MIT App Inventor como Ambiente de Ensino de Algoritmos e Programação

Sérgio de Oliveira[1], Marconi de Arruda Pereira[1], Fernando A. Teixeira[1]

[1]Campus Alto Paraopeba – Universidade Federal de São João del Rei (UFSJ)
Rod.: MG 443, KM 7 Ouro Branco - MG 36420-000 – Brazil

{sergiool, marconi, teixeira}@ufsj.edu.br

Abstract. *Computational environments used to teach algorithms have console interfaces and symbolic languages that do not help in the learning process. This article presents the use of the MIT App Inventor environment to develop an Algorithms course for beginners. For that, the basic contents of algorithms were mapped to structures in App Inventor. The methodology was applied to four classes of students entering higher education with good performance and optimistic observations.*

Resumo. *Ambientes computacionais usados para ensinar algoritmos possuem interfaces de console e linguagens simbólicas que não ajudam no processo de aprendizagem. Este artigo apresenta a utilização do ambiente MIT App Inventor para desenvolvimento de uma disciplina de algoritmos para iniciantes. Para tanto, os conteúdos básicos de algoritmos foram mapeados para estruturas no App Inventor. A metodologia foi aplicada a quatro turmas de ingressantes no ensino superior com bom aproveitamento e observações otimistas.*

1. Introdução

O ensino de algoritmos e programação é amplamente difundido. Cursos de graduação na área de computação, engenharias e demais cursos da área de exatas, cursos técnicos e até cursos de ensino médio mais inovadores incluem algoritmos em seus currículos. A tendência é que essa disciplina se difunda ainda mais, se tornando presente na maioria dos cursos de graduação, ensino médio e até ensino fundamental.

Os ambientes preferidos para o ensino de algoritmos têm interface baseada em console. Linguagens de programação como C/C++, Pascal, Java e Python têm sido usadas no modelo entrada-processamento-saída com apenas uma entrada padrão para o teclado e uma saída padrão em um console no modo texto. Esse modelo é mais simples para o aprendizado, mas o modelo de aplicação em console se tornou específico para usuários avançados. Novatos na computação tem pouca fluência no uso de console e isso dificulta sua motivação para o desenvolvimento de aplicações nessa interface. Além disso, essas linguagens são altamente simbólicas, o que exige um nível de concentração muito alto para estudantes que são iniciantes neste conteúdo.

Diversos ambientes de desenvolvimento de algoritmos focados na facilidade de aprendizado já foram apresentados, destacando-se o MIT Logo [Tempel 2012], MIT Scratch [Wolz et al 2009] e Google Blockly [Pasternak et al 2017]. Os dois primeiros induzem movimentos e ações sobre representações de animais em uma tela gráfica sem

efeito direto para o desenvolvimento de programas. Google Blockly tem uma interface capaz de traduzir algoritmos definidos em uma linguagem de blocos para diversas linguagens de programação, o que pode ser útil no desenvolvimento de software mas necessita de familiaridade com outros ambientes de desenvolvimento para que o resultado gerado possa ser integrado em uma aplicação real.

MIT App Inventor [Saigal e Saigal 2011] utiliza os princípios consolidados no MIT Logo e MIT Scratch em um ambiente de desenvolvimento de aplicativos para smartphones com o sistema operacional Android. O ambiente de desenvolvimento é executado pela web, com projetos e recursos salvos na nuvem mantida pelo MIT. O resultado pode ser distribuído como um aplicativo e até disponibilizado na Google Play Store do sistema Android. Essa abordagem gera maior motivação nos estudantes, pois o aplicativo é um produto palpável do esforço investido no aprendizado [Papert 1994].

Um ambiente de programação deve ter suporte a todo o conteúdo abordado em uma disciplina para que possa ser adotado. As disciplinas de algoritmos possuem ementas que tendem a contemplar: i) entrada e saída; ii) variáveis e manipulações algébricas; iii) operações de lógica boleana; iv) estruturas condicionais; v) laços de repetição; vi) sub-rotinas como funções ou procedimentos; vii) vetores e matrizes. Eventualmente, armazenamento em memória não-volátil, como arquivos, e estruturas de dados heterogêneas também podem entrar num primeiro curso de algoritmos. Assim, a escolha de um ambiente de desenvolvimento para o ensino de algoritmos deve possibilitar o exercício destes conceitos.

