UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAUL PORTO DE SOUZA

USO DA BIBLIOTECA DE PROGRAMAÇÃO EM BLOCOS BLOCKLY COMO FORMA DE AUXÍLIO AO APRENDIZADO DA DISCIPLINA DE ALGORITMOS E PROGRAMAÇÃO UTILIZANDO A LINGUAGEM C

CRICIÚMA
2018

RAUL PORTO DE SOUZA

USO DA BIBLIOTECA DE PROGRAMAÇÃO EM BLOCOS BLOCKLY COMO FORMA DE AUXÍLIO AO APRENDIZADO DA DISCIPLINA DE ALGORITMOS E PROGRAMAÇÃO UTILIZANDO A LINGUAGEM C

Trabalho de Conclusão de Curso apresentado para obtenção do grau em Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Me. Luciano Antunes

Coorientadora: Prof.ª Esp. Louise Miron Roloff

CRICIÚMA

2018

RAUL PORTO DE SOUZA

USO DA BIBLIOTECA DE PROGRAMAÇÃO EM BLOCOS BLOCKLY COMO FORMA DE AUXÍLIO AO APRENDIZADO DA DISCIPLINA DE ALGORITMOS E PROGRAMAÇÃO UTILIZANDO A LINGUAGEM C

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em informática na educação.

Criciúma, 26 de novembro de 2018.

BANCA EXAMINADORA

Prof. Luciano Antunes – Me. - (UNESC) - Orientador

Profa. Louise Miron Roloff – Esp. - (UNESC) - Coorientadora

Prof. Fabrício Giordani - Esp. - (UNESC)

Prof. Gustavo Bisognin – Esp. - (UNESC)

RESUMO

A área de educação está sempre em constante busca de formas para aperfeiçoamento do ensino. O desenvolvimento tecnológico vem como acelerador para o processo de melhoramento no ensino. A disciplina de algoritmos e programação é básica para todos os cursos tecnológico e mesmo assim possui altas taxas de reprovação. A Blockly consiste em uma biblioteca de programação visual em formato de blocos de código interligados e que foi desenvolvida com fins educativos para área de algoritmos e programação. No decorrer deste projeto foi desenvolvido um ambiente de ensino e aprendizagem de programação utilizando da biblioteca Blockly e linguagem C com métricas voltadas para o ensino. Por fim foi aplicado a ferramenta a alunos de algoritmos e programação do curso de ciência da computação da UNESC, usando de critérios de avaliação de sistemas, como usabilidade e facilidade de adaptação, com o objetivo explorar e analisar a implementação da ferramenta Blockly como ferramenta educativa no auxílio e aprendizado de algoritmos e programação.

Palavras-chave: Blockly. Informática na educação. Ensino. Algoritmos e programação.

ABSTRACT

The educational area is always in constant search for ways to improve teaching. Technological development comes as an accelerator for the process of improvement in teaching. The discipline of algorithms and programming is basic for all technological courses and still it has high failure rates. Blockly consists of a visual programming library in interleaved code blocks format that has been developed for educational purposes in the area of algorithms and programming. In the course of this project, an environment of teaching and learning programming was developed using the Blockly library and C language with metrics focused on teaching. Finally, the tool was applied to students of algorithms and programming of the computer Science course of UNESC, using systems evaluation criteria, such as usability and ease of adaptation, with the objective to explore and analyze the implementation of Blockly as an educational tool in the aid and learning of algorithms and programming.

Keywords: Blockly. Informatics in education. Teaching. Algorithms and programming.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BBPE	*Block Based Programming Environment*
FAETERJ	Faculdade de Educação Tecnológica do Estado do Rio de Janeiro
HTML	*Hyper Text Markup Language*
IDE	*Integrated Development Environment*
MIT	*Massachusetts Institute of Technology*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
UNESC	Universidade do Extremo Sul Catarinense
VPL	*Visual Programming Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O forte crescimento da área da computação nos últimos anos tornou o uso do computador comum para a maioria das pessoas. Com a expansão tecnológica as áreas afins da computação ganharam grande destaque principalmente para os jovens. A grande quantidade de vagas disponíveis nas áreas de computação e afins, faz com que muitos jovens ingressem nos cursos computacionais todos os anos.

Como uma das matérias de maior importância nos cursos computacionais, a disciplina de algoritmos se mostra uma barreira para muitos dos alunos (BARBOSA, 2011). A dificuldade no aprendizado dessa matéria é clara em diversas pesquisas, como em Falkembach e Araujo (2013) e também em Raabe e Silva (2005) que apontam problemas, dentre os quais: os diferentes ritmos de aprendizado encontrados em grandes turmas; a impossibilidade do professor se adequar a cada aluno; a metodologia e o ambiente de realização das atividades utilizado; a resistência ao aprendizado devido a ser um conteúdo nunca visto antes e sem bases no ensino médio e básico; além da clara dificuldade na sintaxe das linguagens, apontado por Ribeiro, Brandão e Brandão (2012). Alguns estudos como o de Rodrigues (2004) citam também a questão da motivação dos alunos, que muitas das vezes foram atraídos para a área por projetos grandes como o de jogos digitais, e que então não se motivam no aprendizado dos pequenos algoritmos propostos na sala de aula.

Métodos para solucionar essas dificuldades na disciplina de algoritmos foram propostos, como por Raabe e Silva (2005) com ambiente de aprendizagem, utilizando de assistentes inteligentes para o ensino, ou Kuk et al (2012, tradução nossa), usando modelos baseado em jogos como forma de motivação para a disciplina. Ainda que venha ao longo dos anos sendo desenvolvidas novas estratégias e métodos para suplementar o ensino da disciplina, o estado ideal ainda se encontra longe do estado atual.

Os problemas citados acima ainda são habitualmente encontrados nas soluções de ensino, uma forma alternativa que abrange os problemas citados que são encontrados tanto pelos alunos, quando na dificuldade de uso e implementação são as linguagens de programação em blocos. Uma alternativa de tecnologia ainda pouco explorada, mas que já vem tendo um desenvolvimento constante a décadas.

A Blockly é uma biblioteca desenvolvida pela Google em 2012 junto a equipe de desenvolvimento do instituto de tecnologia de Massachusetts, usando como base a fundamentação teórica de linguagem de blocos do MIT (PAPERT, 1980, tradução nossa). Ela disponibiliza recursos para se trabalhar com programação em conceito de blocos, tendo como entrada a programação de blocos interligados e de saída a sintaxe correta em linguagem de programação. A biblioteca Blockly é gratuita, e totalmente em Javascript. Por ser web, tem as propriedades de programação em nuvem o que a torna de fácil acesso, além de um navegador, não é necessário instalar qualquer recurso para seu correto funcionamento (GOOGLE INC, 2012, tradução nossa).

A biblioteca Blockly não possui saída de código em linguagem C, apenas nas linguagens Javascript, Python, PHP, Lua e Dart. Na pesquisa realizada pela TIOBE, empresa especializada em medir a qualidade e uso de software no mundo, a linguagem C entra como a segunda no ranking com 6.966% de popularidade, perdendo apenas para o Java (TIOBE, 2017, tradução nossa). No Brasil o C é utilizado em diversas faculdades como linguagem de ensino de programação, por ser uma das linguagens base da programação além do alto índice de uso. A ausência da linguagem C na Blockly faz com que a ferramenta não possa ser usada em muitas disciplinas de algoritmos. Para solucionar esse problema, este projeto adicionou a linguagem C desenvolvendo um módulo de conversão de saída de código em um ambiente de aprendizado desenvolvido usando a Blockly.

