reduzir as dificuldades relacionadas com a elevada complexidade da semântica e sintaxe de algumas linguagens textuais e o nível de abstração exigido (GÖKÇE *et al.*, 2017; MIHCI; OZDENER, 2017; PIEDADE *et al.*, 2019) e aumentar os níveis de motivação dos alunos para a aprendizagem da programação, em particular para os iniciantes (DOLGOPOLOVAS *et al.*, 2017). Os ambientes de programação por blocos consiste em uma forma de programação visual que permite aos alunos criar programas mediante a conjugação de blocos de instruções similares a pequenas peças *Lego*. Nesse tipo de ambiente as instruções e estruturas de programação estão organizadas em categorias coloridas e esta organização ajuda os alunos a selecionaram as instruções e estruturas adequadas para a construção dos seus programas (LYE; KOH, 2014). Essa forma de programação previne alguns erros de sintaxe, pois não permite encaixar os blocos caso essa operação resulte numa instrução errada. Muitos dos erros de sintaxe mais comuns

verificados nas linguagens textuais são eliminados usando esses ambientes (KOLLIN *et al.*, 2017).

Atualmente existem muitos ambientes de programação por blocos disponíveis de forma totalmente gratuita, o mais conhecido e utilizado é o *Scratch* desenvolvido pelo *Lifelong Kindergarten Group do MIT Media Lab*. Procurando avaliar as principais características, potencialidades e diferenças entre os diversos ambientes disponíveis, Piedade *et al.* (2019) analisaram 26 destes ambientes e concluíram que, apesar de muitas características comuns associadas à forma de programação, existem alguns aspectos que os distinguem. Os autores identificaram características distintas no tocante aos conceitos de programação que podem ser trabalhados, da tipologia de tarefas, da adequabilidade às diferentes faixas etárias, da possibilidade de evolução na aprendizagem de programação e da viabilidade de programação de objetos tangíveis e programação para *mobile*.

Recentemente tem emergido uma nova tipologia de ambientes híbridos que permitem a programação por blocos e em modo textual. Nesses tipos de ambientes os alunos podem construir blocos de códigos e analisar como esse programa será representado em uma determinada linguagem de programação textual (e.g. *Python, C/C++, JavaScript*). Tal estratégia pode ser muito interessante para os alunos inciantes na aprendizagem de programação, pois permite aos alunos evoluir para a programação mais profissional de forma sustentada e gradual (WEINTROP; HOLBERT, 2017).

Dentre os ambientes analisados por Piedade et al. (2019) destaca-se o MIT *App Inventor*, desenvolvido pelo *Massachusetts Institute of Tecnology*, por meio das suas potencialidades no desenvolvimento de aplicações para *mobile* com sistema operacional *android*. O ambiente MIT *App Inventor* possibilita que a aprendizagem de conceitos essenciais de programação ocorra de forma significativa, uma vez que os mesmos são trabalhados de modo intuitivo e motivador (FILIZOLA *et al.*, 2014). O ambiente é propício para a aprendizagem de algoritmos de forma interativa baseada na construção de aplicativos que serão utilizados em dispositivos móveis (SERALIDOU *et al.*, 2019).

Papadakis (2019) considera que o *App Inventor* é uma ferramenta ideal para estimular o interesse dos alunos, pois ajuda a reduzir os obstáculos na aprendizagem dos conceitos introdutórios de programação. Finizola *et al.* (2014), reportando o resultado de um estudo com alunos do ensino médio, pontuam que o uso desse ambiente pode ser mais atraente e motivador para os alunos, na medida em que os resultados das ações de

programação são imediatos e o aluno não tem que lidar com a angústia e preocupação dos erros de sintaxe (SERALIDOU *et al.*, 2019).

Considerando as potencialidades dessa ferramenta, pareceu-nos pertinente a necessidade de sistematizar o conhecimento já produzido sobre a sua utilização na aprendizagem dos alunos. Assim, no tópico seguinte, apresentamos a estruturação e os resultados da revisão sistemática da literatura sobre a utilização do ambiente MIT *App Inventor* no ensino e aprendizaem de programação.

