Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUCAS DE MATOS PRATES

A FERRAMENTA BLOCKLY COMO APOIO AO PROCESSO DE ENSINO DE ALGORITMOS

Assis/SP
2020

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUCAS DE MATOS PRATES

A FERRAMENTA BLOCKLY COMO APOIO AO PROCESSO DE ENSINO DE ALGORITMOS

Projeto de pesquisa apresentado ao curso de Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundaçã o Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Lucas de Matos Prates
Orientador(a): Dr. Luiz Ricardo Begosso

Assis/SP
2020

FICHA CATALOGRÁFICA

P912f PRATES, Lucas de Matos
A ferramenta blockly como apoio ao processo de ensino de algoritmos / Lucas de Matos Prates. – Assis, 2020.

51p.

Trabalho de conclusão do curso (Ciência da Computação). – Fundação Educacional do Município de Assis-FEMA

Orientador: Dr. Luiz Ricardo Begosso

1.Algoritmos 2.Blockly 3.Ensino

DD005.131

A FERRAMENTA BLOCKLY COMO APOIO AO PROCESSO DE ENSINO DE ALGORITMOS

LUCAS DE MATOS PRATES

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Dr. Luiz Ricardo Begosso

Examinador: Dr. Almir Rogério Camolesi

Assis/SP
2020

RESUMO

A computação é uma área de conhecimento cada vez mais almejada, porém, apesar do grande índice de pessoas que ingressam nos cursos, o índice de desistência também é alto. Vários fatores já foram apontados como principais causas desse problema, e um deles é a dificuldade de aprendizagem de algoritmos, muitos alunos acham difícil e não se sentem motivados a aprender a disciplina. Diante disso, diversas ferramentas de ensino foram criadas para auxiliar no ensino da matéria. O presente trabalho busca então, apresentar e explorar a ferramenta Blockly, que está em desenvolvimento pela Google e é uma forma de incentivar os alunos no processo de aprendizagem, escrevendo códigos de programação simples em forma de blocos, algo que pode prepará-los para realizar a programação em uma linguagem concreta e os motivar a não desistirem, diminuindo assim a evasão nos cursos e o déficit de profissionais na área.

Palavras-chave: Algoritmos; Ensino; Blockly; Google; Computação

ABSTRACT

Computing is an area of knowledge that is increasingly sought after, however, despite the large number of people entering courses, the dropout rate is also high. Several factors have already been identified as the main causes of this problem, and one of them is the difficulty of learning algorithms, many students find it difficult and do not feel motivated to learn the discipline. Therefore, several teaching tools were created to assist in teaching the subject. This article then seeks to present and explore the Blockly tool, which is under development by Google and is a way to encourage students in the learning process, writing simple programming codes in the form of blocks, something that can prepare them to do programming in a concrete language and motivating them not to give up, thus reducing dropout rates and the shortage of professionals in the area.

Keywords: Algorithm; Teaching; Blockly; Google; Computation

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1.INTRODUÇÃO

A computação é uma área de conhecimento que muitas pessoas possuem interesse para seguir sua carreira profissional. No entanto, durante o percurso de aprendizagem dificuldades são encontradas, como o ensino de algoritmos, que antecede o desenvolvimento de software.

Como alternativas para facilitar a aprendizagem de algoritmos, muitas ferramentas de apoio foram criadas. Uma das mais recentes é a ferramenta Blockly, desenvolvida pela Google. Segundo Pasternak, Fenichel e Marshall (2017) o Blockly é uma biblioteca de código aberto desenvolvido em JavaScript para fazer algoritmos baseados em blocos, a ferramenta dispõe de uma interface de usuário para editar os blocos e um framework para transformar os algoritmos em forma de blocos para alguma linguagem de programação. O Blockly foi inicialmente lançado em 2012 e desde 2017 permanece em desenvolvimento ativo.

No processo de aprendizagem, várias técnicas para o ensino de algoritmos podem ser usadas, mas mesmo assim a matéria continua sendo uma das principais causas da evasão dos alunos nos cursos relacionados a computação. Conforme evidenciado por Hoed e Ladeira (2016), a evasão é estimulada pelas dificuldades enfrentadas no estudo de algoritmos, em contrapartida a aprovação nessa disciplina aumenta as chances do aluno não desistir. Vários motivos foram identificados do motivo de tanta dificuldade nessa ciência, como complexidade, técnicas falhas, falta de motivação e raciocínio lógico dos alunos, porém, não se pode afirmar se é por alguma dessas razões especificadamente ou o todo.

De acordo com Borges (2000), a forma como é conduzido o ensino de algoritmos não motiva boa parte dos alunos, pois muitos deles não entendem a importância da disciplina e não se sentem interessados, ainda mais se não tiverem o mínimo de conhecimento básico sobre computação. Além disso, nos primeiros programas desenvolvidos, geralmente não há uma interface gráfica e os exemplos utilizados são na maioria operações matemáticas que contribuem para dificultar a aprendizagem dos alunos.

Fraser (2015) relata que a programação voltada a blocos é um bom ponto de partida se possuir um bom planejamento antes de começar a ensinar, obviamente não deve ser utilizada o tempo todo porque ninguém irá aprender a real programação só construindo

códigos de blocos, porém é algo viciante aos alunos iniciantes e que os prepara para realizar códigos mais difíceis.

Em seu relato de experiência usando a ferramenta Scratch com calouros da computação, Cristovão (2008) conseguiu ótimas evidências de que o uso desse tipo de recurso motiva os alunos a aprenderem, pois o resultado dos códigos escritos por eles era visível imediatamente, fazendo-os aprender estruturas de algoritmos mais rapidamente e passar mais tempo programando. Outro grande ponto deduzido foi que o índice de desistência após o uso da ferramenta diminuiu, e ao programar em uma linguagem de programação o aluno tinha muito mais facilidade.

1.1 OBJETIVOS

O presente trabalho tem por objetivo explorar uma das ferramentas mais recentes desenvolvida pela Google, o Blockly, identificar seus recursos e como ela pode ser um meio para auxiliar a diminuir a dificuldade dos alunos na área da computação ao desenvolver algoritmos, com isso contribuindo para reduzir a taxa de evasão nos cursos de computação, já que a matéria é uma das principais causas desse problema, e motivando os alunos a se interessarem pela programação propriamente dita, preparando-os para escrever códigos mais difíceis em uma linguagem de programação concreta.

1.2 JUSTIFICATIVA

Com base em pesquisas realizadas na área, a dificuldade no aprendizado da matéria de algoritmos é uma das razões para a evasão dos alunos nos cursos de computação em geral. Tendo isto em mente, este estudo pode auxiliar tanto professores, quanto a comunidade da computação propriamente dita a elaborar ferramentas como a que será abordada neste trabalho ou então estratégias melhores para o ensino de algoritmos nos cursos de computação, visando assim facilitar a aprendizagem dos alunos e consequentemente eliminar uma das dificuldades enfrentadas por eles na vida acadêmica e por fim diminuir a evasão, melhorando assim o déficit de profissionais no mercado de trabalho.

1.3 MOTIVAÇÕES

As principais motivações encontradas para realizar este estudo, foram a observação própria principalmente. Nos meus três primeiros anos de vida acadêmica até a realização deste trabalho, pude observar vários colegas de sala tendo grande dificuldade na matéria de algoritmos e estrutura de dados. Muitos chegam nos cursos de computação e acabam se desiludindo após terem dificuldades nas matérias dos anos iniciais, como a dos algoritmos e acabam desistindo. Não se sabe ao certo o motivo de tanta dificuldade no entendimento da matéria, se é a sua complexidade, falta de raciocínio lógico dos alunos ou os métodos de ensino dos professores que dificultam o aprendizado dos alunos. Depois do meu relato de experiência, decidi pesquisar sobre o assunto e acabei encontrando um estudo de Paixão, Fortaleza e Conte (2013) em que eles demonstram a relação entre o perfil psicológico dos alunos e a evasão nos cursos de computação, acabei me interessando e decidi fazer algo para contribuir para alterar esta realidade, este estudo. Observa-se a grande evasão de alunos segundo a SBC (Sociedade Brasileira de Computação) durante os anos de 2016 e 2017, conforme apresentado na Tabela 1.

Evolução dos Números de Ingressantes e de Concluintes por Curso, de 2016 para 2017

Modalidade de Cursos	2016		2017		Evolução (%)	
	Ingressantes	Concluintes	Ingressantes	Concluintes	Ingressantes	Concluintes
Ciência da Computação	22643	6470	22444	6161	-0.88	-4.78
Engenharia de Computação	11707	2114	10680	2267	-8.77	7.24
Engenharia de Software	1518	144	2087	232	37.48	61.11
Licenciatura em Computação	2690	1127	5074	1081	88.62	-4.08
Outros Cursos	1900	484	1299	480	-31.63	-0.83
Sistemas de Informação	25990	10286	25698	9151	-1.12	-11.03
Cursos de Tecnologia (Todos)	66663	21387	76224	20606	14.34	-3.65
Total	133111	42012	143506	39978	7.81	-4.84

Tabela 1 – Relação entre ingressantes e concluintes nos anos de 2016 e 2017, segundo a SBC (Sociedade Brasileira de Computação), disponível em: https://www.sbc.org.br/documentos-da-sbc/summary/133-estatisticas/1200-pdf-png-educacao-superior-em-computacao-estatisticas-2017

1.4 CONTRIBUIÇÃO

Este trabalho pretende contribuir para o processo de ensino de algoritmos, aprofundando os estudos sobre uma ferramenta de programação em blocos, e ao final dos estudos, será desenvolvido um game utilizando a ferramenta Blockly, que terá o objetivo de motivar os alunos a aprenderem conceitos básicos de programação de forma lúdica. O game a ser desenvolvido será chamado de Blocks and Codes.

Pretende-se desenvolver este trabalho começando com um estudo acerca das dificuldades que os alunos iniciantes de computação encontram na disciplina de algoritmos e apresentar experiências e estudos realizados no seu ensino. Logo após, uma apresentação geral da ferramenta Blockly será feita com o intuito de demonstrar como ela funciona, juntamente com experiências que outras pessoas já tiveram com o seu uso. Na reta final deste estudo, será desenvolvido um game utilizando o Blockly com o intuito de apoiar o processo de aprendizagem dos alunos iniciantes da Ciência da Computação. Finalmente, o jogo desenvolvido e a ferramenta Blockly serão aplicados em uma sala de aula real e dados sobre a percepção que os estudantes tiveram serão levantados para demonstrar os resultados e conclusões.

1.5 ESTRUTURA DO TRABALHO

Este trabalho de conclusão de curso está organizado na seguinte forma: o Capítulo 1 apresenta a Introdução; no Capítulo 2 descreveremos os resultados do levantamento bibliográfico a respeito das dificuldades que os alunos apresentam para o aprendizado de algoritmos; o Capítulo 3 descreve a ferramenta Blockly; o Capítulo 4 apresentará exemplos de experiências conduzidas com a ferramenta Blockly e outras do tipo; o Capítulo 5 apresentará os detalhes do desenvolvimento da ferramenta proposta por este projeto; por fim, o Capítulo 6 descreverá os resultados obtidos com o desenvolvimento da ferramenta e as conclusões.

2. ESTUDO SOBRE AS DIFICULDADES NO APRENDIZADO DE ALGORITMOS

O ensino de algoritmos nos dias de hoje segue uma maneira tradicional de ensino, geralmente os professores apresentam a teoria em si, aplicam exemplos práticos, depois exercícios práticos e, por fim, a proposição de projetos mais complexos, como Borges (2000) ilustra na Figura 2.