Para o módulo de conversão da linguagem foram utilizadas técnicas semelhantes a *transpilers,* que podem ser definidas como um compilador que lê uma linguagem de alto nível e devolve em outra linguagem de programação também de alto nível, para isto o código fonte passa por cinco fases similares ao de um compilador, que consistem em: Análise léxica, análise sintática, geração de código intermediário, otimização independente de máquina e tradução (KULKARNI; CHAVAN; HARDIKAR, 2015, tradução nossa).

Existem questões como as diferentes velocidades de aprendizado dentro de uma sala de aula, que também devem ser levadas em consideração quando se pretende melhorar o ensino de algoritmos. Para isso é necessário o trabalho conjunto do professor e ferramenta, para que os alunos não tenham dificuldade com a didática e fiquem presos a problemas por não entender o enunciado.

De acordo os problemas citados acima e suas supostas soluções, esta proposta tem como objetivo utilizar da biblioteca Blockly para auxiliar os alunos de algoritmos dos cursos de computação e para isso, pretende-se nesse projeto desenvolver um ambiente que integre a Blockly em C para uma melhor didática e aproveitamento da disciplina.

1.1 OBJETIVO GERAL

Aplicar a biblioteca Blockly para o auxílio no aprendizado de algoritmos para alunos de computação implementando a linguagem C como saída da programação em blocos.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) desenvolver módulo de conversão para linguagem C da saída de código da Blockly;
b) desenvolver um protótipo de ambiente de ensino e integrar com o módulo de conversão;
c) realizar testes no protótipo desenvolvido a fim de validá-lo;
d) aplicar a plataforma desenvolvida e os instrumentos de coleta dos dados.

1.3 JUSTIFICATIVA

Os diversos problemas já expostos por pesquisas em questão do aprendizado de algoritmos, levam a altos níveis de reprovação e evasão na disciplina; Barbosa (2011) nos mostra que as médias de reprovação e evasão variam de 23% a 40% em algumas das principais universidades públicas brasileiras; uma porcentagem muito alta para uma disciplina que é básica em todos os cursos de computação. De acordo com esses dados fica claro a ineficiência dos métodos tradicionais empregados no ensino de algoritmos. Como forma de auxiliar na melhora dessas estatísticas este projeto, pretende usar da programação visual como uma forma de facilitar o entendimento e aprendizagem na disciplina.

A biblioteca Bockly tem como princípios uma forma intuitiva e visual de programar (GOOGLE INC, 2012, tradução nossa). Dentre as vantagens da Blockly destacam-se ser gratuita e de código aberto, sem instalação ou *plug-in* para seu correto funcionamento; a própria linguagem visual que facilita o entendimento através da expressão sendo mostrada pelo bloco, principalmente nos códigos mais básicos. Já em códigos mais complexos a ferramenta possui a opção de exportar para uma linguagem de programação (FRASER, 2012, tradução nossa). Além dessas vantagens citadas, ela ainda tem o benefício de ser código aberto, o que a torna flexível, permitindo fazer uma melhor implementação para auxiliar o professor na sala de aula.

A escolha da biblioteca Blockly para este projeto, justifica-se por sua flexibilidade para implementações e vantagens da programação visual, que são mostradas nas pesquisas desenvolvidas em relação ao seu potencial como ferramenta educativa, geralmente para o aprendizado de crianças, como em Silveira Júnior et al (2014), que analisa a ferramenta Blockly como forma de ensino de programação para alunos de ensino médio. Os resultados obtidos são geralmente a ampliação do raciocínio lógico e o desenvolvimento do conhecimento em programação.

Neste trabalho foi adotada a linguagem C como foco da ferramenta, pois é a segunda linguagem mais utilizada de acordo com as pesquisas da TIOBE (TIOBE, 2017 tradução nossa) e é usada onde foi aplicado os testes com alunos, na disciplina de algoritmos do curso de Ciência da Computação da Universidade do Extremo Sul Catarinense (UNESC).

Como forma de avaliação do método foi aplicada a ferramenta na sala de aula com alunos de algoritmos, usando de exercícios para o ensino com a plataforma, e então coletado os dados através de questionário respondidos pelos estudantes.

Neste documento foi proposto uma alternativa para a melhoria no ensino de algoritmos com o uso da biblioteca Blockly de programação em blocos, tendo como objetivo de atenuar os problemas gerados pela dificuldade de aprendizado na sintaxe das linguagens de programação e problemas no entendimento da lógica, além de propor uma solução de fácil acesso com as propriedades web para a implementação dentro da sala de aula.

1.4 ESTRUTURA DO TRABALHO

Esta pesquisa é dividida em cinco capítulos, tendo como o primeiro capítulo uma introdução, os objetivos gerais e específicos e sua justificativa.

O capítulo 2 aborda a descoberta de conhecimento em algoritmos e raciocínio lógico, a abordagem de aprendizado desses conteúdos, bem como a forma de ensino das disciplinas de algoritmos e programação.

No terceiro capítulo é apresentada a tecnologia de programação em blocos, suas diferentes formas de uso. São apresentadas no capítulo 3 algumas diferentes ferramentas de programação em blocos e a Blockly. Além da apresentação das ferramentas e suas qualidades também é abordado o desenvolvimento de métodos para tradução de linguagens de programação baseado em compiladores.

O capítulo 4 aborda a descrição dos trabalhos correlatos a esta pesquisa no meio acadêmico.

O capítulo 5 apresenta o desenvolvimento do projeto como sua metodologia de desenvolvimento, abordando o processo de modelagem UML, implementação do ambiente, desenvolvimento do tradutor, elaboração do questionário, aplicação da ferramenta e resultado obtidos.

Por fim no capitulo 6 está a conclusão do projeto, com o debate dos resultados obtidos e avaliação de pontos positivos e negativos da pesquisa, além de sugestões para trabalhos futuros.

2 DESCOBERTA DE CONHECIMENTO EM ALGORITMOS E RACIOCÍNIO LÓGICO

Nos últimos anos houve um evidente crescimento da área computacional, a Tecnologia da Informação (TI) cresceu de forma a tornar-se uma parte indispensável para as nossas vidas, sendo que, tal progresso tecnológico expandiu-se para as mais diversas áreas (WASIN, 2014, tradução nossa). Esse crescimento desenfreado trouxe um aumento significativo da procura pela área tecnológica, jovens buscando cada vez mais nas universidades uma especialização no meio tecnológico.

Já na fase inicial do curso das áreas computacionais os alunos se deparam com a disciplina de algoritmos que de acordo com as diretrizes do MEC para a grade curricular das áreas de computação, disciplina básica e obrigatória (BRASIL, 2016). Na Ciência da Computação ela inicia no primeiro semestre, onde os alunos têm contato direto com o raciocínio lógico, matemática e programação.

Também é possível destacar que o primeiro ponto exigido no perfil de formação de profissionais da ciência da computação de acordo com o ministério da educação é:

> I - possuam sólida formação em Ciência da Computação e Matemática que os capacitem a construir aplicativos de propósito geral, ferramentas e infraestrutura de software de sistemas de computação e de sistemas embarcados, gerar conhecimento científico e inovação e que os incentivem a estender suas competências à medida que a área se desenvolve (BRASIL, 2016).

Com o destaque na construção de aplicativos e ferramentas, observa-se a importância de um bom aprendizado de programação e lógica para os alunos de bacharelado em computação.