Revisão sistemática da literatura

A revisão sistemática da literatura foi desenvolvida com o objetivo de pesquisar o conhecimento existente na literatura acadêmica sobre a utilização do ambiente como ferramenta para aprender e ensinar programação. Procurou-se, assim, uma abordagem sistemática e pragmática para a análise de literatura mediante uma metodologia clara, transparente e passível de ser replicada (GOUGH; OLIVER; THOMAS, 2012). As revisões sistemáticas permitem obter sínteses abrangentes e imparciais de investigações relevantes num determinado domínio, procurando relatar dados e resultados em vez de conceitos ou teorias (AROMATARIS; PEARSON, 2014). Esse tipo de estudo auxilia o investigador a nortear o desenvolvimento de projetos, apresentando novos caminhos para futuras pesquisas e analisar quais métodos de pesquisa foram utilizados em uma área científica.

O protocolo de pesquisa foi organizado considerando as etapas definidas por Kitchenham (2004) e que apresentamos nos tópicos que se seguem.

Questões de Pesquisa

De modo a responder ao objetivo definido para a busca de evidências sobre a utilização do ambiente MIT *App Inventor* no ensino da programação, foram definidas as seguintes questões de pesquisa:

a) Q1. Qual o tipo de utilização do ambiente MIT *App Inventor* na aprendizagem da programação relatada nos estudos?
b) Q2. Quais os níveis de ensino contemplados nas pesquisas?
c) Q3. Quais as metodologias de aprendizagem / instrutivas adotadas nos estudos?

d) Q4. Quais os principais resultados evidenciados sobre a utilização do ambiente *MIT App Inventor* na aprendizagem da programação?

e) Q5. Que potencialidades e limitações emergem nos estudos sobre a utilização do ambiente *MIT App Inventor*?

Fontes e critérios de pesquisa

O processo de pesquisa foi desenvolvido recorrendo às principais bases de dados relacionadas com as ciências da computação e educação: i) Scopus; (ii) ISI *WebofScience*; (iii) ACM – *Association for Computing Machinery Digital Library;* (iv) ERIC – *Education Resources Information Center*; (v) IEEE *Xplore Digital Library*; e (vi) DOAJ – *Directory Open Access Journals*. A pesquisa foi realizada entre os meses de fevereiro e março de 2020, baseada em um conjunto de critérios definidos *à priori*: (i) artigos publicados entre 2011 e 2020, devido ao facto de ambiente *MIT App Inventor* ter sido disponibilizado em 2011; (ii) artigos de acesso aberto; (iii) estudos publicados em Inglês e Português;

De acordo com o foco de pesquisa, foram utilizados como termos de pesquisa "*App Inventor*" **and** "*Programming*" **or** *App Inventor*" **and** "*Coding*", que deveriam estar presentes no título ou no resumo do trabalho. Através da equação de pesquisa, obtivemos o retorno de um conjunto de 66 artigos com a distribuição, pelas fontes, representada na Tabela 1.

Tabela 1 - Total de artigo por base de dados

Base de Dados	Número de Artigos
Scopus	15
ISI WebofScience	20
ACM – Association for Computing Machinery Digital Library	6
ERIC – Education Resources Information Center	9
IEEE Xplore Digital Library	10
DOAJ – Directory Open Access Journals	6

Assim, a primeira versão da base de dados foi construída com 66 artigos, por sua vez organizados numa planilha de Excel. Em seguida, procedeu-se a análise dos títulos dos artigos, autores e fontes de publicação de modo a identificar possíveis artigos repetidos. Após a primeira análise foram eliminados 16 artigos duplicados.

Para um segundo momento de análise definiram-se como critérios de inclusão: (i) estudos sobre o uso do MIT *App Inventor* como ferramenta para ensinar programação (excluindo a utilização do ambiente para outras atividades de aprendizagem em que o foco não é a programação); (ii) participantes do ensino médio (secundário) ou superior; (iii) estudos empíricos de caráter quantitativo, qualitativo ou misto; (iv) artigos publicados em revistas com revisão por pares; (v) estudos com teoria de aprendizagem ou instrutiva definida.

Foram selecionados os 50 artigos para a leitura do resumo, introdução e conclusão para excluir os que não se enquadravam nos critérios de inclusão ora referidos. No final da análise, foram eliminados 16 artigos apresentados em conferência ou pôsteres e 24 artigos que eram relatos de trabalhos em andamento e/ou revisões de literatura. No final da referida fase resultou um conjunto de 10 artigos para a análise final.

Para análise dos resultados, o conteúdo dos artigos foi registrado em uma planilha contendo: título do artigo, nome do(s) autor(es), fonte, índice SJR (*Scimago Journal Rank*), ano de publicação, resumo, palavras-chave, objetivo geral, categoria, amostra, metodologia de pesquisa, teoria de aprendizagem/método instrutivo, principais resultados, região geográfica e base de dados.