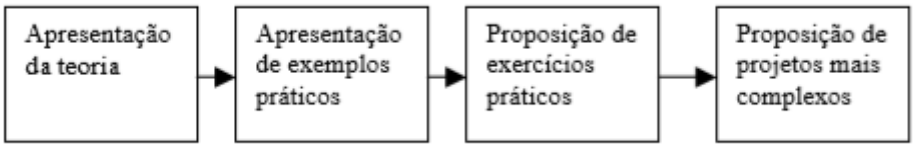

Figura 1: A sequência básica de como algoritmos são ensinados (In: BORGES, 2000, p.2)

Essa sequência ainda é muito utilizada para o ensino de algoritmos nos dias de hoje, mas, isso não está se mostrando muito eficiente na maioria das vezes, justamente por uma questão de que muitos alunos chegam nos cursos voltados a computação e não sabem que a matéria de algoritmos é uma das mais importantes para sua formação, é uma disciplina que é a base para qualquer um que deseja seguir nessa área de atuação.

Conforme demonstrado na pesquisa de Raabe e Silva (2005), os problemas na aprendizagem de algoritmos podem ser divididos em três principais naturezas: Natureza didática, Natureza cognitiva e de Natureza afetiva. Os de natureza didática são as dificuldades no estudo realmente em si de da disciplina, podem ser relacionadas tanto ao professor, quanto ao aluno. Os de natureza cognitiva são os problemas que cada aluno pode ter individualmente, ou seja, características pessoais de cada um. Por fim, os de natureza afetiva, que são obstáculos de caráter pessoal também, mas não relacionados a aprendizagem e sim coisas como acontecimentos na vida de cada pessoa. Estes três tipos de problemas serão descritos com maiores detalhes nas próximas seções.

2.1 PROBLEMAS DE NATUREZA DIDÁTICA

Os alunos não conseguem se interessar muito pela disciplina, porque, na grande maioria das vezes o professor não utiliza de uma linguagem concreta ou algo do tipo, e sim o famoso algoritmo em forma de português, representando as estruturas básicas de código em uma linguagem que não existe de verdade e que o aluno não poderá aplicar em um projeto inicial em um ambiente de desenvolvimento ou algo do tipo.

Quando sentem dificuldades, os estudantes recorrem aos professores e os mesmos muitas vezes também não conseguem acompanhar o pensamento, a lógica dos alunos, acabando assim por não conseguir entender o fluxo em que o aluno está raciocinando e consequentemente pode acabar por confundí-lo ainda mais caso faça alguma explicação de dúvida de forma não coerente ao seu raciocínio lógico.

Ainda que o aluno consiga superar essa parte dos conceitos na teoria e com uma linguagem não concreta, outros problemas podem surgir quando o desenvolvimento da parte prática começa. Os professores de algoritmos vão com seus estudantes para o laboratório de informática e começam a ensinar a disciplina de uma maneira em que o professor faz o código e os alunos vão copiando e tentando entender, muitas vezes por si mesmos. Isso pode ser extremamente desanimador para os alunos, porque, aqueles que não possuírem alguma prática anterior ou que tenham mais dificuldades do que outros, podem ficar para trás, ainda mais se o professor não souber controlar o ritmo em que está programando.

Como se não bastasse todos esses problemas iniciais, a maioria das vezes em que os alunos vão programar em uma linguagem concreta, geralmente a mesma não tem uma interface visual. Qualquer pessoa que utiliza um computador, está acostumada a ver resultados em suas telas a partir de cliques em botões, arrastamento de telas, rolagem do mouse e coisas do tipo. Quando os alunos se deparam com uma tela preta de console, e que exibe de uma forma não muito agradável aos olhos o que eles programaram, sentem um sentimento de inutilidade, já que seu código, a maioria das vezes não fez nada demais justamente por ser um simples problema matemático por exemplo.

Após todo esse fluxo de aprendizado, finalmente chega a hora em que o aluno tem que ser avaliado formalmente. Muitas das vezes, o aluno tem uma sensação de que realmente está aprendendo, caso consiga acompanhar o professor e resolver algoritmos com o seu apoio, realizar todas as tarefas, e fazer pequenos códigos em uma linguagem não estruturada ou

até mesmo no papel, mas quando chega a hora de uma avaliação, pode ser o momento em que o estudante tem uma auto reflexão sobre si.

Vários alunos ficam simplesmente travados no momento da prova, não sabem nem por onde começar o código e como devem realizá-lo, e aliado a isso um tempo limitado de prova e a observação de outras pessoas aparentemente conseguindo resolver os problemas do exame podem deixar o aluno com um nervosismo, que o impede de desenvolver um raciocínio lógico para resolver o teste, e é nesse momento em que o estudante percebe que ele somente acompanhou, mas não absorveu o conhecimento necessário, e sem um apoio do professor ou consulta na internet por exemplo, não consegue realizar os algoritmos da prova sozinho.

Um dos maiores problemas evidentes no ensino de algoritmos é simplesmente a individualidade de cada aluno. Cada estudante pode ter mais facilidade ou dificuldade em entender a lógica por trás dos algoritmos, e justamente por isso o modelo tradicional de ensino é utilizado. Criando assim um contexto parecido com o da seleção natural, onde aqueles que conseguirem acompanhar melhor o professor, e tem mais facilidade em lógica, entendendo todo o fluxo de um algoritmo são selecionados e os que não conseguirem, reprovam ou desanimam completamente do curso e acabam desistindo do mesmo.

Aliado ao problema anterior, vem o caso em que no início dos cursos, geralmente se tem uma grande quantidade de alunos. Isso impacta na capacidade do professor atender as dificuldades de cada um, já que se o tempo todo surgir dúvidas, o desenvolvimento da aula não será satisfatório. Juntamente isso, as pessoas são estranhas umas das outras, o que contribui para o fato de que algumas vezes nem os próprios alunos ajudam entre si com as dificuldades um dos outros.

Por fim, uma parte dos alunos também ingressam nos cursos de computação com uma visão totalmente equivocada sobre o que é a computação e como ela funciona, e simplesmente ficam desanimados quando uma matéria como a de algoritmos é apresentada, criando um descaso e falta de interesse que cria no aluno de não querer se dedicar ao menos para aprender e compreender o motivo dessa matéria ser lecionada, resultando a maioria das vezes numa dificuldade muito maior de fazer o estudante se apegar ao curso, fazendo-o desistir rapidamente, geralmente no primeiro e segundo semestre, onde a matéria de algoritmos é muito mais abordada.

2.2 PROBLEMAS DE NATUREZA COGNITIVA

Os problemas de natureza cognitiva dependem de como o aluno foi formado até o momento de cursar uma faculdade na área de computação, e influenciam diretamente na forma em que o aluno pensa e desenvolve um raciocínio para resolução de problemas, e gera ou não uma determinação de aprender.

Um dos maiores problemas evidenciados por Raabe e Silva (2005) é a falta de perfil do aluno para resolução de problemas. Muitos alunos não possuem a determinação de resolver problemas, justamente por virem, a maioria das vezes de escolas que não desenvolveram essa capacidade corretamente ou que não conseguiram fazer o estudante ser disciplinado o suficiente. Sabe-se que essa é a realidade de grande maioria dos colégios brasileiros.

Por virem com muitas vezes com esse perfil não muito adequado, quando se deparam com a dificuldade lógica e desafiadora da resolução de problemas que a matéria de algoritmo propõe, sentem muitas dificuldades para aprender.

Aliado a isso, muitos estudantes não tem uma base de conhecimento operatório formal, ou seja, a capacidade de se fazer deduções lógicas sem ter o auxílio de algo já concreto, não conseguem criar conceitos e ideias com facilidade.

Essa base operatória e extremamente importante para os alunos, não só na computação mas em qualquer curso superior, já que é o princípio para a compreensão lógica, sem ela, as pessoas não conseguem imaginar hipóteses, caminhos alternativos para a solução de problemas, impactando diretamente na sua capacidade de aprender mais facilmente ou não uma matéria como a de algoritmos.

Como última dificuldade cognitiva, Raabe e Silva (2005) apresentam em sua pesquisa a falta de conteúdo sem proximidade com o conteúdo escolar, já que uma matéria de algoritmos é algo que os estudante novatos de computação nunca tinham visto anteriormente, não conseguindo relacionar a disciplina com aquilo que já foram acostumados a aprender em sua vida antes de ingressar em algum curso de computação.

2.3 PROBLEMAS DE NATUREZA AFETIVA

Por último, mas não menos importante, Raabe e Silva (2005) comentam os problemas de natureza afetiva, esses são muito difíceis de se ter um total controle pois vai da realidade pessoal de cada aluno, o professor não tem como saber o que cada um passa em suas vidas.

Os problemas podem ser ocasionais, ou seja, aqueles em que acontecimentos pessoais na vida particular dos alunos influenciam diretamente na sua motivação, concentração e desempenho na hora de se dedicar aos estudos, como o fim de um relacionamento, uma família problemática ou com problemas financeiros, e etc.

Outra subdivisão dos problemas de natureza afetiva, são aqueles chamados de complicações constantes, sendo esses aqueles que se manifestam durante os estudos, que podem facilmente desmotivar mais e mais a medida que o tempo vai passando, como por exemplo: o medo de reprovar na disciplina, não estar de acordo em como o professor ensina o conteúdo, falta de autoestima e desejo de aprender, entre outros.

3.0 A FERRAMENTA BLOCKLY

Segundo a overview disponível no site oficial do Blockly: https://developers.google.com/blockly, o Blockly consiste de uma biblioteca que adiciona código visual para aplicações web e mobile. Ele utiliza de blocos que se interligam entre si para representar conceitos de código como variáveis, expressões lógicas, loops e etc.

A ferramenta Blockly permite aos usuários aplicarem princípios de programação, mesmo sem saberem nenhuma linguagem de programação, e tudo isso sem se preocupar com sintaxe e um cursor piscando em uma linha de comando.

O Blockly, inclui tudo que o usuário precisa para definir e renderizar os blocos em um editor de clique e arraste. Cada bloco, possui um código que pode ser facilmente “traduzido” em código de uma linguagem de programação real, além disso, os blocos são totalmente customizáveis, podendo ser adicionados novos comportamentos a eles.