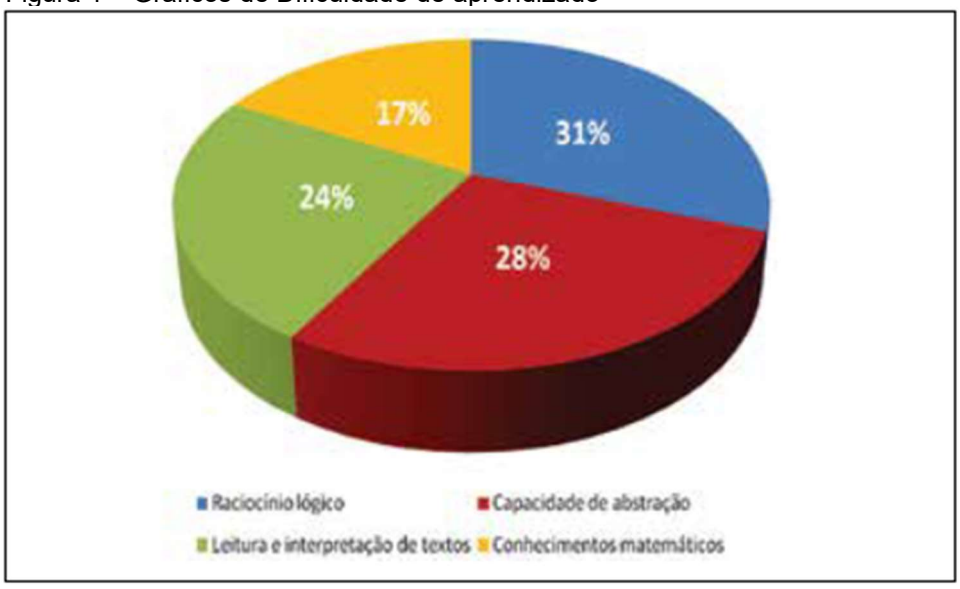
É necessária esclarecer os conceitos e dificuldades encontrados no aprendizado e ensino da lógica programação e algoritmos para ser possível avaliar a dificuldade de aprendizado dos alunos.

2.1 LÓGICA DE PROGRAMAÇÃO

A Lógica de Programação pode ser definida como o raciocínio lógico utilizado para o desenvolvimento de algoritmos, então ao em um programa ele é necessário para encontrar uma sequência de instruções que, quando executadas resolvam o problema proposto (EVARISTO, 2013).

A lógica de programação é a ligação direta do programador com capacidade de desenvolvimento, visto que é uma das habilidades mais importantes ao começar a trabalhar na área de programação. Oliveira et al. (2014) afirma ainda que seria de fundamental importância que o aprendizado de algoritmos começasse nos anos iniciais de estudo, ainda na escola, visto que os cursos exigem habilidades que não são aprendidas anteriormente e que impactam diretamente na habilidade do aluno em compreender e desenvolver algoritmos. Porém, essa disciplina ainda não é abordada na maioria das escolas, sendo um dos fatores que geram o problema no aprendizado de algoritmo. Na pesquisa realizada por Lima Junior, Vieira e Vieira (2015) foram avaliadas as dificuldades de aprendizagem dos alunos na disciplina de algoritmos dentro de unidades de aprendizagem, apontando o raciocínio lógico como o maior problema entre os estudantes (figura 1).

Figura 1 – Gráficos de Dificuldade de aprendizado



Fonte: Lima Junior, Vieira e Vieira (2015).

O raciocínio lógico é o maior problema dos estudantes de acordo com a pesquisa, os valores de problemas com capacidade de abstração e problemas de

leitura e interpretação de textos também são semelhantes em porcentagem. Esses quatro pontos analisados pela pesquisa constituem a base da lógica de programação.

Zanini e Raabe (2012) afirmam que, apesar de existirem diversas estratégias na resolução de algoritmo, independente da estratégia empregada é necessário passar por três etapas: A primeira sendo a interpretação e abstração das informações para que seja possível a compreensão do problema proposto; A segunda etapa que requer o conhecimento do contexto do problema, para assim realizar a sequência lógica para a resolução do problema; e por último o teste que verifica se a sequência lógica proposta resolve ou não o suposto problema.

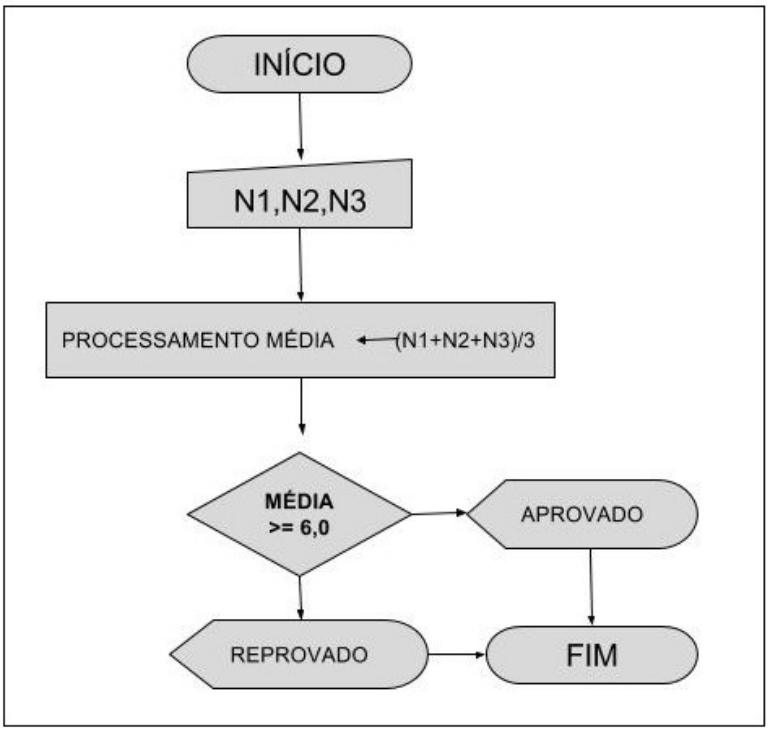
2.2 ALGORITMOS

Uma das principais definições de um algoritmo foi dado por Knuth (1997, tradução nossa) e também de forma semelhante definido por Cormen at al (2009, tradução nossa), definindo um algoritmo como um conjunto de regras que dita a sequência de operações para resolver um tipo específico de problema; além disso ele ainda cita que um algoritmo possui cinco características importantes: Ser finito, um algoritmo sempre deve terminar após uma quantia de passos; ser definitivo, cada passo do algoritmo deve ser precisamente definido; entrada, um algoritmos pode possuir zero ou mais entradas; saída, um algoritmos sempre possuí uma ou mais saídas; Efetividade, é esperado de um algoritmos que ele seja efetivo.

O conceito de algoritmo é muitas vezes ensinado na sala de aula utilizando como exemplo uma receita de bolo ou culinária, pois um algoritmo não precisa ser obrigatoriamente um programa de computador, e sim uma sequência de etapas que quando executadas resolvem um determinado programa.

Abrangendo apenas a área computacional de um algoritmo, ele é definido por Cormen (2009, tradução nossa) como um processo que toma como entrada um valor ou uma série de valores e processa tais valores ou série de valores como saída, sendo então a sequência computacional de passos que transforma o valor de entrada no valor de saída. A figura 2 ilustra em forma de fluxograma a definição de algoritmos dada por Cormen.

Figura 2 – Fluxograma de Média de notas



Fonte: Do autor.

A figura 2 ilustra por meio de um fluxograma um algoritmo de cálculo de aprovação ou reprovação, onde temos o início do algoritmo com a entrada dos valores de notas chamados de N1, N2 e N3, após isso temos o processamento da média simples aplicando a soma dos três valores N1, N2 e N3 e dividindo-os pela quantidade de valores assim gerando a média que passa a próxima figura para a tomada de decisão onde a média deve ser maior ou igual a 6,0 para aprovação ou menor que 6,0 para a reprovação, ao final o algoritmo exibe o resultado e tem seu ciclo concluído.