De modo a reduzir a análise dos estudos publicados em revistas indexadas com fator de impacto, optou-se por critério final a análise dos estudos publicados em revistas Q1, Q2, Q3 e Q4. O SJR é um indicador bibliométrico que mede as produções científicas através do número de citações recebidas, permitindo, também ser medido o prestígio científico do periódico. Na Tabela 2 apresentam-se os títulos dos artigos, respectivos autores, ano de publicação, a revista onde foram publicados e a respectiva codificação.

Tabela 2 - Lista de publicações selecionadas

Código	Título	Autores	Ano	SJR	Publicação
E1	Mobile App Design for Teaching and Learning: Educators' Experiences in an Online Graduate Course	Yu-Chang Hsu; Yu-Hui Ching	2013	Q1	International Review of Research in Open and distance Learning
E2	Microworlds, games, animations, mobile apps, puzzle editors and more: What is important for an introductory programming environment?	Stelios Xinogalos; Maya Satratzemi; Christos Malliarakis	2015	Q1	Journal Education and Information Technologies

E3	An Analysis of Mathematics Education Students' Skills in the Process of Programming and Their Practices of Integrating It into Their Teaching	Semirhan Gökçe; Arzu Aydoğan Yenmez; İlknur Özpınar	2017	Q3	International Education Studies
E4	From Android games to coding in C—An approach to motivate novice engineering students to learn programming: A case study	Vladimiras Dolgopolovas; Tatjana Jevsikova; Valentina Dagiene	2017	Q1	Computer Applications in Engineering Education
E5	Using App Inventor for creating apps to support m-learning experiences: A case study	Antonio Ortega-García; Antonio Ruiz-Martínez; Rafael Valencia-García	2017	Q1	Computer Applications in Engineering Education
E6	Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy	Chun-Yen Tsai	2018	Q1	Computers in Human Behavior
E7	CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs	Christiane Gresse Von Wangenheim; Jean C. R. Hauck; Matheus Faustino Demetrio; Rafael Pelle, Nathalia da Cruz Alves; Heliziane Barbosa; Luiz Felipe Azevedo	2018	Q2	Informatics in Education
E8	Comparing the Effectiveness of Scratch and App Inventor with Regard to Learning Computational Thinking Concepts	Youngki Park; Youhyun Shin	2019	Q1	Electronics
E9	Extending Smart Phone Based Techniques to Provide AI Flavored Interaction with DIY Robots, over Wi-Fi and LoRa interfaces	Dimitrios Loukatos; Konstantinos G. Arvanitis	2019	Q4	Education Sciences
E10	Modeling Different Variables in Learning Basic Concepts of Programming in Flipped Classrooms	Hatice Yildiz Durak	2019	Q1	Journal of Educational Computing Research

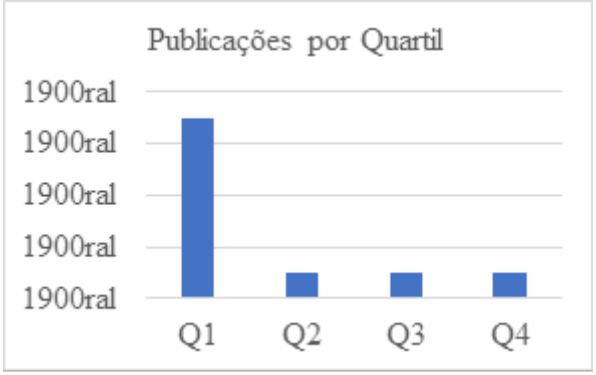
Resultados

A análise dos artigos selecionados se concentrou na resposta às 5 questões de pesquisa definidas *à priori*, apresentando-se em seguida a síntese dos principais resultados encontrados.

Os 10 artigos analisados foram publicados entre os anos de 2013 e 2019, conforme distribuição representada no Gráfico 1.

Gráfico 1 - Distribuição dos artigos por ano de publicação

Gráfico 2 - Publicações por Quartil do *Scientific Journal Ranking* (SJR)



A análise revelou que a maioriea dos artigos (7) foi publicada em revistas posicionadas no 1° Quartil do Scientific Journal Ranking (SJR) e os restantes 3 em revistas do 2°,3° e 4° quartis respectivamente (Gráfico 2).