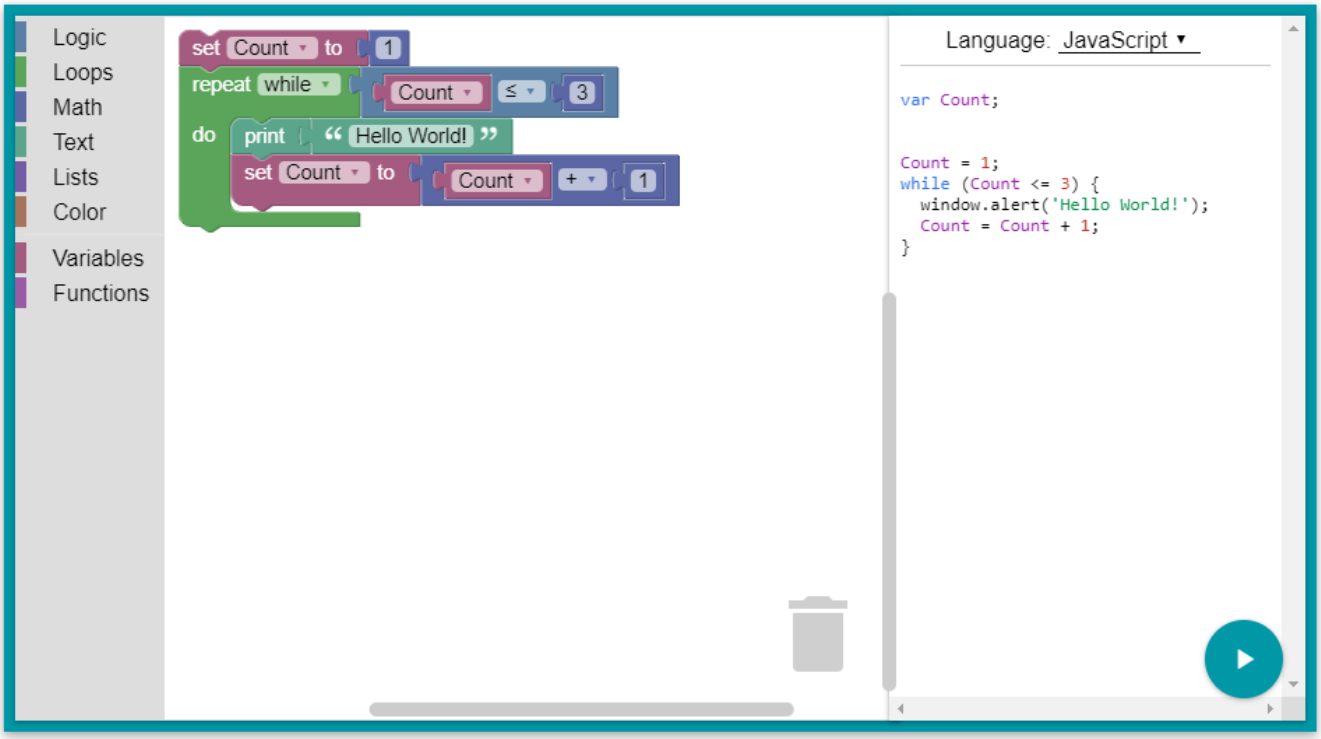

Figura 2: Um exemplo de aplicação Blockly, disponível em: https://developers.google.com/blockly.

Conforme visto na Figura 2, o Blockly possui em sua interface um menu na lateral esquerda, onde os blocos podem ser encontrados, e são separados dependendo de sua funcionalidade, sendo elas os Loops, funções matemáticas, textos, listas, cores, variáveis e funções.

Nessa interface, pode-se encontrar também, uma lixeira do lado direito inferior, ela serve para arrastar os blocos que se deseja excluir e não deixar a tela “poluída”. Do lado direito da aplicação, fica o código traduzido para uma linguagem de programação real, podendo ser JavaScript, Python, PHP, Lua e Dart. Por fim, o botão azul do lado inferior direito serve para executar o código feito na plataforma do Blockly.

O resultado da execução do código da Figura 3, exibe a mensagem “Hello World”, três vezes seguidas, conforme mostra a Figura 3.

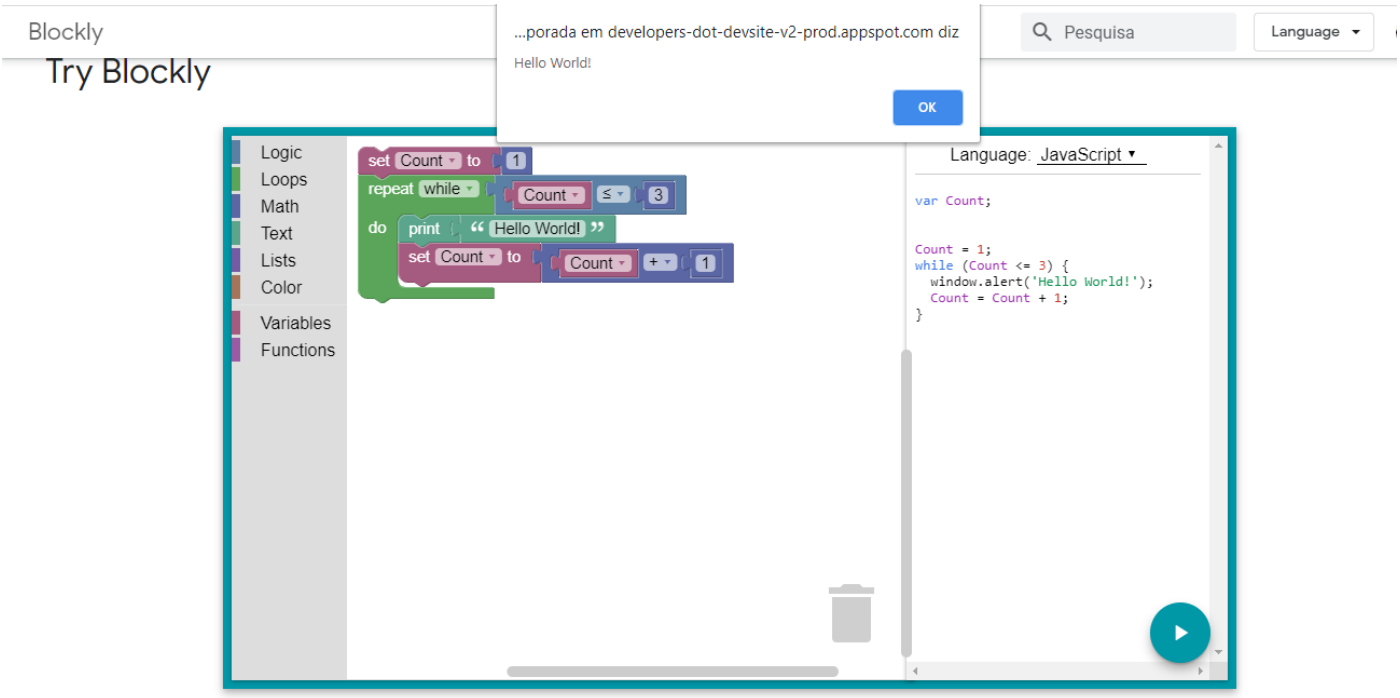

Figura 3: O resultado do exemplo da Figura 3.

3.1 TIPOS DE BLOCOS DISPONÍVEIS ORIGINALMENTE

Conforme descrito anteriormente, o Blockly possui vários tipos de blocos que podem ser utilizados para executar diferentes funcionalidades. Eles são totalmente personalizáveis, podendo ter o seu comportamento alterado da forma como o usuário preferir, além disso, pode-se criar blocos próprios para executar funções específicas, porém nesta pesquisa será apresentado alguns exemplos dos que são disponíveis originalmente pela ferramenta. A descrição dos blocos será feita na ordem top-down, ou seja, de cima para baixo.

Os blocos disponíveis na parte lógica são listados na Figura 4:

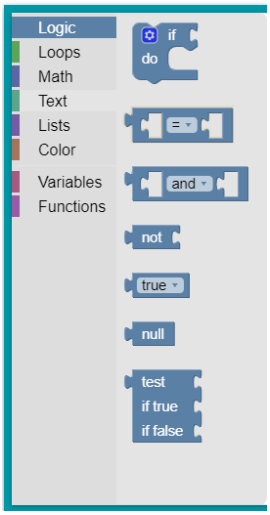

Figura 4: Blocos disponíveis na parte lógica.

Bloco lógico 1 – é o responsável pela lógica de condição, em sua engrenagem azul no lado superior esquerdo, pode-se alterar o IF para um ELSE, ou um ELSE IF.

Bloco lógico 2 – estabelece as condições entre duas variáveis numéricas, podendo ser maior (>), menor (<), maior ou igual (≥), menor ou igual (≤), igual (=) e diferente (≠).

Bloco lógico 3 – estabelece as condições entre duas variáveis booleanas, podendo ser um E (AND), ou um OU (OR).

Bloco lógico 4 – serve para fazer a negação de algum retorno booleano.

Bloco lógico 5 – bloco com o valor booleano esperado, podendo ser true ou false.

Bloco lógico 6 – bloco com o valor nulo.

Bloco lógico 7 – bloco com um teste embutido, se o teste for verdadeiro, executa um bloco, se for falso, executa o outro.

Os blocos disponíveis na parte de Loops são ilustrados na Figura 5:

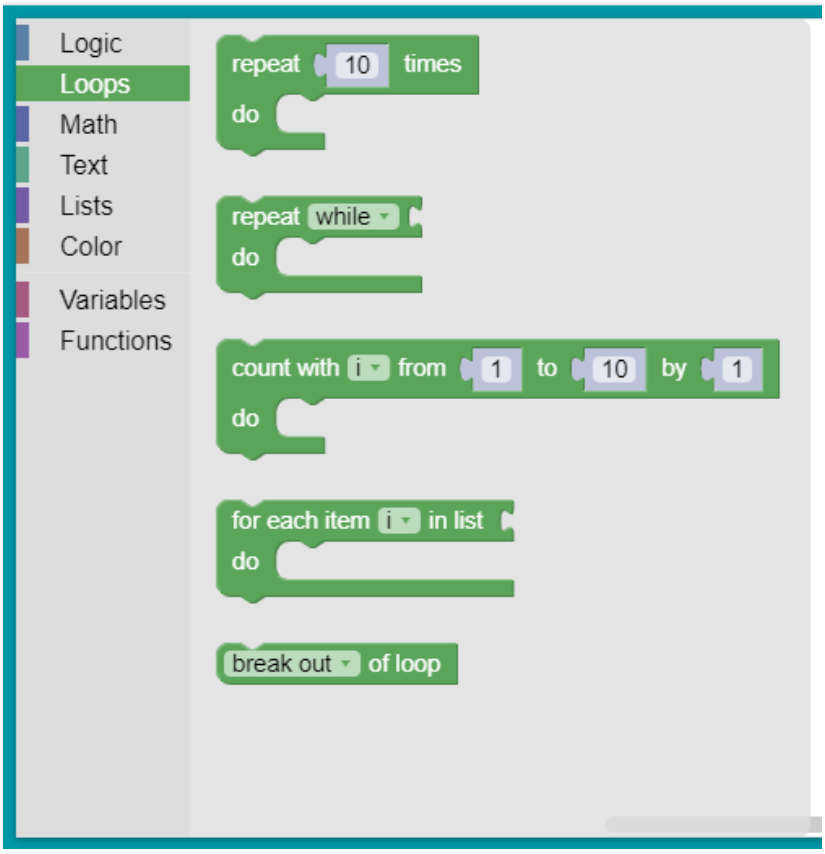

Figura 5: Blocos disponíveis na parte de loops.

Bloco de loop 1 – bloco para repetir um conjunto de blocos por um determinado período de vezes.

Bloco de loop 2 – bloco para repetir enquanto (while) ou até (until) uma condição for verdadeira.

Bloco de loop 3 – este bloco cria uma variável e começa um FOR, que incrementa um valor nessa variável toda vez que executa o código dentro dele, e não para de executar até que a variável atinja o valor final esperado.

Bloco de loop 4 – bloco FOR EACH, ou seja, para cada item em uma lista, ele executará o código dentro dele.

Bloco de loop 5 – serve para parar a execução de um loop (break out), ou pular para sua próxima execução (continue with next interation).