A principal contribuição deste trabalho é mostrar como o MIT App Inventor pode mapear todos os conteúdos comuns a disciplinas de algoritmos e programação para suas estruturas em blocos usando a interface de um aplicativo para smartphone. O artigo apresenta sugestões para definir os principais pontos comuns às ementas de uma disciplina de algoritmos de forma que um aluno iniciante possa começar a desenvolver os conceitos de algoritmos enquanto cria aplicativos.

Como resultado deste trabalho, foi desenvolvido um material composto de 18 vídeos e uma apostila com fundamentação[1], exemplos e exercícios que podem ser usados na aplicação de uma disciplina de algoritmos e programação para estudantes iniciantes de programação em nível superior ou técnico. O material foi aplicado em uma turma presencial e três turmas remotas em cursos de engenharia com grande aceitação e aproveitamento pelos estudantes.

Além desta introdução, este artigo apresenta: os trabalhos relacionados na seção 2, o MIT App Inventor na seção 2, o mapeamento dos conceitos principais na seção 4, o resultado da aplicação em algumas turmas na seção 5 e as conclusões na seção 6.

2. Trabalhos Relacionados

Diversos trabalhos na literatura apresentam experiências da utilização do App Inventor para o ensino dos princípios básicos de programação [Ribeiro et al 2016], [Ramos et al 2015], [Gomes e Melo 2012], [Pereira et al 2020] e vários outros. Os resultados apontam para ganhos no processo ensino-aprendizagem. Em nenhum deles, no entanto, há

[1] Material disponível em: https://sites.google.com/ufsj.edu.br/algapp

compromisso com o conteúdo da disciplina de algoritmos. Em geral, exploram princípios básicos de programação usando recursos de smartphones.

[Karakus et al 2012] apresenta várias possibilidades de utilização do App Inventor em disciplinas de programação introdutórias no ensino superior, com considerações para os cursos na área de computação ou não.

[Soares 2014] apresenta a utilização do App Inventor em disciplinas na área de computação focados em estudantes não-iniciantes, aproveitando recursos apresentados neste artigo, como a utilização de listas de listas para armazenar estruturas de dados heterogêneas, bem como elementos de exportação de dados e de comunicação.

3. MIT App Inventor

MIT App Inventor é um ambiente de desenvolvimento de aplicativos para o sistema operacional Android, de código aberto, criado originalmente pelo Google e mantido atualmente pelo MIT. Lançado em 2010, encontra-se na sua segunda versão. Utiliza o princípio de arrastar e soltar para desenvolver as interfaces dos aplicativos e uma linguagem de blocos baseada no OpenBlocks [Roque 2007].

Além dos componentes gráficos típicos de um aplicativo Android, o MIT App Inventor também possui biblioteca de componentes para acessar recursos do smartphone, como sensor de localização, giroscópio e câmera. Também possui bibliotecas para integração com redes sociais, armazenamento na nuvem, suporte aos mapas do OpenStreetMap.

Os recursos do MIT App Inventor habilitam sua utilização no conceito de *Computational Thinking* [Lu e Flecher 2009], visto que facilitam a expressão da solução computacional para problemas a partir da sua formulação, ligando elementos reais como localização, registros fotográficos, sons, movimentos e interações. Também está alinhado ao conceito de Construcionismo [Papert 1994], que apresenta como resultado produtos palpáveis para a construção do conhecimento. Desta forma, torna-se adequado à utilização no ensino médio e fundamental, seguindo a aplicação destes conceitos.

A criação de aplicativos de forma gráfica e interativa não necessariamente implica na utilização de algoritmos computacionais. É possível criar aplicativos apenas ligando alguns poucos componentes com mínima interação algorítmica entre eles. Vários exemplos estão disponíveis e podem ser localizados em diversos repositórios nos quais os conceitos e conteúdos de algoritmos são minimamente abordados.