A distribuição geográfica das publicações encontra-se representada no Gráfico 3, verificando-se que a maioria representa estudos que foram desenvolvidos nos Estados Unidos da América, não foi encontrando qualquer estudo desenvolvido no contexto brasileiro ou português. Todos os estudos analisados foram publicados em língua Inglesa.

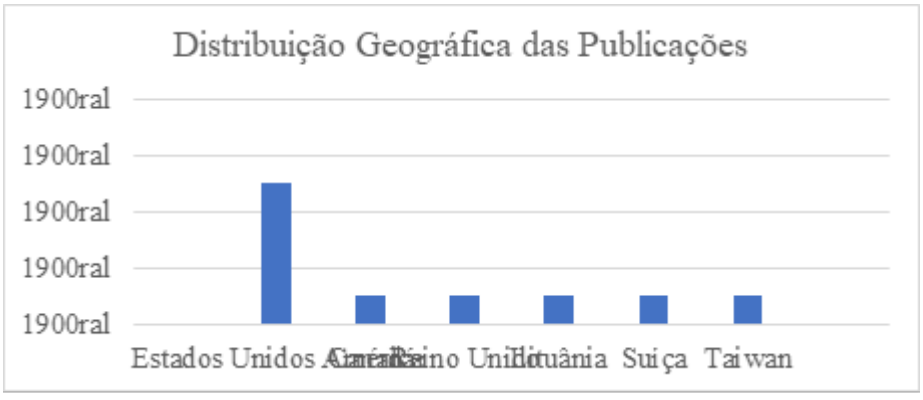


Gráfico 3 - Distribuição Geografia das Publicações

Dos estudos analisados, oito foram desenvolvidos na educação superior, envolvendo como participantes majoritariamente alunos de graduação e apenas dois foram desenvolvidos na educação básica e secundária (ensino médio).

Todos os estudos apresentam investigações empíricas, visto se tratar de um dos critérios definidos para a sua inclusão na revisão sistemática. Em relação à dimensão e design metodológico, verificou-se que sete estudos foram organizados segundo metodologias de caráter quantitativo, dois de caráter qualitativo e um misto.

Os estudos de natureza quantitativa usaram como instrumentos de recolha de dados, majoritariamente, questionários e escalas *self-report*, um dos estudos (E3) recorreu à utilização do *Delphi Research Report* com recursos a pré e pós-teste e um dos estudos (E6) apresenta um design quase-experimental que aplica em pré e pós-testes de conhecimentos uma escala de autoeficácia em programação.

Os estudos de natureza qualitativa recorreram a entrevistas, rubricas e análise qualitativa das tarefas de aprendizagem realizadas pelos alunos como instrumentos de recolha de dados.

Relativamente às teorias de aprendizagem e estratégias instrutivas aplicadas, dois estudos referem aprendizagem baseada em problemas, aprendizagem baseada em projetos, aprendizagem baseada em jogos, um estudo refere aprendizagem construtivista, um estudo aprendizagem construcionista, um autoaprendizagem, um aprendizagem metacognitiva, um aprendizagem baseada em design, um sala de aula invertida e dois não definem claramente as metodologias ou estratégias instrutivas.

Na seção seguinte, discutiremos os principais resultados evidenciados nos 10 estudos selecionados para análise.

Discussão dos resultados

Gökçe *et al*. (2017) (E3) analisaram as habilidades dos alunos, futuros professores de matemática, na utilização do ambiente de programação MIT *App Inventor* para aprender a programar, procurando analisar o impacto da aprendizagem da programação no pensamento crítico, na resolução de problemas e nos estilos de pensamento holístico e analítico.

Dolgopolovas *et al.* (2017) (E4) usaram como proposta desenvolver jogos utilizando o ambiente de aprendizagem MIT *App Inventor* para alunos iniciantes na disciplina de programação.

Em um estudo com alunos de um curso de informática e de comunicação, Ortega-García *et al.* (2017) (E5) trabalharam com o ambiente de programação para criar aplicativos

m-learning que auxiliaram alunos a aprender alguns conceitos relacionados com banco de dados.

Para o processo de análise e discussão dos principais resultados encontrados, recuperaremos as questões de pesquisa 1, 4 e 5, bem como os dados relativos às questões 2 e 3 que foram apresentados na seção anterior.

A questão 1 norteou a análise do tipo de utilização do ambiente MIT *App Inventor* na aprendizagem da programação relatada nos estudos.

A utilização do *MIT App Inventor* mostrou-se de grande utilidade para a criação de aplicativos móveis, mediando a possibilidade e praticidade de ensinar/aprender *on-line* (E1) (HSU; CHING, 2013).