Os blocos disponíveis na parte matemática são ilustrados nas Figuras 6 e 7:

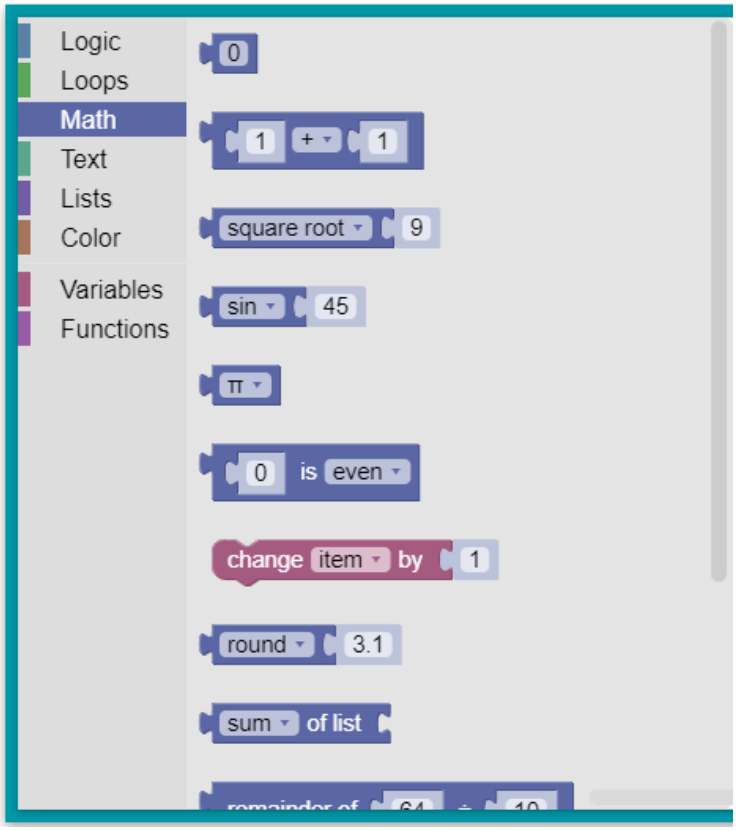

Figura 6: Blocos disponíveis na parte matemática.

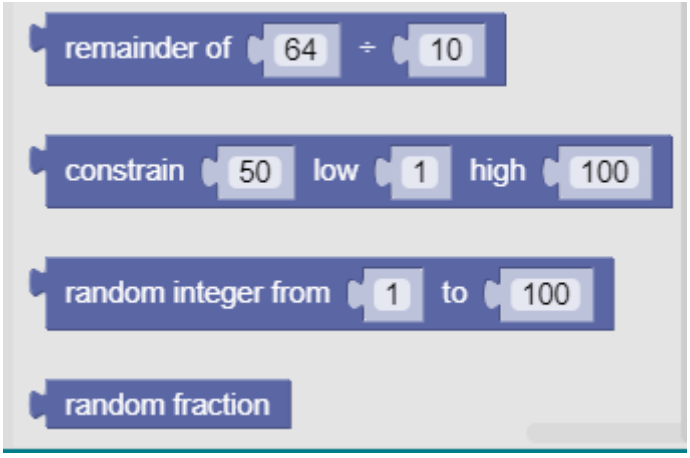

Figura 7: Continuação dos blocos disponíveis na parte matemática.

Bloco matemático 1 – representa um número.

Bloco matemático 2 – representa as operações matemáticas entre dois números, podendo ser soma (+), subtração (-), multiplicação (x), divisão (÷) e potenciação (^).

Bloco matemático 3 – representa as operações matemáticas complexas envolvendo um número, podendo ser raiz quadrada (square root), valor absoluto (absolute), número negativo (-), log (ln), exponencial (e^).

Bloco matemático 4 – retorna seno (sin), cosseno (cos), tangente (tan), arco-seno (asin), arco-cosseno (acos) e arco tangente (atan) de um ângulo.

Bloco matemático 5 – retorna constantes de valores matemáticos, por exemplo o π.

Bloco matemático 6 – retorna se um número é ímpar (odd), par (even), primo (prime), inteiro (whole), positivo (positive), negativo (negative) ou divisível por algum outro número (divisible by).

Bloco matemático 7 – adiciona um valor a uma variável numérica.

Bloco matemático 8 – arredonda um número para o inteiro mais próximo (round), retorna o menor número inteiro maior ou igual a determinado valor (round up) ou retorna o menor número inteiro dentre um número (round down).

Bloco matemático 9 – retorna a soma dos valores (sum), o menor (min), maior (max), valor médio aritmético (average), o número mediano (median), uma lista dos números mais comuns (modes), um item randômico (random item) ou o desvio padrão de uma lista (standard deviation).

Bloco matemático 10 – retorna o resto de uma divisão de dois números.

Bloco matemático 11 – define que um número deva estar em um determinado intervalo.

Bloco matemático 12 – retorna um valor inteiro aleatório de um determinado intervalo.

Bloco matemático 13 – retorna um fracionário aleatório entre 0.0 e 1.0.

Os blocos disponíveis na parte de textos são listados nas Figuras 8 e 9:

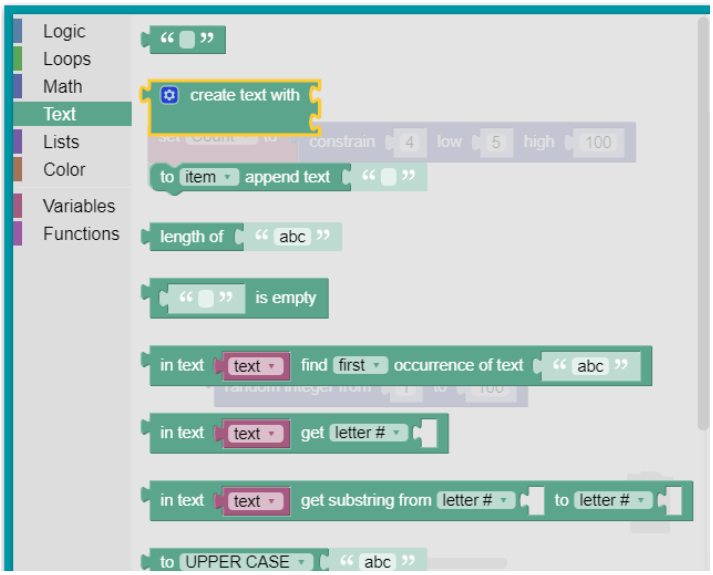

Figura 8: Blocos disponíveis na parte de textos.

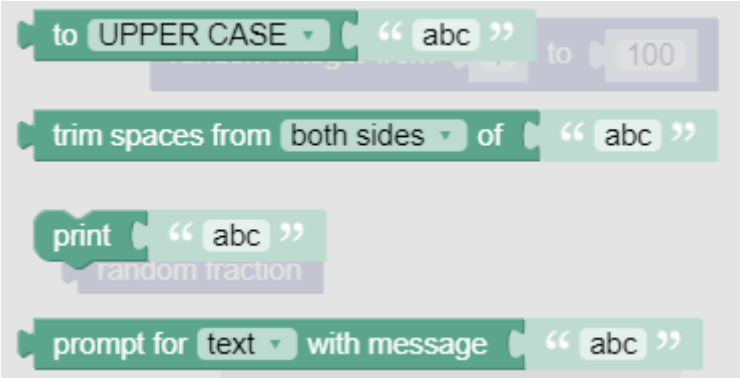

Figura 9: Continuação dos blocos disponíveis na parte de textos.

Bloco de texto 1 – define uma letra, palavra ou linha de texto.

Bloco de texto 2 – cria um pedaço de texto unindo dois ou mais itens.

Bloco de texto 3 – concatena um texto a um determinado item.

Bloco de texto 4 – retorna o número de letras, incluindo espaços, de um texto.

Bloco de texto 5 – retorna um booleano dependendo se o texto é vazio ou não.

Bloco de texto 6 – retorna o índice da primeira (first) ou última ocorrência (last) que um texto aparece em um segundo texto, caso não seja encontrado, retorna 0.

Bloco de texto 7 – retorna a letra de uma posição especifica no texto, podendo ser a primeira (letter #) a partir do inicio, a primeira a partir do final (letter # from end), a última (last letter), a primeira (first letter) ou uma letra aleatória (random letter).

Bloco de texto 8 – retorna um subtexto de um texto a partir de uma letra a outra, de uma letra em determinada posição (letter #), uma letra a partir do final (letter # from end) ou a partir da primeira letra (first).

Bloco de texto 9 – formata o texto para maiúsculo (UPPER CASE), minúsculo (lower case) ou fonte de título (title case).

Bloco de texto 10 – retorna uma cópia do texto sem espaços em brancos, podendo estes serem retirados de ambos os lados de um texto (both sides), somente do lado esquerdo (left side) ou somente do lado direito (right side).

Bloco de texto 11 – imprime um texto, número ou qualquer outro valor.

Bloco de texto 12 – mostra o prompt para o usuário digitar algum texto (text) ou número (number).

Os blocos disponíveis na parte de listas são listados nas Figuras 10 e 11:

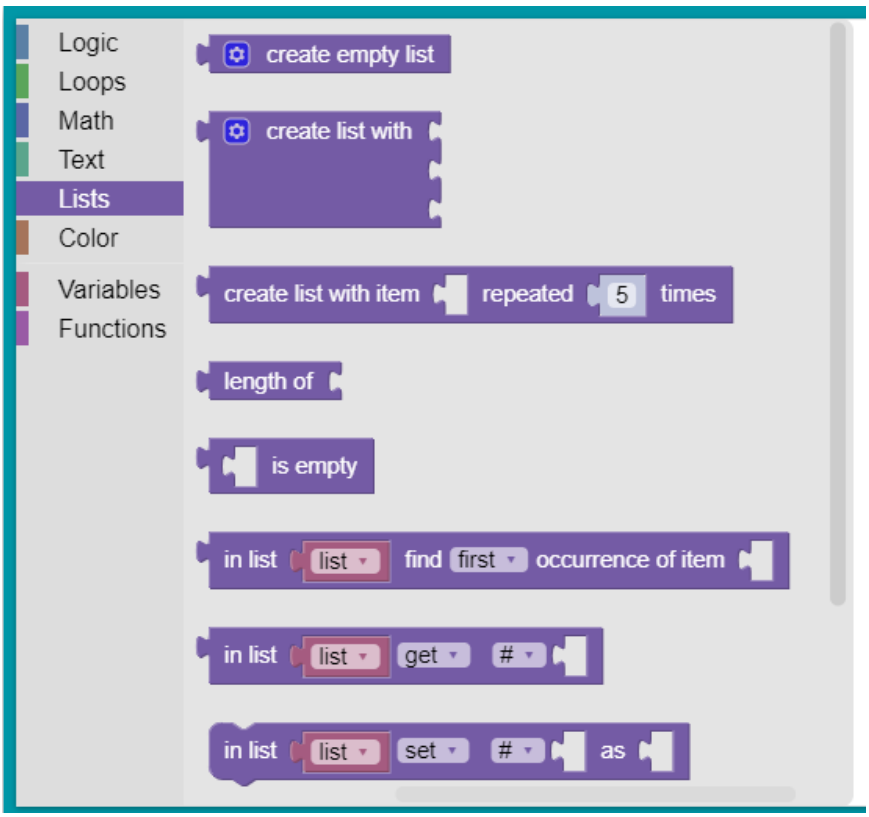

Figura 10: Blocos disponíveis na parte de listas.