A contribuição deste trabalho busca preencher essa lacuna, utilizando o MIT App Inventor para desenvolver os conceitos fundamentais de uma disciplina de algoritmos e programação. Apesar do enfoque principal do MIT App Inventor seja desenvolver aplicativos de forma rápida e sem exigências de conhecimento prévio de algoritmos e programação, é possível utilizar essa ferramenta para apresentar os principais conceitos de algoritmos no nível básico. Seria possível avançar, ainda, para um nível mais intermediário, apresentando algoritmos de ordenação e pesquisa, por exemplo. A Figura 1 apresenta a implementação do algoritmo de ordenação Quicksort usando o App Inventor bem como uma interface para a entrada e saída de dados.

O processo de implementação dos algoritmos clássicos para o App Inventor passa por algumas adaptações: não há o conceito de entrada/saída padrão, sendo necessário definir os componentes para a entrada e saída de dados, e não há uma definição de

execução sequencial do algoritmo, pois segue orientação de eventos para definir o sequenciamento de ações. Assim, para dar ênfase aos algoritmos no App Inventor, é preciso abstrair os processos de entrada e saída, bem como o tratamento de eventos.

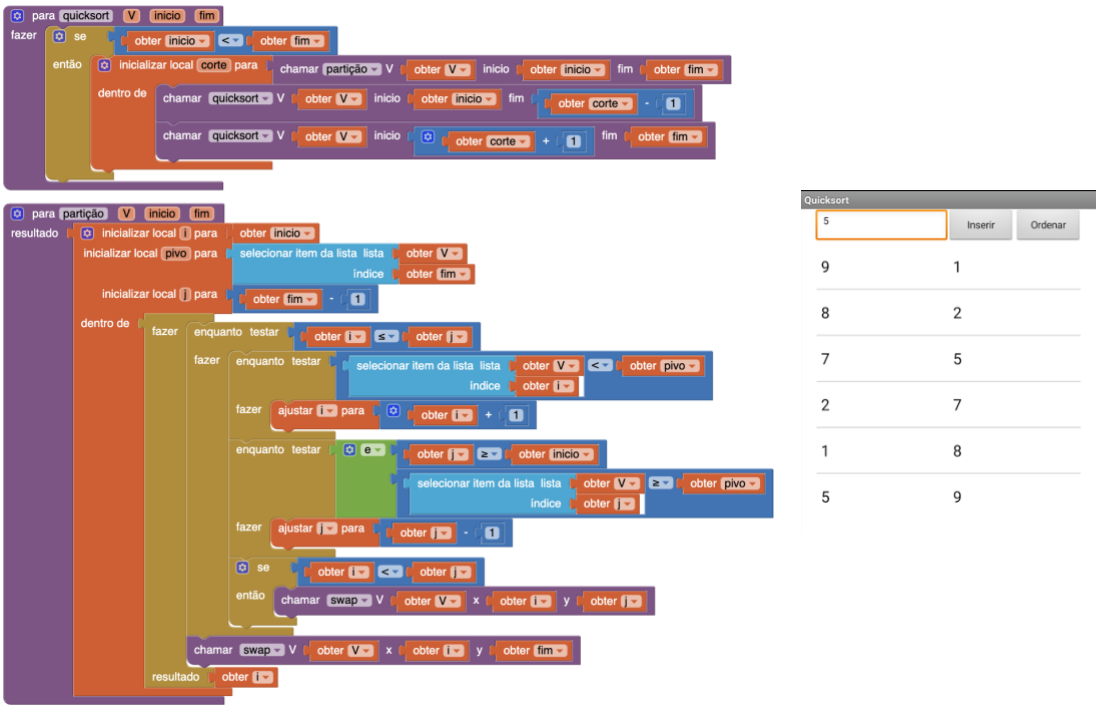

Figura 1 - Algoritmo de ordenação Quicksort no App Inventor

4. Mapeamento do Conteúdo

O enfoque principal deste trabalho foi mapear a ementa de uma disciplina de Algoritmos e Programação de um curso de Engenharia para aplicação no App Inventor. Aplicações clássicas, como manipulação de matrizes, são necessárias na disciplina pois serão usados posteriormente ao longo do curso. Para tornar o curso mais interessante, também foram apresentadas atividades envolvendo o uso de GPS, armazenamento de dados em nuvem, fotos e mapas.