Xinogalos, Satratemi e Malliarakis (2015) (E2), examinaram o desempenho e os recursos disponibilizados pelo ambiente de programação *MIT App Inventor* para introduzir conceitos introdutórios de programação para iniciantes.

Tsai (2018) (E6) analisou o uso da linguagem de programação visual, recorrendo ao MIT *App Inventor* para melhorar a compreensão e os conhecimentos dos alunos iniciantes na programação, com diferentes níveis de autoeficácia.

Em seu estudo, Wangenheim *et al.* (2018) (E7) usaram a aplicação *CodeMaster* com a finalidade de avaliar e classificar programas desenvolvidos por alunos usando os ambientes de programação *MIT App Inventor* e *Snap!*.

Park e Shin (2019) (E8) utilizaram uma nova rubrica com o Dr. Scratch para avaliar os projetos desenvolvidos por alunos usando o *Scratch* e o MIT *App Inventor* relativamente a conceitos de pensamento computacional e programação.

Loukatos e Arvantis (2019) (E9) analisaram a percepção dos alunos com histórico de dificuldades de aprendizagem e horário limitado sobre a utilização do MIT *App Inventor* para desenvolverem atividades e tarefas de aprendizagem de programação.

Durak (2019) propôs avaliar o desempenho dos alunos iniciantes na aprendizagem de conceitos básicos de programação usando ambiente de programação MIT *App Inventor* recorrendo à metodologia de aprendizagem sala de aula invertida.

Na questão de pesquisa 4, procuramos analisar os principais resultados evidenciados sobre a utilização do ambiente *MIT App Inventor* na aprendizagem da programação, em particular, a aprendizagem de conceitos introdutórios.

No estudo desenvolvido por Hsu e Ching (2013) (E1), os participantes, alunos iniciantes com pouca ou nenhuma experiência em programação, consideraram que o

ambiente de programação *MIT App Inventor* foi de grande valia para criação de aplicativos móveis e de fácil utilização à distância, sendo um fator motivador para a aprendizagem. Os dados revelaram, ainda, níveis favoráveis de sentimento de pertença à comunidade criada, bem como níveis igualmente favoráveis de envolvimento nas tarefas de aprendizagem.

Os autores concluem que o ambiente supra citado poderá ser uma ferramenta útil para motivar os alunos para a aprendizagem de programação através do seu engajamento em atividades criativas de resolução de problemas, recorrendo ao desenvolvimento de aplicações para dispositivos móveis.

Essas mesmas potencialidades foram referidas por Xinogalos, Satratemi e Malliarakis (2015) (E2) em um estudo em que um grupo de alunos analisou as características de 5 ambientes de programação visual, entre eles o MIT *App Inventor*. Os participantes do estudo salientaram como características que podem motivar e envolver os alunos na aprendizagem, a usabilidade e facilidade de utilização, a possibilidade de valer-se da ferramenta para iniciar a programação para *Android*, a facilidade no desenho e programação de *Apps* para dispositivos móveis, a facilidade de testar os programas desenvolvidos e o suporte ao aluno no desenvolvimento dos programas.

Em um estudo com alunos de graduação, futuros professores de matemática, Gokçe *et al.* (2017) (E3) utilizaram o MIT *App Inventor* para introduzir conceitos de programação e perceber o impacto da aprendizagem de programação no desenvolvimento de competências de pensamento crítico e resolução de problemas. Os resultados do estudo mostraram que aprender a programar não teve efeito significativo sobre as habilidades de resolução de problemas dos participantes. Contudo, foram evidenciados efeitos significativos no pensamento holístico dos participantes, verificando um aumento significativo nos valores médios da escala no pós-teste em comparação ao pré-teste.

Dolgopolovas *et al.* (2017) (E4) usaram o ambiente MIT *App Inventor* com o objetivo de reduzir as dificuldades iniciais evidenciadas pelos alunos na aprendizagem de programação utilizando a linguagem C. Os autores concluíram que a utilização do ambiente poderá ser um fator motivador para os alunos, pelo facto de permitir aprender os conceitos iniciais de programação mediante o desenvolvimento de aplicações que podem facilmente ser integradas e disponibilizadas no *Google Play*. Alertam, contudo, que se o objetivo é fazer os alunos evoluírem para linguagens textuais, como por exemplo a linguagem C, as tarefas de aprendizagem devem ser cuidadosamente desenhadas com esse propósito.