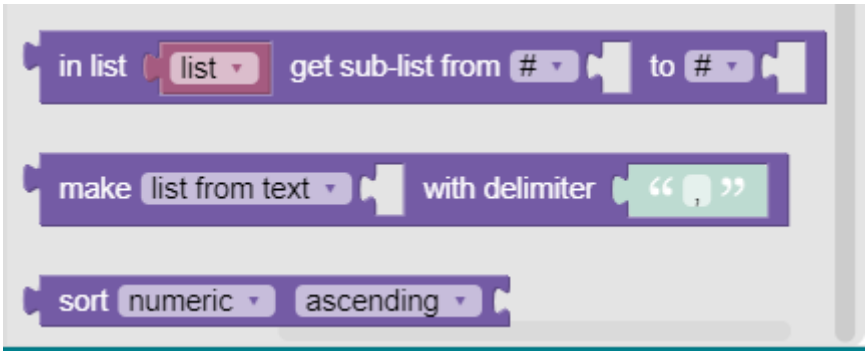

Figura 11: Continuação dos blocos disponíveis na parte de listas.

Bloco de lista 1 – cria uma lista, vazia ou com um número determinado de itens.

Bloco de lista 2 – possui função idêntica ao bloco de lista 1.

Bloco de lista 3 – cria uma lista com um determinado item repetido por um número de vezes.

Bloco de lista 4 – retorna o comprimento da lista.

Bloco de lista 5 – retorna um booleano se a lista é vazia ou não.

Bloco de lista 6 – retorna o índice da primeira (first) ou última (last) ocorrência de um item em uma lista.

Bloco de lista 7 – retorna (get), retorna e remove (get and remove) ou somente remove (remove) um item de uma lista em determinada posição.

Bloco de lista 8 – insere um novo valor (set) ou um novo item em determinada posição de uma lista (insert at).

Bloco de lista 9 – cria uma cópia de uma lista, a partir de um determinado intervalo, podendo começar ou terminar de uma posição específica (#), uma posição a partir do final (# from end), primeiro item (first), ou do item final (last).

Bloco de lista 10 – cria uma lista a partir de um texto (list from text), quebrando os itens a partir de um delimitador, ou cria um texto a partir de uma lista (text from list), separando os itens a partir de um delimitador.

Bloco de lista 11 – ordena a lista em ordem alfabética (alphabetic), ordem alfabética ignorando dependendo de uma condição (alphabetic ignore case), numérica (numeric), em ordem crescente (ascending) ou decrescente (descending).

Os blocos disponíveis na parte de cores são listados na Figura 12:

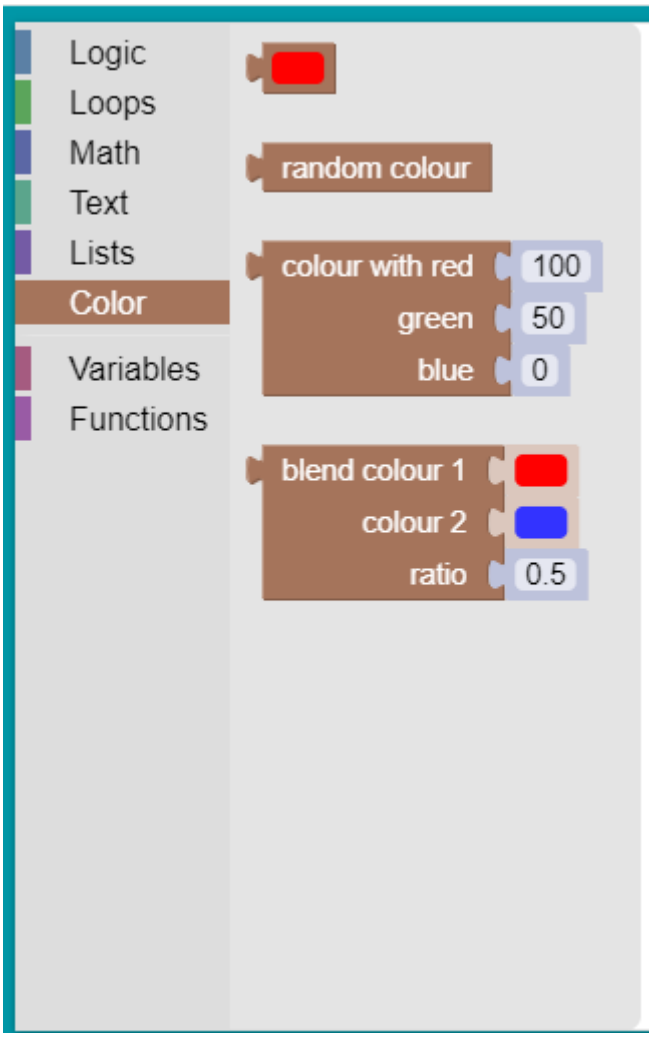

Figura 12: Blocos disponíveis na parte de cores.

Bloco de cor 1 – define uma cor a partir de uma paleta de cores.

Bloco de cor 2 – retorna uma cor aleatória.

Bloco de cor 3 – retorna uma cor a partir de um valor para vermelho, verde e azul.

Bloco de cor 4 – retorna uma cor a partir da mistura de duas cores e uma proporção.

Os blocos disponíveis na parte de variáveis são listados na Figura 13:

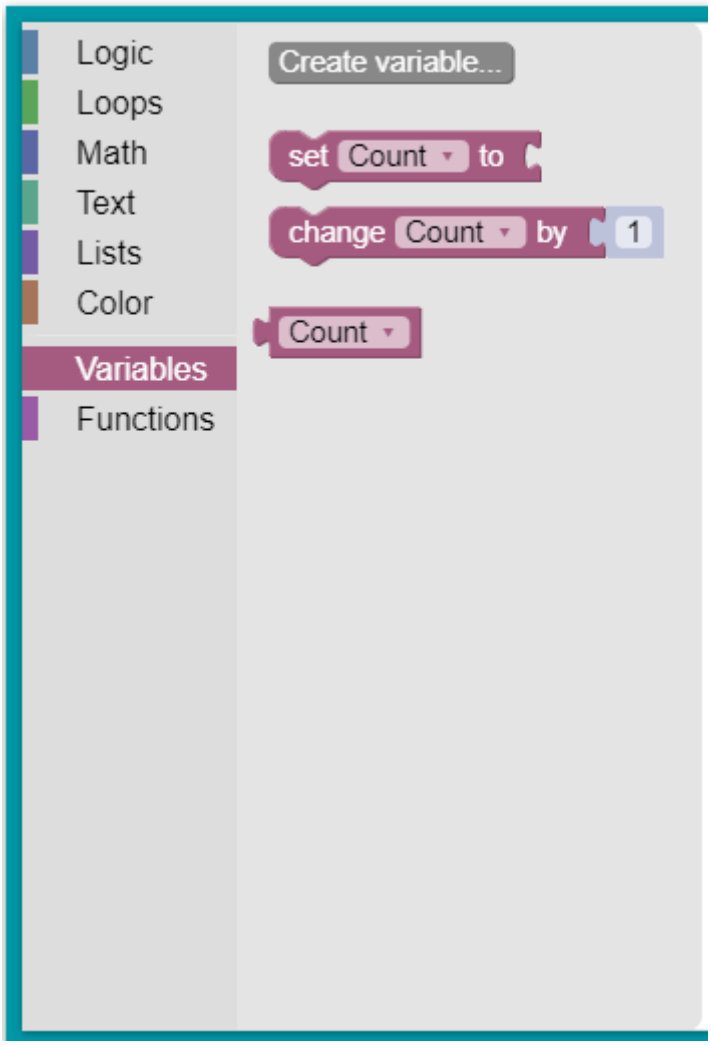

Figura 13: Blocos disponíveis na parte de variáveis.

Botão “create variable” – cria uma variável com nome escolhido pelo usuário.

Bloco de variável 1 – inicializa a variável de acordo com o valor inserido.

Bloco de variável 2 – adiciona um número a variável.

A partir do bloco 2 de variáveis, estão disponíveis as variáveis criadas, sendo a criada por padrão pela ferramenta a variável “Count”.

Os blocos disponíveis na parte de funções são listados na Figura 14:

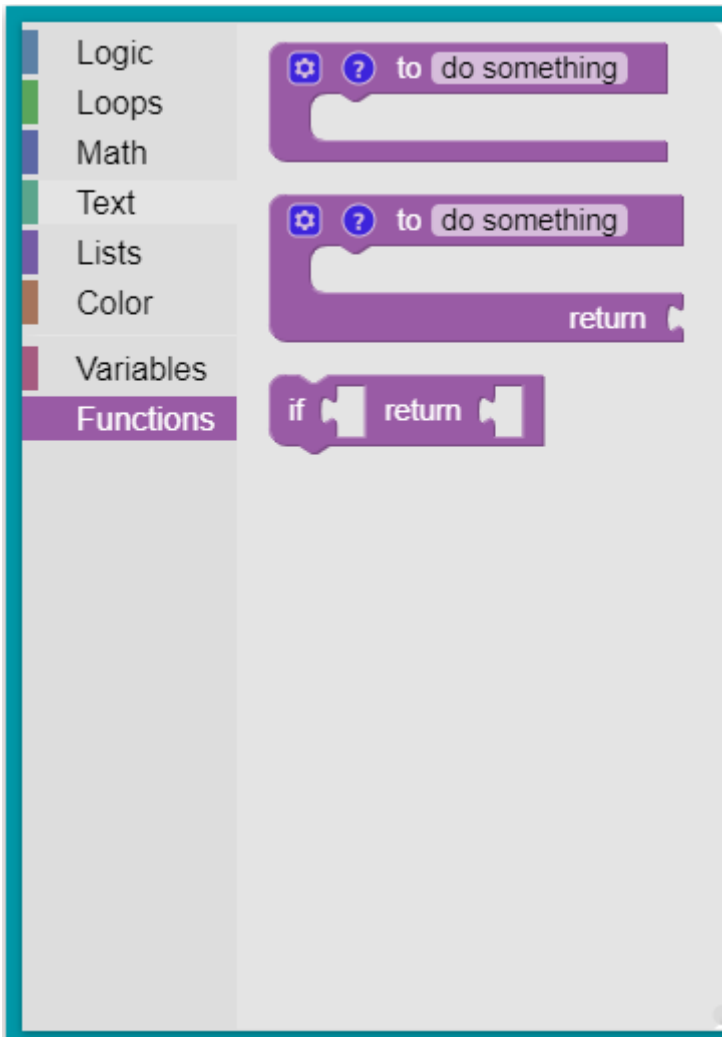

Figura 14: Blocos disponíveis na parte de funções.

Bloco de função 1 – cria uma função, no ícone de “?” define-se seu nome, e no ícone de engrenagem, define-se os parâmetros da função.

Bloco de função 2 – semelhante ao bloco de função 1, a diferença é que este bloco retorna algum valor.

Bloco de função 3 – retorna algum valor a partir de uma condição

3.2 CONSTRUINDO UM APP BLOCKLY

Para se construir um aplicativo Blockly, primeiro é necessário entender que existem duas perspectivas dessa biblioteca, a primeira é a do usuário e a segunda do desenvolvedor. Para o usuário, é uma forma simples e visual de se construir pequenos códigos e programas. Para o desenvolvedor é uma interface já pronta para se fazer uma linguagem visual que consegue exportar para uma linguagem de programação real, podendo ser até o momento, traduzido para as seguintes linguagens: Javascript, Python, PHP, Lua e Dart.

Existem três pilares para se desenvolver uma aplicação com o Blockly, e são:

1. Integrar com o editor Blocky: O editor do blockly nada mais é do que o básico para começar a construir a aplicação, consiste de uma caixa de ferramentas para armazenar os tipos de blocos, e um workspace para organizá-los.
2. Criar seus próprios blocos: Quando o usuário já estiver com sua aplicação integrada ao Blockly, deve-se criar os blocos para os seus usuários codificarem e depois adicionar a sua caixa de ferramentas.
3. Construir o resto da aplicação: O Blockly é uma maneira de gerar código, o que o usuário poderá fazer com ele depende da sua imaginação, e isso será o mais importante.