4.1. Entrada, Saída e Eventos

App Inventor utiliza a biblioteca de componentes básicos do Android para a implementação de suas interfaces, o que torna o aplicativo mais leve e mais fácil a migração para outros ambientes. Assim, estão disponíveis diversos componentes de entrada e saída, entre eles: caixas de texto, legendas, caixas de senha, caixas de seleção e manipuladores de listas.

Para a implementação dos primeiros algoritmos, uma Caixa de Texto para a entrada e uma Legenda para a saída são suficientes para ler e escrever dados simples, possibilitando o funcionamento dos algoritmos básicos. A associação é mais fácil chamando a Caixa de Texto de “Entrada” e a Legenda de “Saída”. A Figura 2 mostra os componentes Caixa de Texto e Legenda com suas propriedades de acesso a seu conteúdo. As variáveis no App Inventor não são tipificadas e a conversão para inteiros e reais é automática. Logo, essas propriedades podem ser usadas diretamente em operações matemáticas, o que facilita a manipulação pelos estudantes.

Figura 2 - Campos básicos de entrada e saída

Para a leitura e escrita de dados múltiplos, como uma sequência de valores, é possível usar um botão para inserir os dados em uma lista. E o componente visualizador de listas pode ser usado para a visualização dos valores inseridos bem como saídas múltiplas. A **Figura 3** mostra como a entrada e saída dos dados para o Quicksort foram tratados.

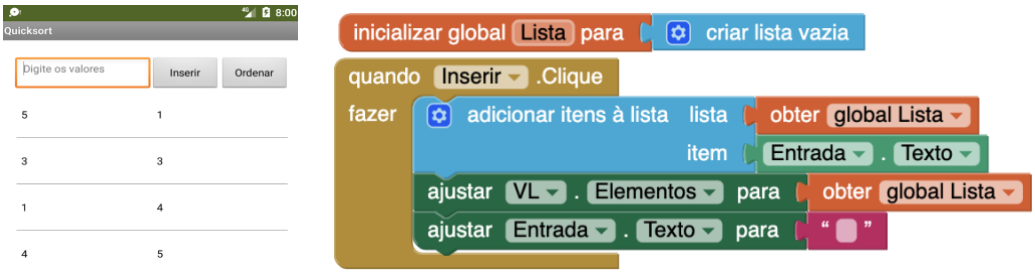

Figura 3 - Entrada e saída de dados múltiplos

O conceito de lista usado para entradas múltiplas é o mesmo usado para estruturas de dados homogêneas, visto que o App Inventor não possui estrutura de vetor. Logo, essa associação pode ser apresentada também como alternativa ao ensino de vetores.

Outro ponto que pode ser observado na **Figura 3** é o paradigma de resposta a eventos para iniciar os algoritmos. Não existe um "início" como nos programas sequenciais nas linguagens de programação textuais. Para dar início a execução de um algoritmo, é preciso lançar mão de um botão e programar um evento, a partir do bloco de tratamento de eventos disponível e associado ao botão. O evento mais comum é o clique do botão, facilmente associado ao início da execução do algoritmo. Caso sejam necessárias leituras múltiplas, como na **Figura 3**, o ideal é associar um botão à leitura de dados e outro à execução do algoritmo.

4.2. Variáveis e Manipulação Algébrica

Variáveis no App Inventor não são tipificadas e possuem conversão automática para inteiros e reais. A vantagem imediata é a possibilidade de aplicar operações sem ter de considerar o tipo e conteúdo dos dados na programação. Caso dados alfabéticos sejam considerados em fórmulas matemáticas, uma mensagem de erro é apresentada.

As operações aritméticas básicas estão disponíveis, incluindo quociente e resto da divisão inteira. Operações de álgebra booleana (E, OU e NÃO) também estão disponíveis, contemplando as operações básicas.

Em relação à variáveis, é possível considerar variáveis locais, válidas apenas dentro do blocos definidos por elas e variáveis globais, apresentados na Figura 4. A inicialização de variáveis é obrigatória. Para atribuição de valor às variáveis, o bloco "ajustar" é utilizado, evitando muitos erros dos iniciantes.