2.2.1 Formas de representação de algoritmos

Existem diferentes formas de representar um algoritmo, e seus usos são dependentes conforme seu propósito. As formas de representação encontradas em livros têm seu foco em facilitar o entendimento rápido para o leitor. Os fluxogramas e formas de ilustração de algoritmos por representação gráfica, de acordo com Manzano e Oliveira (2005) são uma maneira simples e concisa de representar a linha de raciocínio do programador usando de formas geométricas já preestabelecidas quanto a representação de dados e fluxo de ação do algoritmo.

Além do fluxograma, outras formas de descrever um algoritmo são encontradas nas literaturas, dentre elas as mais comumente encontradas são: Descrição narrativa, pseudocódigo ou pseudolinguagem, fluxograma e Linguagens de programação visuais.

2.2.2.1 Descrição narrativa

A descrição narrativa pode ser considerada a forma mais básica de se exemplificar um algoritmo, e por esse motivo é geralmente usada em livros introdutórios e nas aulas iniciais das matérias de algoritmos. Essa representação consiste em dividir uma tarefa ou problema em passos que, quando seguidos em sequência resultará na sua resolução. Essa forma de representação foi utilizada por Knuth em seus livros introdutórios a algoritmos (KNUTH, 1997, tradução nossa).

2.2.2.2 Pseudocódigo ou Pseudolinguagem

O pseudocódigo ou pseudolinguagem é amplamente encontrado em livros de programação, utiliza de estruturas ou alguns comandos de linguagens de alto nível, porém com características facilitadoras (figura 3).

Figura 3 - Exemplo de Pseudocódigo

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 . . j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Fonte: Cormen et al. (2009).

O autor adotou as convenções de recuo de linha para indicar a estrutura do algoritmo; os comandos *while*, *for*, *repeat-until*, *if-else* similar às linguagens C, C++,

Java, Python e Pascal; o símbolo “//” para indicar comentários entre outras regras similares a linguagens de alto nível (CORMEN et al., 2009, tradução nossa).

Além de pseudocódigos como o adotado por Cormen ainda existem as pseudolinguagem voltadas ao aprendizado onde possuem características mais próximas das linguagens humanas. Um exemplo é a linguagem ILA utilizada no livro Aprendendo a Programar em C de Jaime Evaristo (EVARISTO, 2013). A figura 4 exemplifica o funcionamento da estrutura.

Figura 4 - Exemplo de Pseudolinguagem ILA

```
Variaveis
        Tipo de dado Lista de identificadores
        Funcao
        Inicio
                //comandos da função
        Fim
Inicio
        //comandos do programa principal
Fim
```

Fonte: Evaristo (2013).

A facilidade de compreensão da pseudo linguagem ILA é evidente à primeira vista, por seus comandos serem escritos em português. A estrutura do algoritmo é definida pelo recuo de linha e os comandos são simples e escritos em português como “Ler”, “Escrever”, “Escrever Lista de Parâmetros”, “Inicio”, “Fim” entre outros (EVARISTO, 2013).

2.2.2.3 Fluxograma

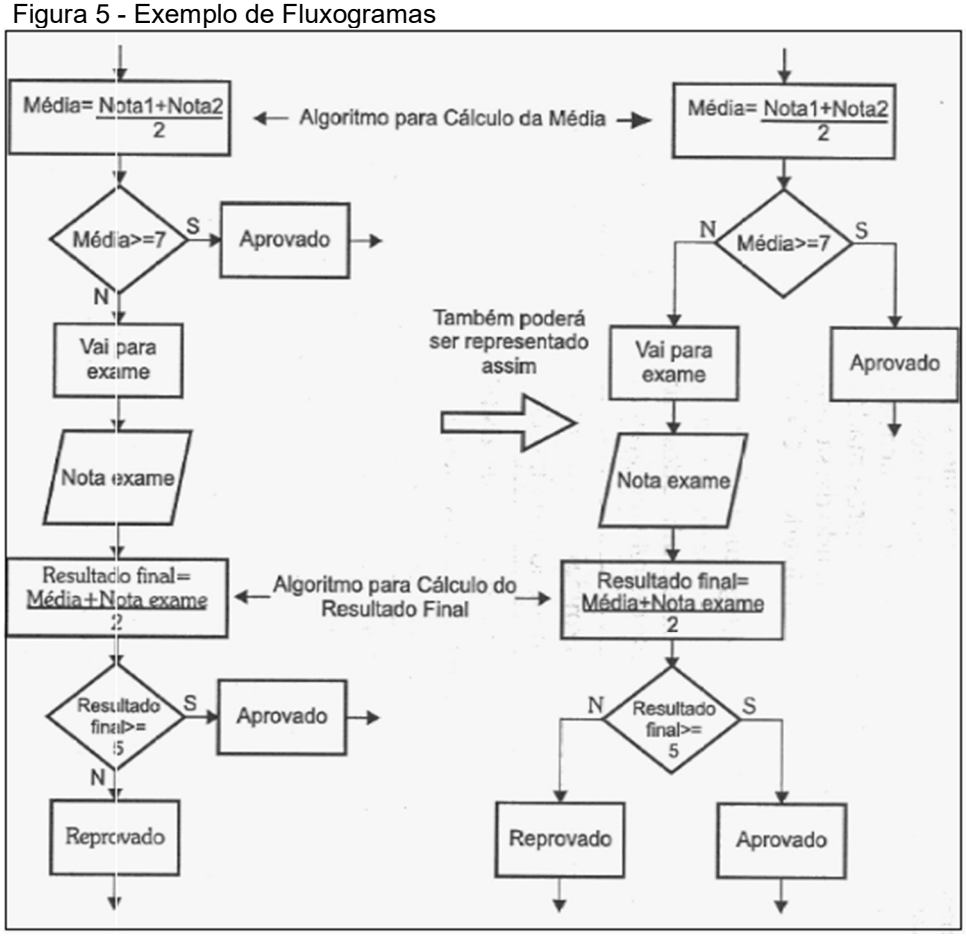
O fluxograma foi desenvolvido pela primeira vez, por Goldstine e Von Neumann em 1947 em seu trabalho não publicado chamado *Planning and coding of problems for an electronic computing instrument, Part II, Volume 1* (KOWALTOWSKI, 1996, tradução nossa; ROSEMANN, 2006, tradução nossa).

Um fluxograma é definido como uma representação gráfica de uma sequência lógica de programação, processo de manufatura, organograma ou estrutura similar; sua finalidade é fazer a representação com símbolos pré-definidos

do fluxo de ação do raciocínio lógico representado (AGUILAR-SAVÉN, 2004, tradução nossa; ROSEMANN, 2006, tradução nossa;)

De acordo com Aguilar-Savén (2004, tradução nossa) a principal característica de um fluxograma é sua flexibilidade e facilidade de uso onde o processo pode ser descrito com infinidade de formas diferentes e fáceis de se compreender, porém a sua força reside em sua facilidade de expressar as informações, pois apenas observando é possível reconhecer o processo descrito. A figura 5 nos mostra exemplos de fluxograma onde dois processos iguais são representados de formas diferentes.

Figura 5 - Exemplo de Fluxogramas



Fonte: Manzano e Oliveira (2005).

A figura 5 representa como dito por Aguilar-Savén uma forma fácil de compreender todo o fluxo lógico do modelo, onde em ambos é possível perceber, à primeira vista, que se trata de uma representação para um modelo de cálculo voltado à aprovação e reprovação escolar.

2.2.2.4 Linguagens de programação visuais

Linguagem de Programação Visual, do inglês *Visual Programming Language* (VPL) é uma grande área que pode ser vista através de sua definição. Expressa mais de uma dimensão usada para transmitir a semântica, ou seja, qualquer objeto potencialmente significativo ou relação, pode representar o que, em uma linguagem de programação textual é representado por uma palavra, esse objeto ou relação, é visto nos exemplos atuais como diagramas, ícones, blocos, demonstrações de ações feitas por objetos gráficos (como visto em jogos sérios), ou até mesmo em uma relação de ordenação antes-depois (BURNETT, 1999, tradução nossa).