Como qualquer tecnologia, o Blockly também possui os seus pontos fortes, mas não quer dizer que é a única ferramenta disponível para desenvolver esse tipo de código. Alguns dos benefícios do Blockly serão apontados abaixo:

1. Código exportável: ou seja, os usuários podem extrair os seus programas baseados em blocos para linguagens de programação comuns.
2. Open souce: tudo sobre o Blockly é aberto, podendo ser utilizado sem problemas em qualquer projeto.
3. Extensível: o Blockly é moldável para atender as suas necessidades, pode-se criar blocos customizáveis e remover os blocos que não forem úteis.

4. Muito capaz: o Blockly não é um brinquedo, isso quer dizer que é possível desenvolver aplicações complexas com esse tipo de ferramenta.
5. Internacional: o Blockly é traduzido para mais de 40 idiomas.

4.0 EXPERIÊNCIAS COM O BLOCKLY E OUTRAS FERRAMENTAS

Neste capítulo, serão relatadas algumas experiências de outros autores que utilizaram ferramentas para ensinar algoritmos, como o Blockly, Scratch, etc. Pretende-se observar se essas experiências geraram bons resultados, se esses utensílios provaram incentivar e auxiliar no momento de aprendizagem de algoritmos.

4.1 HENRIQUE MONTEIRO CRISTOVÃO UTILIZANDO SCRATCH

Em seu relato de experiência, Cristovão (2008) começa descrevendo a maior dificuldade enfrentada nos cursos de computação, aprender algoritmos, dizendo que os professores tentam cada dia mais experimentar novas tecnologias e metodologias para o ensino dessa disciplina, principalmente porque alunos do ensino médio estão ingressando nos cursos superiores com posturas inadequadas como o medo de errar por exemplo.

Ele aponta que um dos maiores problemas na aprendizagem de algoritmos é o fato de os alunos não conseguirem feedback imediato de suas tentativas de resolver um problema como um algoritmo, por conta da falta de recursos computacionais que consigam demonstrar essa resposta a eles, considerando assim que no geral o estudante possui todo o raciocínio lógico para poder conseguir ver seus resultados somente através do código escrito por eles.

Após isso ele descreve a ferramenta Scratch, um software freeware desenvolvido pelo Lifelong Kindergarten group do Massachussets Institute of Technology (MIT) Media Lab. Cristovão (2008) ainda descreve as principais características dessa ferramenta, como o seu ambiente fácil de programação que não requer conhecimentos prévios na área para começar a usar, sendo indicado para ser utilizado por pessoas a partir dos 8 anos de idade, além de se conseguir compartilhar o código desenvolvido em um site, para ser executado no próprio browser.

Sua experiência foi realizada em dois semestres letivos em turmas de calouros do curso da Ciência da Computação na disciplina de algoritmo 1 da FAESA em Vitória no Espirito Santo. O trabalho com o Scratch durou 1 mês e meio e 2 a 3 alunos utilizavam a mesma máquina

para utilizar a ferramenta, no restante do semestre os alunos aprenderam programação utilizando a linguagem Java, sendo que antes os alunos desenvolviam algoritmos com o portugol e fluxogramas.

Cristovão criou um wiki aos alunos para os mesmos poderem compartilhar os seus códigos feitos no Scratch, além de preparar 5 esquemas para exemplificar alguns comandos e construções de algoritmos básicos aos estudantes além de alguns problemas para serem resolvidos.

Após os alunos desenvolverem e trabalharem com a ferramenta Scratch, Cristovão (2008) observou alguns resultados positivos em relação aos alunos, como por exemplo:

- A motivação: foi observada uma maior motivação nos alunos, já que a ferramenta devolvia os resultados dos códigos desenvolvidos imediatamente, aumentando a vontade dos alunos de realizar as atividades.
- Qualidade na aprendizagem: os alunos conseguiram fazer boas estruturas de código em um tempo de aprendizagem relativamente menor do que quando estavam programando utilizando o “portugol” ou fluxogramas.
- Menor impacto na transição para Java: os estudantes tiveram bem menos dificuldades para aprender em uma linguagem de programação real, como o Java.
- Rapidez no aprendizado de estruturas mais complexas: os alunos já se depararam com estruturas mais complexas logo nas primeiras aulas, fazendo assim que assimilassem mais facilmente os tipos de estruturas.
- Feedback imediato: como o Scratch consegue aceitar modificações enquanto executa, os alunos conseguiram ver os resultados de seus códigos imediatamente, não dependendo assim do professor para validar o seu código, o que era um problema já que a carga horária da disciplina dificulta quanto a isso.
- Modularização de programas: a criação de funções geralmente é abordada em um período mais tardio dos cursos, já utilizando o Scratch, os alunos conseguiram ter conhecimento desses módulos de código muito antes que o normal, sem precisar dos ensinamentos do professor.
- Facilidade com programação paralela: normalmente esse tipo de programação não é abordada nos primeiros semestres das faculdades, mas no Scratch esse tópico é facilmente explorado já que é comum em um projeto dessa ferramenta possuir

trechos de programação paralela, assim os alunos iniciantes já tinham uma base antes de ir ao Java, tendo assim menos dificuldades.

- Cooperação: com o Wiki desenvolvido pelo professor os alunos puderam compartilhar seus códigos e suas experiências entre si, gerando assim muito mais cooperação em sala de aula.
- Mais tempo programando: com a maior motivação dos alunos, eles quiseram se dedicar mais em suas tarefas, adquirindo assim maior experiência.
- Menor desistência: após a utilização do Scratch em sala de aula, foi notado uma menor desistência dos alunos.
- Valorização de todos os alunos: mesmo os alunos que tem dificuldades em sala de aula conseguiram ser valorizados por meio de criação de extensos projetos utilizando somente a criatividade individual.

Por fim, Cristovão (2008) diz que o mais interessante nessa experiência foi que os alunos já adquiriram bons conhecimentos sobre programação logo nas primeiras semanas, ficando assim motivados e os incentivando a programar, fazendo-os obter resultados rápidos a partir de pequenos códigos feitos na ferramenta Scratch.

É destacado também pelo autor o respeito pelo conhecimento prévio em programação que cada aluno tem, mesmo com pouco, os alunos conseguiram valorizar esse conhecimento, conseguindo escrever códigos mais complexos a cada aula, sem fazer com que os iniciantes se percam no processo, o que também acaba os motivando.

4.2 WILDNER, FRANZEN E GOMES UTILIZANDO HELPBLOCK

Os autores começam a sua pesquisa apresentando a disciplina de algoritmos e como ela é a base para qualquer um que deseja seguir carreira em computação, mas que, como já visto, muitos problemas estão relacionados a ela o que acaba por intimidar os alunos, os fazendo reprovar ou evadir dos cursos, sendo considerada a matéria que mais causa reprovação, assim ela virou foco de vários estudos que visam diminuir a dificuldade dos alunos na hora de aprender.

A partir desses índices de reprovação e evasão, os autores desenvolveram uma ferramenta utilizando a biblioteca Blocky, o HelpBlock. A metodologia utilizada na pesquisa foi a realização de um pré-teste, uma intervenção pedagógica usando a ferramenta construída pelos pesquisadores, e por fim uma pesquisa de satisfação em 23 anos da disciplina de Algoritmos e Programação. Segundo Wildner, Franzen e Gomes (2018) pré-teste visava medir o conhecimento prévio dos estudantes e o pós teste medir se o utilitário feito por eles auxiliou realmente no aprendizado de algoritmos.

A ferramenta desenvolvida por eles, o HelpBlock, tem como intuito ser um ambiente fácil de compreender e intuitivo para o usuário. Ela dispõe de um cadastro de exercícios de algoritmo por nível de dificuldade e uma área para programação visual, onde o usuário pode escrever seus programas utilizando os blocos da biblioteca Blockly. Para um teste inicial, os autores utilizaram de exercícios que os professores da disciplina de Algoritmo e Programação recomendaram. Os usuários podem escolher qual problema querem resolver e depois são orientados a uma página de resolução de atividades.

Os pré-testes foram feitos, e eles eram constituídos de duas atividades, em que o aluno deveria resolver os problemas na linguagem Java. O primeiro exercício a maioria conseguiu resolver, tendo apenas alguns problemas como erro de sintaxe. Na segunda tarefa houveram dificuldades como o enunciado do exercício, os autores acreditam que isso foi por conta do envolvimento de porcentagem, mas, ainda assim os estudantes conseguiram ir bem e de novo tiveram transtornos com sintaxe.

Antes de iniciar os pós-testes, os autores apresentaram a ferramenta HelpBlock aos alunos, que tiveram uma interação inicial, com demonstrações e resolução de dúvidas. Após isso, os pós-testes foram realizados, em que os estudantes tiveram que resolver dois exercícios parecidos com os do pré-teste.

Os alunos tiveram dúvidas sobre quais blocos utilizar para realizar determinada função, e justamente a parte de tirar as incertezas dos estudantes foi um ponto de destaque dos autores, que relataram que ficou muito mais fácil, pois eles conseguiam mostrar onde os blocos se encaixavam e o motivo deles funcionarem de determinada maneira, o que também facilitou a aprendizagem dos escolares, que conseguiam seguir programando após poucas explicações.

Após todos esses testes, um questionário foi feito aos alunos, e os mesmos perceberam uma melhora no entendimento e resolução de problemas. Os estudantes alegaram que é possível entender melhor os algoritmos, pois as formalidades ficam por conta do software utilizado.

Os autores concluem dizendo que a ferramenta foi bem satisfatória aos alunos, auxiliando na aprendizagem, na correção de exercícios por parte dos professores, e na compreensão de algoritmos, já que através do utilitário, conseguia-se ver o código desenvolvido em blocos em uma linguagem de programação real. Eles ainda confirmam sua hipótese que novas maneiras de ensinar algoritmos despertam o interesse e curiosidade dos estudantes, e que não adianta somente o professor trazer conteúdo, mas que os próprios aprendizes tenham o interesse em buscar e adquirir esse conhecimento.

4.3 RAFAEL SALAZAR, VALGUIMA ODAKURA, CARLA BARVINSKI

No relato de experiência de Salazar, Odakura e Barvinski (2015) a ferramenta Scratch é descrevida, bem como o seu objetivo, que segundo eles, seria uma abordagem inicial para que qualquer pessoa pudesse programar. Eles investigam o quanto o instrumento de ensino tem sido utilizado para fins educacionais pelo mundo, e é notável a sua popularidade.

Os autores então, decidem investigar qual o nível de motivação dos alunos na aprendizagem de algoritmos, observando sua percepção ao utilizar o Scratch. Para isso, eles apresentam a ferramenta para 22 alunos do curso de Bacharelado em Sistemas de Informação que já cursaram disciplinas de programação e algoritmos, realizam um questionário inicial para saber suas impressões sobre essas matérias e após utilizarem o instrumento de ensino, realizam outro interrogatório para avaliar a sua motivação ao utilizá-lo.

No questionário inicial sobre a experiência dos alunos com as disciplinas de algoritmos e programação, 91% deles acham elas fundamentais para a sua formação, enquanto os outros 9% estavam distribuídos naqueles que acreditam que as matérias são relevantes ou desnecessárias. Os autores ainda descobriram que 95% dos estudantes estavam motivados a cursar essas ciências e a razão desse ânimo segundo os acadêmicos seria a didática e a metodologia utilizada pelo professor, pois essas características afetam a motivação.