A utilização de variáveis não tipificadas, a conversão automática entre texto e números e blocos específicos para a atribuição ou leitura de variáveis, minimizam os erros de linguagem e possibilitam o enfoque no aprendizado de algoritmos.

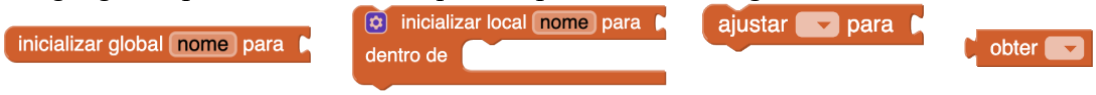

Figura 4 - Blocos para manipulação de variáveis

4.3. Estruturas de Controle

As estruturas de controle do App Inventor, sejam condicionais ou laços de repetição, são bastante intuitivas e possibilitam sua aplicação direta, tal qual nas linguagens de programação clássicas.

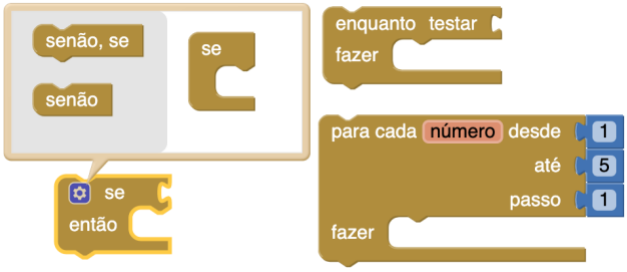

Figura 5 - Estruturas básicas de controle

A Figura 5 apresenta os blocos básico de controle. A estrutura condicional "se" pode ser configurada para incluir "senão" e um ou vários "senão, se", tornando-se uma estrutura de múltiplos caminhos de decisão.

A simplicidade e o apelo visual dos blocos de controle possibilitam que sua aplicação nos algoritmos seja direta e de fácil assimilação pelos estudantes, o que possibilita melhor aproveitamento do tempo da disciplina.

4.4. Sub-rotinas

A modularização e agrupamento de um conjunto de instruções em uma sub-rotina como função ou procedimento é essencial no ensino de algoritmos e também suportado no App Inventor. Dois tipos de sub-rotinas são suportados, com e sem retorno, denominadas ambas como procedimentos, ilustradas na Figura 6. A configuração dos parâmetros é acessada no botão azul no canto superior esquerdo.

Figura 6 - Tipos de procedimentos

A construção dos procedimentos tem o mesmo alcance das linguagens de programação convencionais, suportando procedimentos recursivos e retornos dos mais diversos tipos. Não há suporte à passagem de parâmetros por referência nem tampouco retornos múltiplos. Também é possível retornar listas.

Os procedimentos com retorno devem ter o valor de retorno associado ao encaixe do resultado. Para aritmética simples, essa associação é imediata. No entanto, para

procedimentos que contém laços e estruturas condicionais, é preciso usar blocos específicos para propagar o retorno além das estruturas, como pode ser visto na Figura 7. O segundo exemplo também ilustra a construção de procedimentos recursivos.

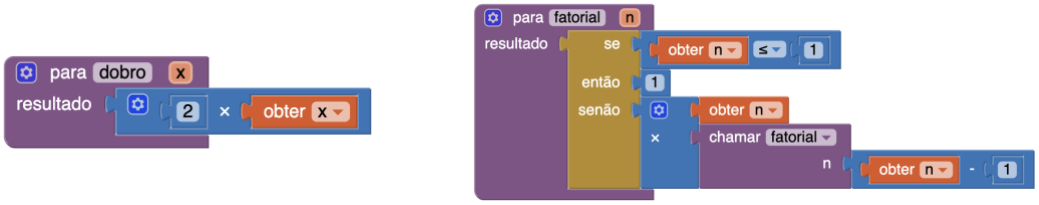

Figura 7 - Procedimentos com retorno

4.5. Vetores e Matrizes

A exemplo do que ocorre em outras linguagens como Python e Javascript, App Inventor não possui a implementação de vetores com memória limitada previamente associada. A implementação básica para manipulação de conjunto de dados homogêneos é a lista, já apresentada para a leitura de dados múltiplos. Tal qual demais implementações de listas, no ambiente App Inventor, as operações de criação, inserção, acesso, substituição de elementos e verificação do número de elementos, estão devidamente implementadas.