De acordo com Alexandrova, Tatlock e Cakmak (2015, tradução nossa) existem diversos tipos de *VPLs* e que podem ser diferenciadas em como elas exploram as expressões visuais, como baseadas em fluxos, ao exemplo do *FlowCode* (Matrix Technology Solutions Limited, 2017, tradução nossa) que é uma *IDE* para a linguagem de programação visual que permite a criação de complexos sistemas elétricos e eletromecânicos a partir da sua interface baseadas em linguagens de programação gráfica; ou baseadas em blocos tais como o Scratch que, de acordo com Oliveira et al (2014), é uma linguagem visual que auxilia a aprendizagem de programação de forma lúdica, através de blocos e animações.

2.3 ESTUDO DAS FORMAS DE ENSINO E APRENDIZAGEM DE ALGORITMOS

Ao se estudar as formas de ensino e aprendizagem de algoritmos, é difícil encontrar um padrão de ensino em todos os cursos que, apesar de ser apresentado na maioria por aulas expositivas elas ainda variam como nos mostra Raabe e Silva (2005) de acordo com as universidades e professores, em todas as formas de ensino é possível perceber sua base em propor problemas à resolução, que consiste no professor apresentando os problemas para os alunos resolverem em forma algorítmica.

Os modelos tradicionais de ensino de algoritmo possuem muitos problemas que são percebidos ao analisar as taxas de reprovação dos alunos. Na pesquisa realizada por Lima Junior, Vieira e Vieira (2015) denominada Dificuldades no Processo de Aprendizagem de Algoritmos, analisou-se as taxas de reprovações na disciplina de

algoritmos entre os anos 2010 e 2012 na FAETERJ-Paracambi, onde obtiveram por resultados finais porcentagens de aprovação de 46% e 54% para reprovação. Além da alta taxa de reprovação avaliada na pesquisa, também foram encontrados valores que mostram que apenas 30% dos alunos são aprovados diretamente e que 40% são reprovados diretamente.

De acordo com Leônidas (2011) em seu levantamento sobre o aproveitamento da disciplina de algoritmos, ele atribui dois principais problemas no aprendizado, o primeiro é o desenvolvimento do raciocínio lógico necessário para a disciplina, enquanto o segundo são subproblemas que incluem a falta de experiência semelhante com programação no ensino médio, metodologia usada pelos professores, a distância do mundo real da disciplina e dificuldades no aprendizado da linguagem de programação.

Alguns autores apontam que algumas das dificuldades encontradas pelos alunos no desenvolvimento da lógica de programação estão nos próprios livros utilizados, Zanini e Raabe (2012) nos mostram que a maioria dos livros de programação básica utilizam enunciados para seus exemplos e exercícios que fogem a realidade e contexto do aluno, são utilizados exemplos matemáticos em mais de 50% dos livros avaliados pela pesquisa, sendo que, de acordo com a pesquisa, o aluno tende a ter melhor desempenho quando utiliza de materiais dentro de seu contexto.

Ao falar dos cursos que ensinam algoritmos e programação, as linguagens normalmente utilizadas são C/C++, JAVA e Python, sendo base no ensino dos estudantes no início do curso, e utilizam para isso ferramentas como DEV C++ que requerem a correta sintaxe para o funcionamento do código necessitando que os estudantes lembrem da sintaxe da linguagem de programação ou eles não serão capazes de compilar seus programas em arquivos executáveis (LIANG; PENG; LI, 2016, tradução nossa).

Em alguns casos, a metodologia utilizada pelos professores no ensino de algoritmos se concentra muito em decorar a sintaxe de uma linguagem qualquer e não o próprio aprendizado da lógica com que funciona a programação. Em alguns cursos observa-se o uso de Plataformas de Desenvolvimento Integrado, do inglês *Integrated Development Environment* (IDE) como o Eclipse, Netbeans, VisualStudio, Code Blocks entre outras e que, de acordo com Noschang, Pelz e Eliezer (2014) o ensino

inicial de programação deve ser focado na resolução de problemas e desenvolvimento da lógica de programação nos alunos. Tendo em vista o uso de IDEs de uso profissional que são focadas na produtividade e não na aprendizagem do usuário, o uso dessas ferramentas ao ensino nos estágios iniciais da programação são inadequadas e geram problemas ao desenvolvimento do estudante.

Os problemas com ferramentas e linguagens não voltadas ao aprendizado são muito estudados, porém, ainda não solucionados. Dentre alguns métodos e ferramentas para ensino de algoritmos estudadas, é possível destacar algumas características de cada método.

Os jogos sérios são um exemplo que unem o aprendizado com diversão e ajudam em uma série de fatores, como na criatividade, cooperação, estímulo na resolução de problemas e principalmente na motivação dos alunos (PONTES, 2013).

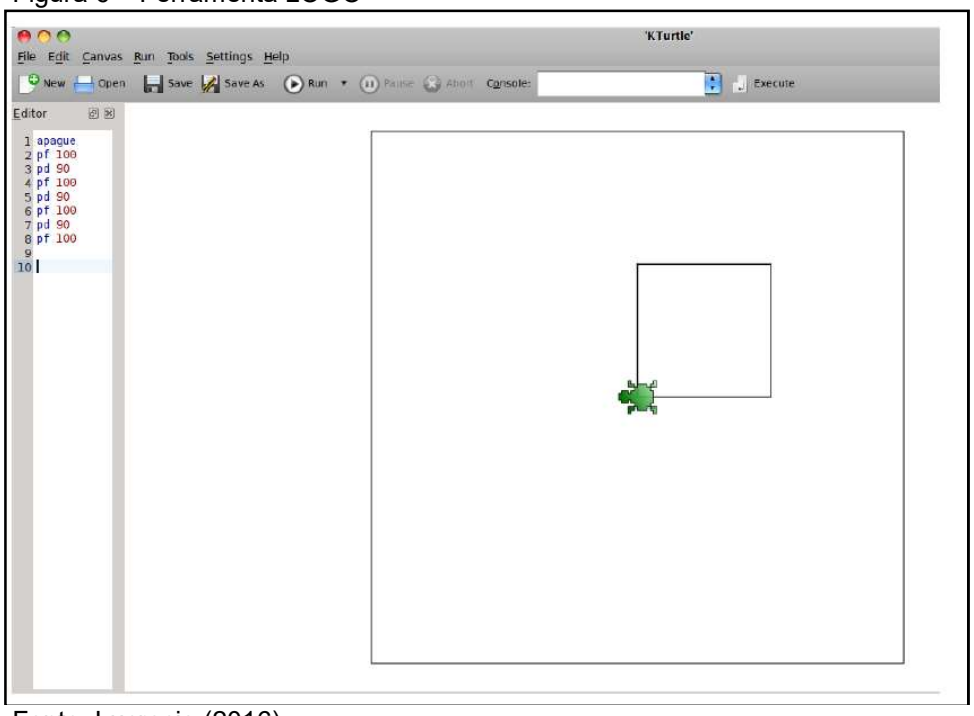
Em ferramentas temos o projeto de Noschang, Pelz e Eliezer (2014) que defendem o uso de uma *IDE* voltada para o aprendizado que utiliza da linguagem Portugol. As vantagens mostradas na pesquisa estão em questão da simplicidade da linguagem Portugol, que utiliza de escrita em português e outros símbolos conhecidos em outras linguagens de programação e do próprio fato do português ser a base dos comandos, facilitando para a maioria dos alunos que não possui compreensão da língua inglesa.