O recurso mais necessário levantado pelos alunos foi aulas práticas em laboratório, mesmo que apenas 45.5% afirmaram que realmente gostam de programar, e esse número é motivo, segundo os autores, da necessidade de adoção de práticas que incentivam o gosto por programação.

Foi descoberto ainda que nas matérias de algoritmos e programação a porcentagem de alunos que foram aprovados na primeira vez foi de 63.7% e 68.2% respectivamente, porém, os estudantes que ainda estão tentando ser aprovados ou passaram na segunda vez, somam 36.3% e 31.8%, respectivamente, o que é um número relativamente alto.

Depois da utilização do Scratch pelos alunos, os autores fizeram outras perguntas a eles, e de começo, descobriram que grande parte deles (90.5%) não conheciam a ferramenta antes e 71% afirmou que ela seria muito útil para o ensino, porém, apenas 38.1% dos estudantes disseram que gostariam de ter aprendido com o utensílio.

Os autores descobriram também que 72% dos alunos consideram que a ferramenta facilita o entendimento de conceitos de algoritmos, antes de entrar em uma linguagem de programação concreta.

Com base nesses dados, o relato de experiência de Salazar, Odakura e Barvinski (2015) diz que a motivação não é um dos fatores determinantes para aprovação nas matérias de algoritmos e programação, porque apesar de muitos estarem motivados, isso não foi suficiente para uma parte deles acabar por reprovar.

Por fim, segundo, esse relato, o Scratch não se mostrou suficiente/eficaz para aprendizagem de programação, mas que para os conceitos iniciais ele motiva os alunos, aumentando a sua motivação em aprender.

Nesse sentido, pode-se questionar se o Blockly vai conseguir dar aos alunos uma experiência melhor do que eles conseguiriam com o Scratch, já que eles conseguirão ver o algoritmo montado por eles, não só por meio dos bloquinhos, mas sim em uma linguagem de programação real.

5.0 BLOCKS AND CODES

O jogo desenvolvido como objetivo da realização deste projeto de Conclusão de Curso chama-se Blocks and Codes. Ele foi feito utilizando HTML (linguagem utilizada para criar páginas web), CSS (mecanismo para adicionar estilo as páginas web), Bootstrap (framework utilizado para desenvolver componentes de páginas web) e JavaScript (linguagem de programação utilizada na web) e a biblioteca da Google, Blockly, utilizada para o jogador criar seus algoritmos com bloquinhos e executá-los.

O objetivo do jogo é ensinar as estruturas mais básicas de algoritmos, cada nível possui uma introdução, dando uma explicação inicial ao aluno sobre determinada estrutura, seguido de um exercício onde o jogador tem que desenvolver um algoritmo proposto.

Os níveis dos jogos foram baseados no que o IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos) considera no livro Computer Science Curricula (2013) como tópicos básicos a serem abordados para se ensinar algoritmos, como: introdução a uma linguagem, tipos de dados, controle de fluxo, condicionais e loops, arrays, funções.

No total, são 5 níveis, cada um ensinando uma estrutura básica de algoritmos, sendo o nível 1 o que aborda o tema variáveis e tipos de dados, o nível 2 condicionais, o nível 3 loops, o nível 4 propõe um algoritmo utilizando os temas do nível 2 e 3 juntos, e por fim o nível 5 explora o tema de funções.

5.1 ESTRUTURA GERAL DO JOGO

A primeira página a ser exibida ao jogador é a tela de introdução, ela consiste de uma página web com uma explicação sobre o Blocks and Codes e como o jogo funciona, visando esclarecer possíveis dúvidas iniciais que o usuário pode ter, conforme visto na Figura 15.

Figura 15: Página inicial do Blocks and Codes.

As considerações apresentadas ao jogador no início do jogo são:

- É muito importante que você siga fielmente o que foi proposto.
- Letras maiúsculas e minúsculas contam, portanto, preste atenção aos detalhes.
- O que você fez só será validado quando você “executar” o código, então não esqueça disso.
- Caso você não consiga fazer o código proposto, não se preocupe. As respostas serão disponibilizadas após 3 tentativas.
- Uma lixeira no canto inferior direito é usada para remover os blocos que não se deseja usar.
- Você pode posicionar o mouse por alguns segundos em cima de um bloco, para ver sua descrição.
- Em alguns níveis, foi adicionada uma validação de loops, para que não aconteça de um loop infinito acontecer e dar problemas na aplicação.

Após a exibição da página inicial o jogo irá começar, apresentando uma página de introdução à estrutura e depois indo para a página onde um algoritmo é proposto para o

jogador e ele tem que criá-lo utilizando os conceitos da introdução do nível. A Figura 16 ilustra a página de introdução do nível 1.

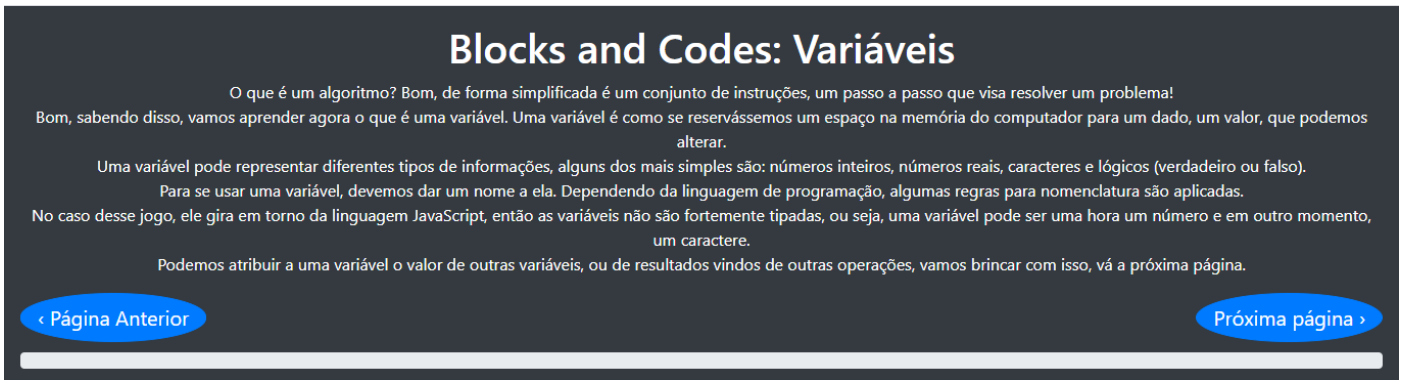

Figura 16: Página de introdução ao nível 1.

As páginas de introdução aos níveis consistem de um título, com o nome do jogo, seguido pelo assunto a ser abordado no nível especifico, um texto explicando ao jogador sobre a estrutura de algoritmo, um botão para voltar uma página e um outro para avançar para a próxima página e uma barra de progresso, que se preenche à medida que o usuário avança de nível no jogo, conforme ilustrado na Figura 17.

Figura 17: Página do nível 1.

As páginas de níveis consistem de um título com o nome do jogo seguido pela numeração do nível, a proposta de algoritmo a ser realizado pelo jogador, quatro botões, um que mostra o código em JavaScript do algoritmo feito pelo usuário, um que executa esse código JavaScript, um que mostra a resposta, ou seja, o algoritmo correto para o nível e um que avança para o próximo nível. Possui também uma área onde os blocos são disponibilizados nas abas laterais, e a área principal onde o jogador irá arrastar esses blocos e criar seu algoritmo. A Figura 18 ilustra a página do nível 1 do jogo com a aba de blocos aberta.

Figura 18: Página do nível 1 com a aba de blocos “Nível 1” aberta.

5.2 APROFUNDANDO NO DESENVOLVIMENTO POR TRÁS DO JOGO

Todos os níveis do jogo tem os seus layouts e funções praticamente iguais, mudando apenas alguns detalhes nas funções de acordo com o nível. Por isso, somente as funções do nível 1 serão descritas aqui, pois a lógica serve para qualquer um dos próximos níveis do jogo.

Para se passar para a próxima página ou voltar para a anterior, nos botões das páginas de introdução dos níveis, foi feito um código em JavaScript que altera para qual página a janela do navegador está apontando, conforme ilustrado na Figura 19.

```
function nextPage() {
        location.href = 'nivel1.html';
    }
    function previousPage() {
        location.href = '../index.html';
```

Figura 19: Funções para alternar entre as páginas da introdução do nível 1.

Quando o usuário clica no botão “Página Anterior” a função previousPage é chamada, levando o usuário para a página anterior, e quando o botão “Próxima Página” é clicado, a função nextPage é chamada, mandando o usuário para a próxima página.

Nas páginas de níveis do Blockly, é importante saber que para os Blocos serem carregados, traduzidos e convertidos para JavaScript é necessário fazer a importação de alguns scripts na tag head do código html, conforme mostrado na figura 20.

```
<script src="../../tcc/scripts/blockly_compressed.js"></script>
  <script src="../../tcc/scripts/blocks_compressed.js"></script>
  <script src="../../tcc/scripts/javascript_compressed.js"></script>
  <script src="../../tcc/scripts/pt-br.js"></script>
```

Figura 20: Scripts necessários para o Blockly funcionar nas páginas web.

De cima para baixo, as duas primeiras instruções fazem o Blockly funcionar na aplicação, a terceira é o que consegue converter os códigos dos blocos para JavaScript e a quarta é a responsável por traduzir os blocos e suas descrições para a língua Portuguesa. Com isso a página já está pronta para utilizar o Blockly.

A seguir, iremos explicar as funções que são responsáveis por fazer a verificação, execução e conversão do algoritmo feito pelo jogador com os blocos, e como é feito a área onde os blocos são arrastados para montagem do algoritmo.

A área onde os blocos são disponibilizados e onde o jogador monta os algoritmos consiste de um XML (linguagem de marcação) em que pode-se colocar o estilo para as abas, seu nome, e escolher quais blocos vão dentro dela. Assim que a página carregar, uma função da biblioteca do Blockly transforma esse XML em uma área executável. A Figura 21 ilustra

o XML da área executável do nível 1 e a Figura 22 mostra a função que transforma esse XML para uma área jogável.

```
<xml xmlns="https://developers.google.com/blockly/xml" id="toolbox" style="display: none">
    <category name="Nível 1" colour="#a55b80">
      <block type="math_arithmetic">
        <field name="OP">ADD</field>
      </block>
      <block type="math_number">
        <field name="NUM">0</field>
      </block>
      <block type="text_print"></block>
    </category>
    <category name="Variáveis" colour="#a55b80" custom="VARIABLE"></category>
  </xml>
```

Figura 21: XML da área jogável do nível 1.

```
var demoWorkspace = Blockly.inject('blocklyDiv',
    {
      media: '../../media/',
      toolbox: document.getElementById('toolbox')
    });
  Blockly.Xml.domToWorkspace(document.getElementById('startBlocks'),
    demoWorkspace);
```

Figura 22: Código que transforma XML em área jogável.

O botão “Mostrar código JavaScript” das páginas de nível executa uma função da biblioteca Blockly que pega o código montado em blocos, transforma para código JavaScript e imprime ao usuário, conforme ilustrado na Figura 23.

```
function showCode() {
    Blockly.JavaScript.INFINITE_LOOP_TRAP = null;
    var code = Blockly.JavaScript.workspaceToCode(demoWorkspace);
    alert(code);
  }
```

Figura 23: Código que mostra o código JavaScript do nível 1.