A maior dificuldade na utilização das listas no App Inventor se deve ao tamanho dos blocos para acesso às listas. Os principais blocos de acesso às listas são apresentados na Figura 8.

Figura 8 - Blocos de manipulação de listas

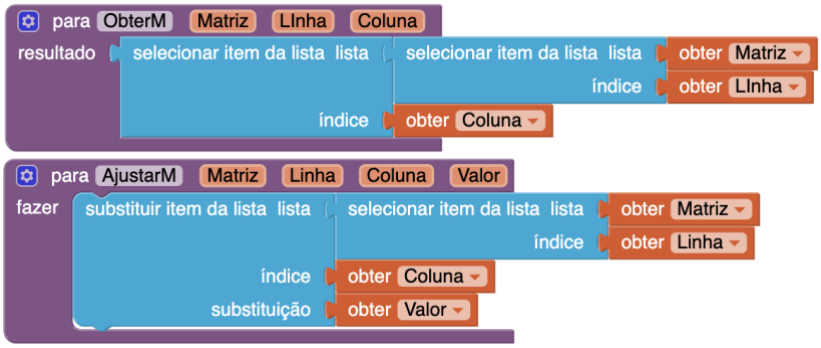

Figura 9 - Blocos para manipulação de matrizes

Os algoritmos para manipulação de vetores, como, por exemplo, determinar a mediana de um vetor ou, ainda, ordenar um vetor, podem ser implementos com relativa facilidade usando os blocos de manipulação de listas. As operações sobre matrizes, no entanto, implementadas como listas de listas, podem gerar conjuntos de blocos visualmente mais extensos e de difícil interpretação. Uma forma de reduzir a quantidade e a extensão de blocos é a utilização de procedimentos para encapsular a leitura e escrita em elementos da matriz.

A Figura 9 apresenta os blocos propostos. O primeiro “ObterM” retorna o valor armazenado na matriz, linha e coluna passados como parâmetros. O segundo, “AjustarM”, atribui o valor indicado também como parâmetro. O encapsulamento do

acesso aos elementos das matrizes por esses procedimentos é bastante útil para implementar operações como multiplicações de matrizes ou cálculo de determinantes.

A interface para a entrada de dados de uma matriz também merece atenção. Digitar os dados um-a-um é cansativo num smartphone. O que também não é diferente num console. Em ambas as situações, é possível entrar os valores linha-a-linha. A Figura 10 apresenta uma proposta de interface para um aplicativo de multiplicação de matrizes, bem como o método para tratar essa entrada. As linhas são digitadas na Caixa de Texto com os elementos separados por espaço e podem ser adicionadas à matriz M1 ou M2, de acordo com o botão acionado após a digitação dos valores. As matrizes M1, M2 e Resultado são apresentadas em visualizadores de listas.

Para adicionar os elementos digitados às matrizes, foi utilizado o bloco "repartir texto", usando um espaço em branco como delimitador para repartir os elementos. Esse bloco retorna uma lista, que já é atribuída diretamente à matriz.

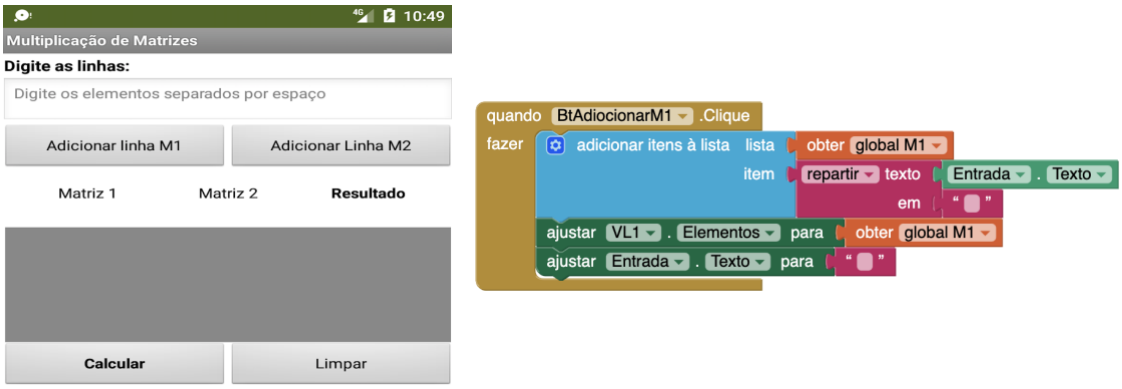

Figura 10 - Interface e tratamento para entrada de dados em matrizes

5. Registro de Aplicação

A metodologia apresentada neste artigo foi aplicada em três turmas em 2020, na modalidade de ensino remoto emergencial, implantado na universidade durante a pandemia da Covid-19.