3 PROGRAMAÇÃO EM BLOCOS

A ideia de trabalhar com programação em forma de blocos é antiga, e podemos encontrar referências sobre o assunto desde 1980, onde Seymour Papert coloca a programação através de blocos para o ensino de programação para crianças, e nos diz ainda que, ao invés de ensinar a forma heurística, que na época consistia no aprendizado apenas pela tomada de decisões, é mais simples para a crianças aprenderem a lógica por meio de exemplos concretos como o de simular os movimentos brincando com uma tartaruga, isso conforme o autor cria uma conexão entre uma atividade (de brincar com a tartaruga) e a criação de um conhecimento formal, estabelecendo um meio facilitador para a transferência de conhecimento de um contexto familiar para um novo contexto (PAPERT, 1983, tradução nossa).

A ferramenta LOGO, apresentada na figura 6, foi criada por Papert junto a equipe de desenvolvimento do Massachusetts Institute of Technology (MIT) e constitui-se de uma das primeiras formas de se trabalhar com a programação de forma visual em blocos, trazendo um novo paradigma para o ensino de programação onde o foco não é apenas no código.

Figura 6 – Ferramenta LOGO



Fonte: Lucrecio (2016).

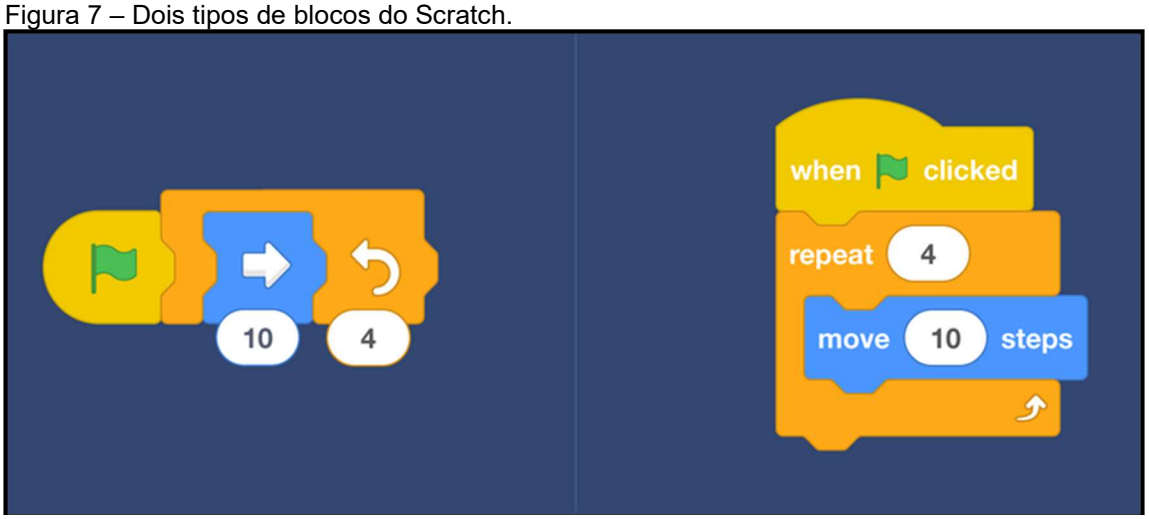
As linguagens de programação em blocos são classificadas dentro da grande categoria das linguagens de programação visual, elas consistem em ambientes de programação que usam de técnicas como a de arrastar e soltar através de blocos de código e que, quando unidos formam a sequência de código do programa, minimizando a possibilidade de erros de sintaxe e a necessidade de memorizar os nomes dos comandos da linguagem (PRICE; BARNES, 2015, tradução nossa). Sua diferença entre as demais linguagens de programação visuais é sua característica, de trabalhar justamente com os blocos, onde outras *VPLs* podem usar como base fluxos ou regras (ALEXANDROVA; TATLOCK; CAKMAK, 2015, tradução nossa).

As vantagens que se destacam nas linguagens de programação em blocos sobre as linguagens tradicionais encontradas na literatura (KAMIYA; BRANDÃO, 2009; LIANG; PENG; LI, 2016, tradução nossa; OLIVEIRA; et al, 2014; PAPERT, 1983, tradução nossa) são:

a) evitar problemas de sintaxe;
b) evita problemas de grafia;
c) evitar problemas de configuração do ambiente;
d) a forma lúdica torna o aprendizado mais intuitivo e agradável;
e) estimula o aprendizado da lógica de computação.

Existe mais de uma forma de abordagem de uma VPL de tecnologia de blocos, o Scratch por exemplo nos traz uma abordagem para crianças, onde é possível desenvolver a ferramenta para que haja a omissão dos comandos de programação nos blocos, facilitando para os mais jovens a forma de funcionamento da aplicação (SCRATCH, 2017, tradução nossa). A figura 7 nos mostra os dois tipos de blocos disponibilizados pelo Scratch.

Figura 7 – Dois tipos de blocos do Scratch.

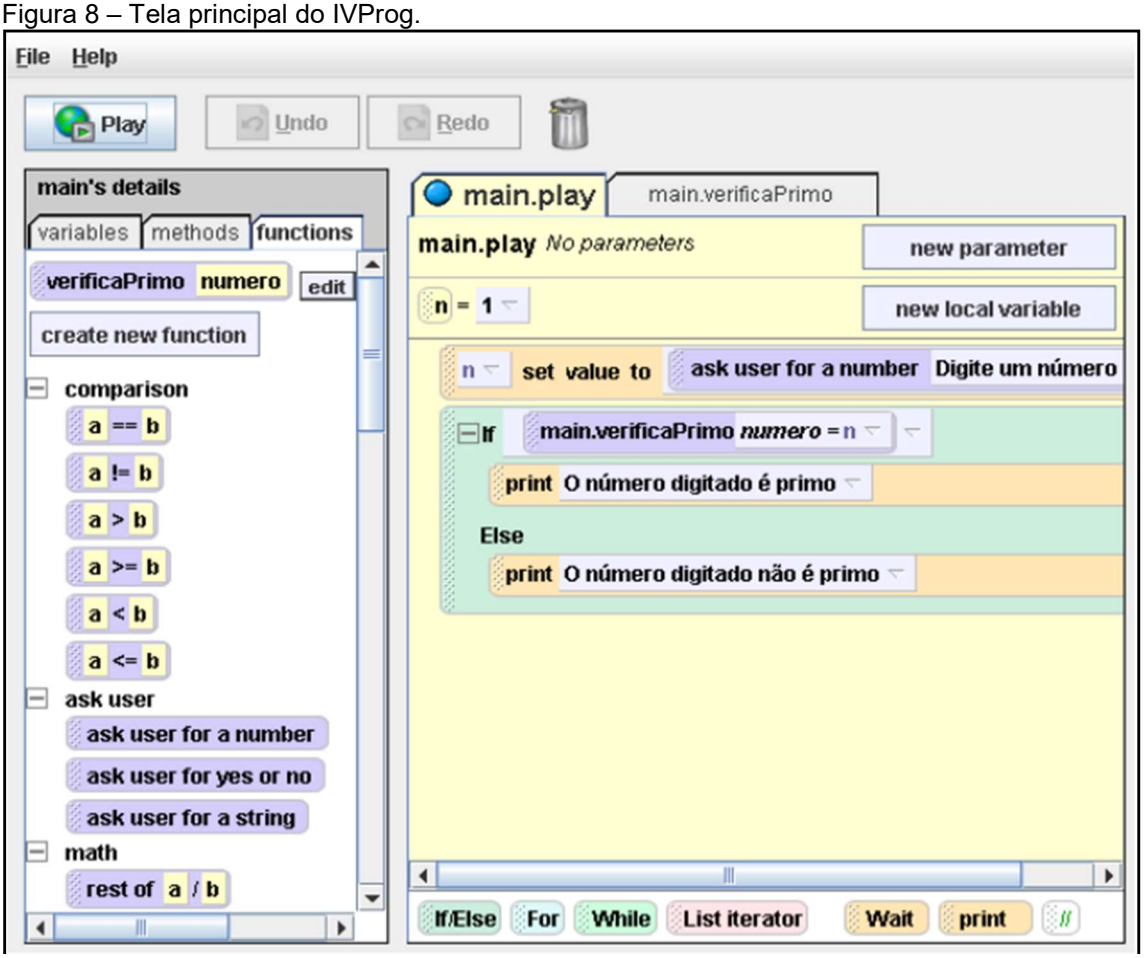


Fonte: Scratch (2017).