O botão “Rodar código JavaScript” executa o código gerado em JavaScript e também faz uma verificação dele, comparando o código que foi desenvolvido pelo jogador com um ou mais códigos que foram definidos como corretos para o nível. Ao mesmo tempo, esse botão também faz uma contagem de quantas vezes o usuário tentou acertar o código e caso essa contagem passe de 3, o botão “Mostrar Resposta” é habilitado. Caso o jogador tenha sucesso e acerte o algoritmo, uma mensagem é exibida dizendo que ele conseguiu, e a barra de progresso recebe uma porcentagem, indicando avanço. Uma verificação se o usuário já acertou o código também é feita, fazendo assim que ele não ele não seja verificado infinitas vezes. A validação de loops infinitos, é feita a partir do valor definido na variável window.LoopTrap, no exemplo da Figura 24, 1000 loops são necessários para que o código pare de executar.

```
function runCode() {
    window.LoopTrap = 1000;
    Blockly.JavaScript.INFINITE_LOOP_TRAP =
      'if (--window.LoopTrap == 0) throw "Infinite loop.";\n';
    var code = Blockly.JavaScript.workspaceToCode(demoWorkspace);
    Blockly.JavaScript.INFINITE_LOOP_TRAP = null;
    try {
      if (!T1) {
        verifyCode();
      }
      eval(code);
    } catch (e) {
      alert(e);
    }
}
```

Figura 24: Função que executa o código JavaScript do nível 1.

```
function verifyCode() {
var codeToVerify = Blockly.JavaScript.workspaceToCode(demoWorkspace);
codeToVerify = codeToVerify.replace(/\s/g, '');
tries++;
if (tries >= 3) {
btn.disabled = false;
}

var OPCAO1 = "varN1,N2,SOMA;N1=5;N2=5;SOMA=N1+N2;window.alert(SOMA);";
var OPCAO2 = "varN1,N2,SOMA;N1=5;N2=5;SOMA=N2+N1;window.alert(SOMA);";
var OPCAO3 = "varN1,N2,SOMA;N2=5;N1=5;SOMA=N1+N2;window.alert(SOMA);";
var OPCAO4 = "varN1,N2,SOMA;N2=5;N1=5;SOMA=N2+N1;window.alert(SOMA);";
var OPCAO5 = "varN2,N1,SOMA;N1=5;N2=5;SOMA=N2+N1;window.alert(SOMA);";
var OPCAO6 = "varN2,N1,SOMA;N2=5;N1=5;SOMA=N2+N1;window.alert(SOMA);";
var OPCAO7 = "varN2,N1,SOMA;N2=5;N1=5;SOMA=N1+N2;window.alert(SOMA);";

if (OPCAO1 == codeToVerify ||
OPCAO2 == codeToVerify ||
OPCAO3 == codeToVerify ||
OPCAO4 == codeToVerify ||
OPCAO5 == codeToVerify ||
OPCAO6 == codeToVerify ||
OPCAO7 == codeToVerify) {
var pgBar = document.getElementById("pgBar");
pgBar.style.width = "20%";
        T1 = true;
        window.alert('Parabéns, você conseguiu!');
      }
    }
```

Figura 25: Função que verifica o código JavaScript do nível 1.

O botão “Mostrar resposta” é ativado depois que três tentativas mal sucedidas são realizadas. Ele abre uma imagem com a resposta do algoritmo proposto no nível, conforme ilustrado na Figura 26.

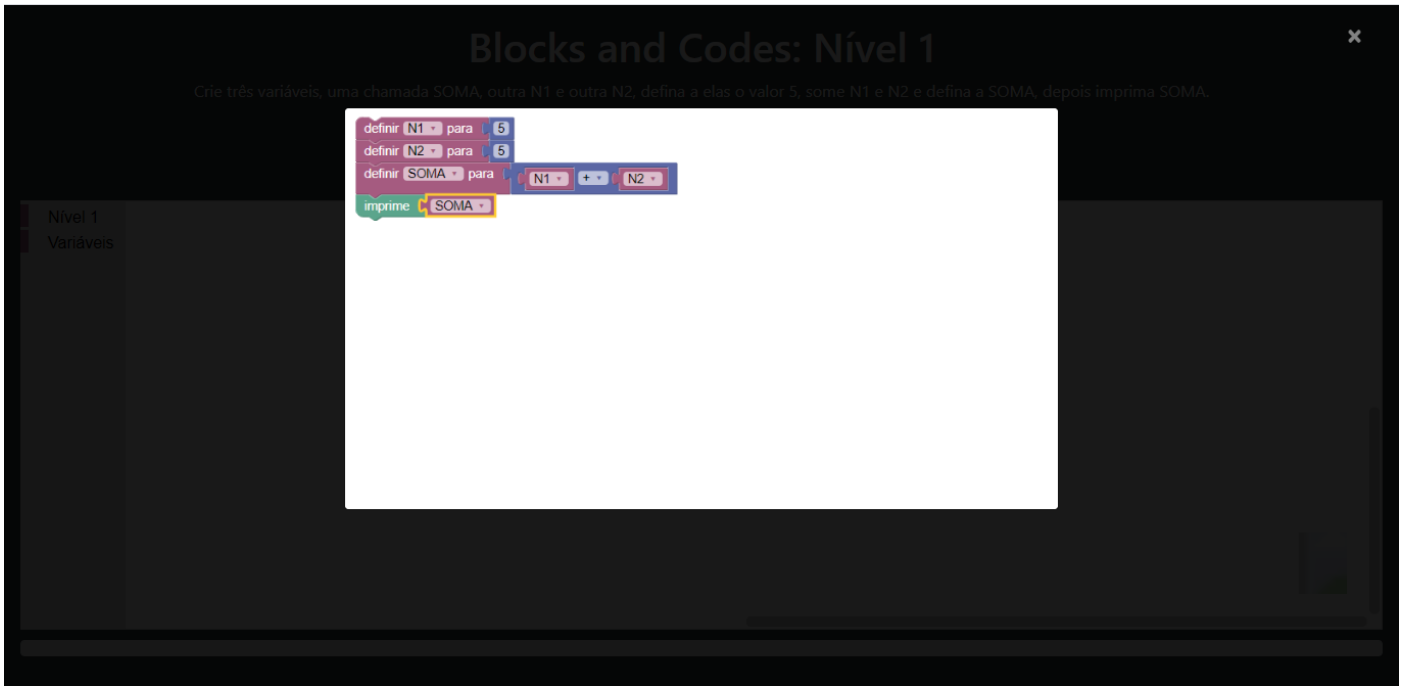

Figura 26: Resposta do nível 1 sendo exibida ao jogador.

```
function showAnswer() {
    var modal = document.getElementById("myModal");
    var modalImg = document.getElementById("img01");
    var captionText = document.getElementById("caption");
    modal.style.display = "block";
    modalImg.src = '../../tcc/image/answer1.png';
    var span = document.getElementsByClassName("close")[0];
    span.onclick = function () {
      modal.style.display = "none";
    }
  }
```

Figura 27: Função que mostra a imagem com a resposta do nível 1.

O botão “Ir para o nível ...” verifica se o usuário já acertou o algoritmo proposto no nível, caso verdadeiro, mostra uma mensagem parabenizando o jogador por ter conseguido e passa para a página de introdução do próximo nível, caso falso, exibe uma mensagem pedindo para que o algoritmo proposto seja feito e que o usuário tente novamente. O código da função do botão é demonstrado na Figura 28.

```
function nextLevel() {
    if (T1) {
      window.alert('Parabéns! Próximo nível!');
      location.href = 'nivel2Intro.html'
    }
    else {
      window.alert('Faça o algoritmo proposto e tente novamente!');
    }
  }
```

Figura 28: Função do botão de passar de nível do nível 1.

6.0 CONCLUSÃO E TRABALHOS FUTUROS

Este projeto teve o objetivo de desenvolver um jogo para auxiliar no processo de ensino de algoritmos para estudantes das séries iniciais de cursos de Computação. Para isso, realizamos diversos estudos e levantamentos bibliográficos a respeito das dificuldades que os alunos dos cursos de Computação apresentam para aprender algoritmos e lógica de programação. Além disso, realizamos um estudo sobre a ferramenta Blockly e a consideramos como uma boa oportunidade para conduzir o desenvolvimento do jogo proposto, que recebeu o nome de Blocks and Codes. Os detalhes da implementação do jogo foram descritos no Capítulo 5, sendo este a grande contribuição deste trabalho de Conclusão de Curso.

O jogo foi totalmente implementado, conforme a proposta inicial deste projeto. Entretanto, devido à situação de pandemia que o país e o mundo estão vivenciando, não foi possível aplicar o jogo para os alunos das séries iniciais dos cursos de Computação. Esta aplicação é considerada importante para podermos validar o processo adotado e verificar se o jogo foi importante para auxiliar no processo de aprendizagem de algoritmos.

Como trabalhos futuros, pretende-se aplicar o jogo desenvolvido Blocks and Codes para alunos de séries iniciais de cursos de Computação, bem como para estudantes de ensino médio, de forma a verificar como o jogo pode contribuir para o processo de aprendizagem de algoritmos entre estes alunos, bem como receber feedback de alunos e professores que possam contribuir para a melhoria do jogo proposta neste trabalho.

REFERÊNCIAS

ACM-IEEE Computer Science Curricula 2013. Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. December 2013.

BORGES, Marcelo Augusto F. Avaliação de uma metodologia alternativa para a aprendizagem de programação. In: VIII Workshop de Educação em Computação – WEI 2000. 2000, Curitiba, Brasil.

CRISTOVÃO, Henrique Monteiro. Aprendizagem de Algoritmos num Contexto Significativo e Motivador: Um Relato de Experiência. In: XVIII Congresso da Sociedade Brasileira de Computação, 2008, Belém do Pará, Brasil.

FRASER, Neil. Ten Things We've Learned from Blockly. In: IEEE Blocks and Beyond Workshop. 2015, Atlanta, Estados Unidos.

HOED, LADEIRA, Raphael Magalhães, Marcelo. Análise de evasão nos cursos superiores de Computação: Uma abordagem usando análise de sobrevivência e algoritmo apriori. In: 4ª Conferência Ibero Americana Computação Aplicada. 2016, Lisboa, Portugal.

PAIXÃO, FORTALEZA, CONTE, Camila, Luiz Leandro, Tayana. Desafios no Ensino de Computação: um estudo da relação entre perfil psicológico de alunos e evasão. In: Anais do XXXIII Congresso da Sociedade Brasileira de Computação (CSBC)—XXI Workshop sobre Educação em Informática (WEI). 2013. p. 720-729.

PASTERNAK, FENICHEL, MARSHALL, Erik, Rachel, Andrew N. Tips for Creating a Block Language with Blockly. In: IEEE Blocks and Beyond Workshop. 2017, Raleigh, Estados Unidos.

RAABE, SILVA, André Luis, Júlia Marques. Um ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos. In: Anais do XXV Congresso da Sociedade Brasileira de Computação (CSBC).2005. p. 2326-2337.

SALAZAR, ODAKURA, BARVINSKI, Rafael, Valguima, Carla. Scratch no ensino superior: motivação. In: Anais do XXVI Simpósio Brasileiro de informática na Educação (SBIE 2015). 2015. p.1293-1302.

WILDNER, FRANZEN, GOMES, Maria Claudete, Evandro, Eduardo Rodrigues. HELPBLOCK: uma ferramenta web baseada na biblioteca blockly para apoio ao ensino de algoritmos. In: Revista Tecnologias na Educação. 2018. p.25.