Para uma análise quantitativa, foram considerados os estudantes aprovados e reprovados por nota, desconsiderando os reprovados por frequência. Devido ao ensino remoto, as aulas foram gravadas em 18 vídeos, com propostas de atividades avaliativas práticas e apostila disponibilizada. As três turmas participaram da disciplina apresentada em 12 semanas, seguindo calendário da universidade. O número de estudantes aprovados foi 30 em 32 (94%), 39 em 41 (95%) e 38 em 40 (95%). A Figura 11 mostra esse resultado de forma gráfica, para as turmas usando o MIT App Inventor (MAI), comparando com outras seis turmas nas quais a mesma disciplina foi ministrada usando o Python como ambiente de desenvolvimento, com taxas de aprovação de 94%, 91%, 90%, 89%, 80% e 78%, respectivamente. Os cursos em Python seguiram basicamente a mesma ementa, além da metodologia de aulas síncronas e assíncronas, atividades práticas e avaliações objetivas, aplicadas por outros três docentes para turmas com perfil similar.

Outro dado relevante para essa análise quantitativa se refere às respostas dos estudantes aos questionários de avaliação docente no item referente à "Utiliza metodologias, técnicas e recursos compatíveis com os objetivos de ensino-

aprendizagem”: 4,72; 4,71 e 4,67 em 5 pontos possíveis, respectivamente, para as três turmas consideradas.

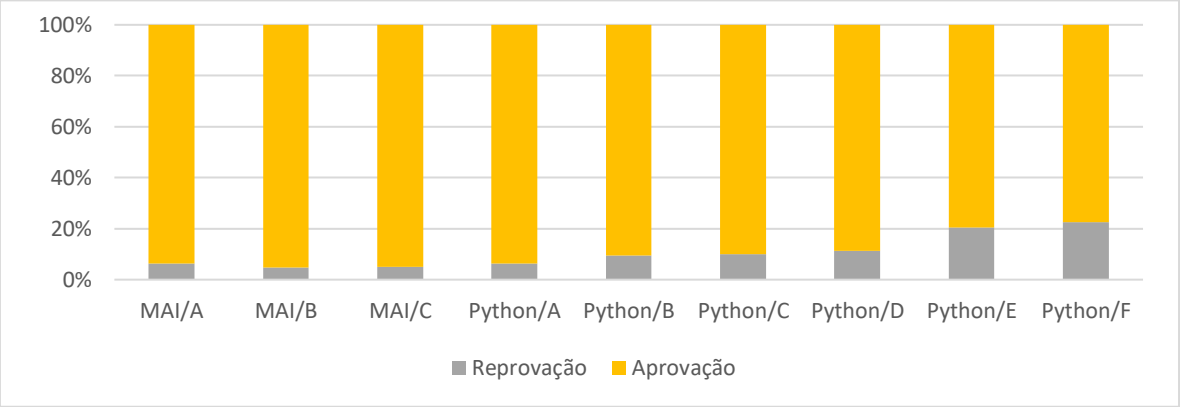

Figura 11 - Representação gráfica da taxa de aprovação/reprovação

Além da redução da reprovação, também foi possível perceber alguns outros aspectos positivos na utilização do App Inventor:

- Os estudantes apresentam-se mais motivados a desenvolver aplicativos para smartphone do que programas console;
- Linguagens de blocos são mais fáceis de assimilar do que linguagens textuais;
- Os recursos de smartphones como sensores, câmeras, recursos de áudio e vídeo, acesso a lista de contatos e ligações telefônicas, possibilitam diversas aplicações que inspiram nos estudantes o conceito de *computational thinking*, o que facilita a criação de exemplos, trabalhos práticos e o envolvimento dos estudantes;
- Alguns estudantes se interessaram pela disciplina seguinte de algoritmos, denominada Algoritmos e Estruturas de Dados II, optativa para a turma, que é apresentada usando a linguagem C++. O processo de migração do App Inventor para a linguagem C++ foi apresentado em duas aulas com ótimo aproveitamento.