A figura 7 representa a mesma expressão lógica, porém uma com os comandos escritos e a outra apenas a implementação por ordem de sequência. Ambas informam pela bandeira verde o início do ciclo lógico que é dado ao clicar-se, após isso tem-se o início de um laço de repetição de quatro vezes de dez passos. A forma de acessibilidade que esta ferramenta consegue obter, principalmente com crianças, fez dela objeto de estudo de diversos projetos voltados ao ensino de lógica de programação a crianças (OLIVEIRA et al, 2014).

Dentre as ferramentas de programação baseadas em blocos, também chamadas de ambiente de programação baseado em blocos ou do inglês, Block-Based Programming Environment (BBPE*)* existem ainda ferramentas que optam por um uso de blocos com mais texto, como é o caso da IVProg que é uma ferramenta baseada no sistema *Alice,* uma ferramenta desenvolvida para melhorar o aprendizado de programação. O IVProg é um projeto para programação visual na internet, que reformulou o *Alice* para ter novos recursos didáticos para o uso efetivo em sala de aula (ALICE, 1999, tradução nossa; KAMIYA; BRANDÃO, 2009). É possível notar na figura 8 da tela principal do IVProg a diferença entre os tipos de linguagem de blocos. O *Scratch* baseia-se em formatos e flechas para mostrar tanto a sequência lógica e de instruções quanto cada a representação de cada bloco, enquanto o IVProg nos traz uma abordagem baseada em blocos de texto bem mais próximo da programação tradicional.

Figura 8 – Tela principal do IVProg.



Fonte: Kamiya e Brandão (2009).

3.1 BIBLIOTECA DE PROGRAMAÇÃO EM BLOCOS BLOCKLY

A Blockly[1] é uma biblioteca produzida com o propósito de auxiliar o aprendizado de programação. Foi desenvolvida pela empresa Google em colaboração com o time de desenvolvimento do laboratório do MIT Media Lab (Laboratório do Instituto de Tecnologia de Massachusetts), que desenvolveu o Scratch.

A definição original da Blockly, de acordo sua criadora, é uma biblioteca que adiciona um editor de código visual para aplicações web e Android, onde utiliza blocos gráficos interligados para representar o código e conceitos como variáveis, expressões lógicas, loops entre outros, permitindo que usuários com pouco conhecimento na área apliquem os princípios da programação sem ter que se

[1] Disponível em: https://developers.google.com/blockly/ acesso em 04 Nov. 2017

preocupar com a sintaxe das linguagens de programação (GOOGLE, 2016, tradução nossa).

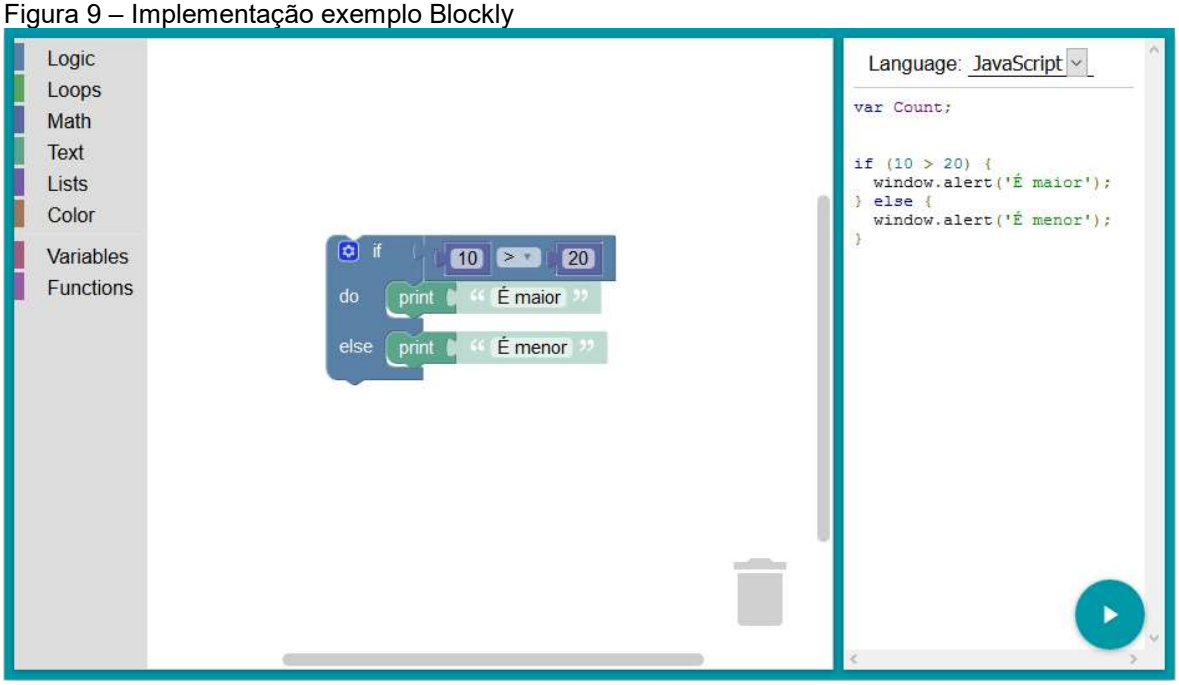
As características da biblioteca Blockly são:

a) possuir código livre;
b) possível implementação tanto para web Android e Ios;
c) funcionar 100% *client side*, não tendo nenhuma dependência de servidor;
d) desenvolvida totalmente em Javascript;
e) compatível com a maioria dos navegadores: Chrome, Firefox Safari, Opera e Internet Explorer.

A biblioteca Blockly possibilita ao usuário editar seus programas, interligando peças gráficas ou blocos em seu ambiente, possuindo construções como variáveis, expressões lógicas, expressões matemáticas e loops no formato de blocos (SHIH, 2017, tradução nossa).

A implementação da biblioteca consiste em uma área para arrastar os blocos e formar a sequência lógica, uma área para a seleção dos tipos de blocos disponível, os blocos de programação arrastáveis e modificáveis e um local para exibição da saída de código em uma das linguagens de programação disponíveis. A saída de código é necessária devido a Blockly não ser compilável, então ela oferece a saída nas linguagens JavaScript, Python, Php, Lua e Dart. Na figura 9 tem-se o exemplo da implementação da biblioteca pelo Google. Ao lado direito da figura é possível observar os blocos separados por categorias, blocos lógicos, blocos de Loops, blocos matemáticos, blocos para textos, listas, cores, variáveis e funções. No meio da imagem fica a área para a elaboração da sequência lógica e logo abaixo a figura da lixeira para a remoção de blocos. Logo ao lado direito posiciona-se a saída do código tendo a escolha da linguagem compilável de saída, e abaixo o botão para geração do código.

Figura 9 – Implementação exemplo Blockly

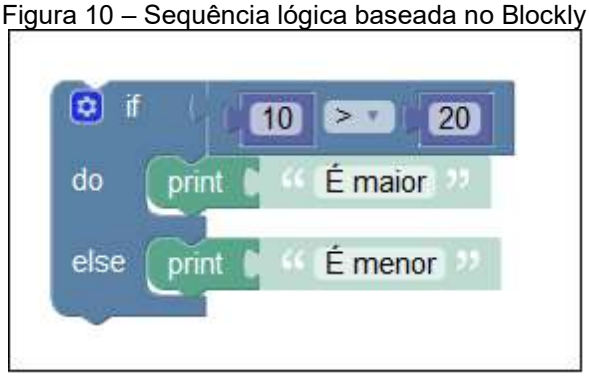


Fonte: Google (2016).