O MIT *App Inventor* permitiu a adaptação e modificação de todos os conteúdos antes da aula, de acordo com os conceitos nos quais os alunos apresentaram mais problemas e dificuldades, conforme apontado por Ortega-García *et al.* (2017) (E5).

Tsai (2018) (E6) relatou que a utilização de linguagens de programação visual, recorrendo ao MIT *App Inventor*, melhorou a compreensão dos alunos sobre os conceitos básicos de programação, em especial nos alunos com baixa autoeficácia no grupo experimental. Os dados revelaram que a autoeficácia tem um papel preponderante na aprendizagem de conceitos de programação. Com base nos resultados, o autor conclui que o ambiente de programação estudado tem um grande potencial para aprendizagem de conceitos iniciais de programação no contexto universitário.

Wangenhein *et al.* (2018) (E7) desenvolveram uma ferramenta para avaliar projetos de programação realizados por alunos do ensino médio, a partir do uso dos recursos e dos ambientes MIT *App Inventor* e Snap!. Nesse sentido, Park e Shin (2019) (E8), recorrendo à ferramenta Dr. Scratch, avaliaram um conjunto de projetos desenvolvidos por alunos usando *Scratch* e *MIT App Inventor*. Mediante a análise dos resultados, os autores concluem que os projetos desenvolvidos em *Scratch* tiveram pontuações mais elevadas nos conceitos de paralelismo, sincronização e controlo de fluxo, enquanto os projetos desenvolvidos no MIT *App Inventor* tiveram pontuações mais elevadas nos conceitos de interatividade do utilizador e representação de dados. Contudo, os autores resalta que as pontuações dos programas tendem a aumentar à medida que a complexidade dos programas se expande e que essas mesmas pontuações estão altamente dependentes da tipologia dos projetos desenvolvidos pelos alunos.

Loukatos e Arvantis (2019) (E9) concluíram que o uso do MIT *App Inventor* contribuiu positivamente para a aquisição de habilidades de programação e para o desenvolvimento de aplicativos. Os autores reforçam, ainda, resultados positivos para alunos com dificuldades de aprendizagem de programação.

Durak (2019) (E10) revelou que o ambiente de programação visual apresentou vários pontos positivos para o aluno, a sintaxe simples, a possibilidade de codificação sem erros, e por ter uma depuração fácil. Esses aspectos ajudam os alunos a ultrapassar algumas das dificuldades no contacto inicial com a programação.

Por último, a questão de pesquisa 5 procurou identificar que potencialidades e limitações emergem nos estudos analisados, sobre a utilização do ambiente MIT *App Inventor*.

Os vários estudos analisados apresentaram um conjunto de características do ambiente MIT *App Inventor*, que por sua vez podem potencializar a aprendizagem da programação. A sua tipologia de programação visual permite tornar a sintaxe mais simples e compreensível, evitar a ocorrência de erros de código (DURAK, 2019), facilidade de utilização e de testagem dos programas, a usabilidade e interação com utilizador (XINOGALOS; SATRATEMI; MALLIARAKIS, 2015).

O ambiente permite o desenvolvimento de aplicações para dispositivos móveis, e esta característica, pode ser importante para a motivação dos alunos e para a aprendizagem de programação (DOLGOPOROVAS *et al.*, 2017; HSU; CHING, 2013; XINOGALOS *et al.*, 2015), principalmente por parte dos alunos com maiores dificuldades de aprendizagem (LOUKATOS; ARVANTIS, 2019; ORTEGA-GARCÍA *et al.*, 2017; TSAI, 2018). Xinogalos; Satratemi; Malliarakis (2015) reportaram a possibilidade de iniciar a programação para *Android* como uma potencialidade interessante do ambiente, que pode ser de extrema utilidade para os alunos.

Relativamente às limitações da plataforma, alguns alunos participantes nos estudos apontaram a língua inglesa como uma limitação (GÖKÇE *et al.*, 2017; XINOGALOS; SATRATEMI; MALLIARAKIS (2015), no entanto a versão atual da plataforma está disponível em 15 línguas diferentes. Outro relato seria que o ambiente permite pouco controle sobre como e onde os componentes são incluídos no design da tela (HSU; CHING, 2013). Outra limitação reportada foi o fato de o ambiente apenas permitir desenvolver aplicações para *Android*, sendo desejável que permitisse o desenvolvimento para outras plataformas (ORTEGA-GARCÍA *et al.*, 2017).