6. Conclusões

Disciplinas de algoritmos e programação normalmente representam obstáculos para os estudantes novatos nessa área. O problema se agrava em cursos que não tem computação como atividade fim. Em geral, os estudantes não possuem motivação suficiente para se dedicar à disciplina e são exigidos em um conteúdo totalmente novo para eles.

Este artigo apresentou um relato da utilização do MIT App Inventor em uma disciplina de algoritmos e programação, mantendo-se o ementário da disciplina e, ainda, com os devidos subsídios para que o aluno continue o estudo dos algoritmos em linguagens textuais.

Os resultados do trabalho demonstram que a metodologia possibilita maior motivação, facilita o entendimento, melhora o mapeamento dos problemas para soluções computacionais (*computational thinking*) e tem como resultados produtos reais criados a partir da construção do conhecimento (construcionismo). Os estudantes tiveram melhores índices de aproveitamento e avaliaram de forma positiva a metodologia utilizada.

Referências

Gomes, T. C., MELO, J. C. (2012). App Inventor for Android: Uma proposta construcionista para experiências significativas de aprendizagem no ensino de programação. *Anais do IV Simpósio Hipertexto e Tecnologias na Educação: Redes Sociais e Aprendizagem*. Recife.

Karakus, M., Uludag, S., Guler, E., Turner, S. W., Ugur, A. (2012). Teaching computing and programming fundamentals via App Inventor for Android. In *2012 International Conference on Information Technology Based Higher Education and Training (ITHET)* (pp. 1-8). IEEE.

Lu, J. J., Fletcher, G. H. (2009). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 260-264).

Mauch, E. (2001). Using technological innovation to improve the problem-solving skills of middle school students: Educators' experiences with the LEGO mindstorms robotic invention system. *The Clearing House*, 74(4), 211-213.

Papert, S. (1994). A máquina das crianças. Porto Alegre: Artmed, 17.

Pasternak, E., Fenichel, R., Marshall, A. N. (2017). Tips for creating a block language with blockly. In *2017 IEEE Blocks and Beyond Workshop (B&B)* (pp. 21-24). IEEE.

Pereira, F. G. H. S., de Araújo, G. S., Cheung, L. M., de Araújo, A. V., Zunta, H. B. (2020). Relato da utilização da plataforma App Inventor como ferramenta de ensino de lógica de programação para professores da Rede Básica de Ensino. In *Anais do XXVIII Workshop sobre Educação em Computação* (pp. 86-90). SBC.

Ramos, N., Freitas, C., Avila, S., Costa, P., Testoni, V., Borin, J. (2015). Ensino de programação para alunas de ensino médio: Relato de uma experiência. In *Anais do XXIII Workshop sobre Educação em Computação* (pp. 386-395). SBC.

Ribeiro, J. P., Manso, M. A., Borges, M. (2016). Dinâmicas com App Inventor no Apoio ao Aprendizado e no Ensino de Programação. In *Anais do Workshop de Informática na Escola* (Vol. 22, No. 1, p. 271).

Roque, R. V. (2007). *OpenBlocks: an extendable framework for graphical block programming systems* (Doctoral dissertation, Massachusetts Institute of Technology).

Saigal, A. K., Saigal, A. (2011). Saathimobile and the rapid deployment of prototypes to build applications for social enterprise in the developing world. In *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief* (pp. 351-356).

Soares, A. (2014). Reflections on teaching App Inventor for non-beginner programmers: Issues, challenges and opportunities. *Information Systems Education Journal*, 12(4).

Tempel, M. (2012). Logo: A language for all ages. Comput. *Sci. K–8 Build. a Strong Found*, 16-17.

Wolz, U., Leitner, H. H., Malan, D. J., Maloney, J. (2009). Starting with scratch in CS 1. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 2-3).