CONSIDERAÇÕES FINAIS

Nesta breve revisão sistemática da literatura procuramos analisar estudos que focassem a utilização do ambiente MIT *App Inventor* para ensinar conceitos introdutórios de programação com alunos iniciantes. Desse modo, foi nossa intenção sistematizar o conhecimento científico produzido e publicado sobre as potencialidades desse ambiente no ensino da programação, em particular na análise do seu impacto nas aprendizagens dos alunos.

Os estudos analisados permitiram identificar algumas das potencialidades da utilização do MIT *App Inventor*, bem como algumas das suas limitações. Salienta-se que a

maioria dos estudos foi desenvolvido com alunos de ensino superior, sendo que apenas dois estudos foram realizados com alunos do ensino secundário, ficando evidente a ausência de estudos no âmbito do ensino médio ou secundário. Outro aspecto sinalizado foi a ausência de estudos empreendidos em contexto brasileiro e português.

Ficou evidente a dificuldade em encontrar estudos experimentais que permitam analisar o efeito do ambiente MIT *App Inventor* nos conhecimentos de programação dos alunos em comparação com a utilização das clássicas linguagens textuais (C, C++, Java).

Apenas um dos estudos analisados assumiu um design quase experimental, que permitiu explicar e analisar os efeitos da utilização do ambiente nos conhecimentos e na autoeficácia.

Desse modo, como trabalho futuro nos propomos a desenvolver um estudo experimental com alunos do ensino médio-técnico com o objetivo de analisar os efeitos da utilização do MIT *App Inventor*, em associação com uma estratégia instrutiva baseada em problemas, nos conhecimentos de programação e na motivação dos alunos para a aprendizagem.

REFERÊNCIAS

AROMATARIS, Edoardo; PEARSON, Alan. The systematic review: an overview. **American Journal of Nursing**, v. 114, n. 3, 2014, p. 53-58.

BOSSE, Yorah; GEROSA, Marco. Hy is programming so difficult to learn?: patterns of difficulties related to programming learning mid-stage. **ACM SIGSOFT Software Engineering Notes,** v. 41, n. 6,2017, p. 1-6.

CHENG, Gary. Exploring factors influencing the acceptance of visual programming environment among boys and girls in primary schools. **Computers in Human Behavior**, v. 92, n. 1, mar. 2019, p. 361-372.

DOLGOPOLOVAS, Vladimiras; JEVSIKOVA, Tatjana; DAGIENE, Valentina. From android games to coding in C-An approach to motivate novice engineering students to learn programming: a case study. **Computer Applications in Engineering Education,** v. 26, n. 1, 2017.

DURAK, Hatice. Modeling different variables in learning basic concepts of programming in flipped classrooms. **Journal of Educational Computing Research**, v. 58, n. 1, 2019, p. 160-199.

FERREIRA, Fábio et al. Aprendizagem na programação: um modelo de continuidade de aprendizagem de programação. In: Iberian conference on information systems and technologies, 12., 2017. **Anais** [...]. Lisboa, Portugal: CIST, 2017. p. 1-6.

FINIZOLA, António et al. O ensino de programação para dispositivos móveis utilizando o MIT-App inventor com alunos do ensino Médio. In: Workshop de informática na escola, 20., 2014. **Anais** [...] Porto Alegre: SBC, 2014. p. 337-341.

GOMES, Anabela; HENRIQUES, Joana; MENDES, António. Uma proposta para ajudar estudantes com dificuldades na aprendizagem inicial de programação de computadores. **Educação, Formação & Tecnologias**, v. 1, n. 1, 2008, p. 93-103.

GÖRÇE, Semirhan; YENMEZ, Arzu; ÖZPINAR, Ilknur. An analysis of mathematics education students' skills in the process of programming and their practices of integrating it into their teaching. **International Education Studies,** v. 10, n. 8, 2017. p. 61-76.

GOUGH, David; THOMAS, James; OLIVER, Sandy. Clarifying differences between review designs and methods. **Systematic Reviews**, v. 1, n. 1, 2012. p. 28.

HOLANDA, Wallace; COUTINHO, Jarbele; FONTES, Laysa. Uma intervenção metodológica para auxiliar a aprendizagem de programação introdutória: um estudo experimental. In: Congresso brasileiro de informática na educação, 7., 2018. **Anais** [...] Porto Alegre: SBC, 2018. p. 699-708.

HSU, Yu- Chang; CHING, Yu-Hui. Mobile app design for teaching and learning: educators' experiences in an online graduate course. **International Review of Research in Open and Distance Learning,** v. 14, n. 4, 2013. p. 117-139.

JENKINS, Tony. On the difficulty of learning to program. In: Annual conference of LTSN-ICS, 3., 2002. **Anais** [...] Loughborough: UK, 2002. Disponível em: http://www.psy.gla.ac.uk/~steve/localed/jenkins.html. Acesso em: 10 set. 2020

KOLLING, Micahel; BROWN, Neil; ALTADMRI, Amjad. Frame-based editing. **J. Vis. Lang. Sentient Syst**, v. 3, 2017. p. 40-67.

LOUKATOS, Dimitrios; ARVANITIS, Kostas. Extending smart phone based techniques to provide aI flavored interaction with DIY robots, over Wi-Fi and LoRa interfaces. **Education Sciences**, v. 9, n. 3, 2019, p. 1-18.

LYE, Sze; KOH, Joyce. Review on teaching and learning of computational thinking through programming: what is next for K-12?. **Computers in Human Behavior**, v. 41, 2014. p. 51-61.

MARTINS, Scheila Wesley; MENDES, Antonio Jose; FIGUEIREDO, António. D. D. Diversifying activities to improve student performance in programming courses. **Commun. Cogn**, v. 46, n. 1, jun. 2013. p. 39-58.

MAYA, Israel et al. Supporting all learners in school-wide computational thinking: a cross-case qualitative analysis. **Computer & Education**, v. 82, 2015. p. 263-279.

MIHCI, Can; OZDENER, Nesrin. Programming education with a blocks-based visual language for mobile application development. In: International association for the development of the information society, 1., 2014. **Anais** [...] Madrid, 2014. p. 149-156.

MOONS, Jan; BACKER, Carlos. The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. **Computers & Education**, v. 60, n. 1, 2013. p. 368-384.

NAVARRETE, Cesar C. Creative thinking in digital game design and development: a case study. **Computers & Education**, v. 69, n. 1, set. 2013. p. 320-331.

ORTEGA-GARCÍA, Antonio; RUIZ-MARTÍNEZ, Antonio; VALENTE-GARCÍA, Rafael. Using app inventor for creating apps to support m-learning experiences: a case study. **Computer Applications in Engineering Education**, v. 26, n. 3, 2017.

PARK, Youngki; SHIN, Youhyun. Comparing the effectiveness of scratch and app inventor with regard to learning computational thinking concepts. **Electronics**, v. 8, n. 1269, 2019. p. 1-12.

PIEDADE, João et al. A Cross-analysis of block-based and visual programming apps with computer science student-teachers. **Education Sciences**, v. 9, n. 181, jul. 2019. p. 1-19.

SALCEDO, Sebatian; IDOBRO, Ana. New tools and methodologies for programming languages learning using the scribbler robot and Alice. In: ASEE/IEEE frontiers in education conference, 41., 2011. **Proceedings [...]**. IEEE: Rapid City, 2011. p. F4G1-F4G-6.

SERALIDOU, Eleni; DOULIGERIS, Christos. Learning with the app inventor programming software through the use of structured educational scenarios in secondary education in Greece. **Education and Information Technologies**, v. 24, 2019. p. 2243-2281.

SILVA, Walquirua et al. Levantamento sobre as dificuldades dos discentes nas disciplinas de programação no curso técnico de Informática. **Diversitas Journal,** v. 3, n. 3,2018, p. 761-770.

TSAI, Chun-Yen. Improving students' understanding of basic programming concepts through visual programming language: the role of self-efficacy. **Computers and Human Behavior**, v. 95, 2019. p. 224-232.

WANGENHEIM, Christiane et al. CodeMaster – automatic assessment and grading of app inventor and snap! Programs. **Informatics in Education**, v. 17, n. 1, 2018. p. 117-150.

WEINTROP, David; HOLBERT, Nathan. From blocks to text and back: programming patterns in a dual-modality environment. In: ACM technical symposium on computer science education, 48., 2017. **Proceedings** [...] New York, USA: ACM, 2017.

XINOGALOS, Stelios; SATRATEMI, Maya; MALLIARAKIS, Christos. Microworlds, games, animations, mobile apps, puzzle editors and more: what is important for an introductory programming environment?. **Educations and Information Technologies**, v. 22, p. 145-176, 2017.

Recebido em: 11/12/2020
Parecer em: 04/01/2021
Aprovado em: 10/02/2021