A ferramenta possibilita que os blocos sejam modificados e implementados de acordo com o programador, além disso a interface mostrada na figura 8 pode ser modificada dependendo do propósito de desenvolvimento do projeto para com a ferramenta (GOOGLE, 2016, tradução nossa).

Um bloco representa um comando de programação que em outras linguagens é dado através do comando escrito, porém com a Blockly basta arrastar e conectar para formar a lógica do programa, sem a necessidade de escrita, apenas com o mouse é possível formar toda sequência lógica para a resolução do problema. (SILVEIRA at al, 2006). Na figura 10 temos um exemplo de implementação por blocos de uma sequência do *"if else"* onde, se um valor for maior que o seguinte escreverá "É maior" ou se for menor "É menor".

Figura 10 – Sequência lógica baseada no Blockly



Fonte: Google (2016).

A lógica simples do exemplo não corresponde à capacidade da ferramenta, que pode criar uma lógica complexa sem muitos problemas. O que se destaca na ferramenta é que pelos formatos dos blocos é possível perceber com facilidade qual pode ser encaixado com qual, o que evita diversos erros para um estudante que possui pouco conhecimento na lógica de programação. Além disso a possibilidade de retirar os blocos é benéfica quando se pretende ensinar o conteúdo básico a um aluno, onde ao se limitar a quantia de blocos para os mais simples ele não ficará confuso com diversos blocos complexos e que não são o foco do aprendizado atual.

3.1.1 Módulo de tradução para a linguagem C com a Blockly

A biblioteca Blockly já disponibiliza as saídas de código em JavaScript, Python, Php, Lua e Dart. Porém ainda não é disponibilizada uma saída de código para a linguagem C. O C é uma linguagem de programação muito usada nas áreas de pesquisas e da engenharia, por isso é uma das linguagens base dentre as áreas de estudo da computação, engenharias elétrica e eletrônica. Seu estudo dentre os cursos citados é dado sempre baseado em textos até agora, o que causa à maioria dos iniciantes na área certos problemas, principalmente ao se tratar do aprendizado da sintaxe da linguagem, e por esse motivo faz com que a maioria deles não se interessem pela linguagem (LIANG; PENG; LI, 2016, tradução nossa).

Com a possível obtenção do código fonte do programa em uma linguagem escrita é possível realizar a tradução para outra linguagem de programação também escrita. A técnica de tradução de uma linguagem de programação para outra é chamada de *transpiler*, e consiste no uso de etapas de um compilador que lê a linguagem de alto nível e reescreve-na em outra linguagem de alto nível, usando para isso cinco etapas que consistem em: Análise léxica, análise sintática, geração de código intermediário, otimização independente de máquina e tradução (KULKARNI; CHAVAN; HARDIKAR, 2015, tradução nossa).

A análise léxica é a primeira tarefa que um compilador executa, sua execução começa logo que é recebida a entrada dos caracteres, então a análise léxica realiza a identificação e associação das palavras (chamados *tokens* em compiladores)

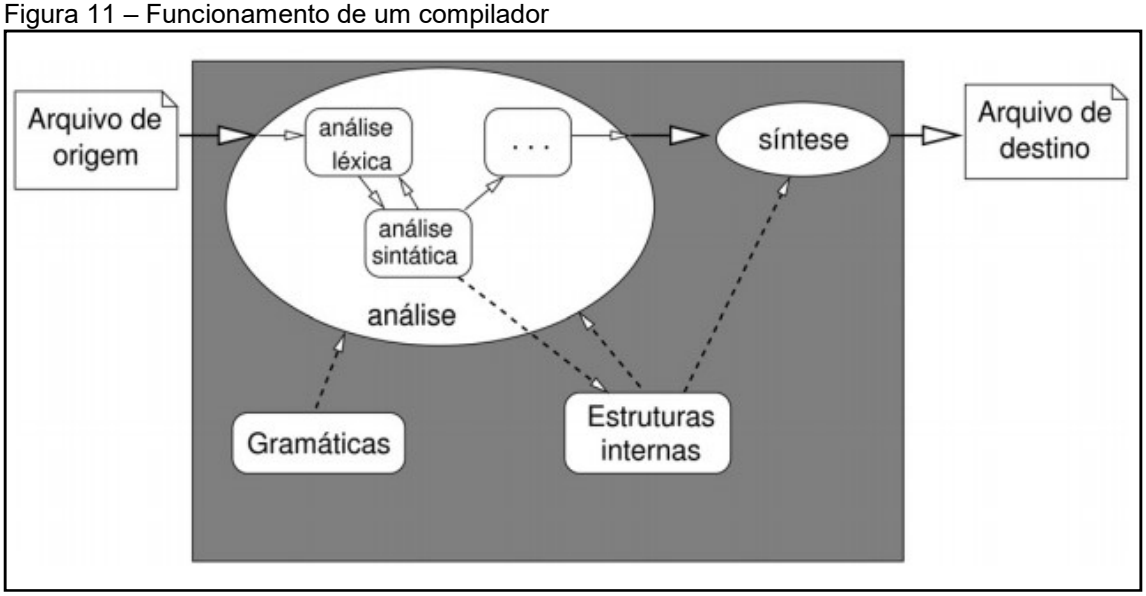
com seu tipo, podendo ser nomes de comandos, nomes de variáveis, nomes de funções, constantes, delimitadores, entre outros (RICARTE, 2008).

Nas características da biblioteca Blockly a sintaxe léxica da linguagem é dada pelos próprios blocos de comandos, então não é possível escrever um comando de forma errada, e mesmo onde é possível a edição escrita ainda está ela sujeita as próprias regras da ferramenta impedindo erros de gramática. As ferramentas que usam de blocos de código sem a necessidade de muita escrita, fazem com que os erros de análise léxica, que consistem em erros na gramática de linguagem, sejam minimizados ou quase nulos.

A análise sintática é a tarefa que ocorre logo após a análise léxica num compilador, ela consiste na tarefa de analisar a ordem em que as palavras e a estrutura do código estão dispostas, pois para o correto funcionamento de um código fonte de uma linguagem, não basta que existam palavras isoladas dentro do código fonte, elas precisam obedecer a sequência de regras de combinação de palavras, ou seja, a forma como tais símbolos devem ser combinados. Dessa forma é possível a formação de uma sentença em uma linguagem natural ou um programa em código fonte (RICARTE, 2008). A estrutura gramatical do código, ou seja, a avaliação que a análise sintática executa está sujeita na Blockly, aos encaixes possíveis dos blocos, então problemas de ordem sintática devem ser minimizados.

A figura 11 representa um compilador com a primeira e segunda etapa de análise léxica e análise sintática, além dos demais recursos inclusos para o funcionamento. O processo de um compilador inicia-se desde a entrada do arquivo com o código fonte, até sua saída com o arquivo de destino com sua nova linguagem. A análise léxica, como representado na figura, é a primeira etapa de análise após a entrada do arquivo de origem. Logo após apresentada a análise sintática que utiliza das estruturas internas para sua análise. A síntese é o processo de geração da nova linguagem, baseada na estrutura elaborada na análise, e também é a parte do compilador responsável pelo processo de otimização do novo código.

Figura 11 – Funcionamento de um compilador



Fonte: Ricarte (2008).

Para a tradução da Blockly não é necessária a elaboração total de um *transpiler*, que é um projeto de grande escala e necessita de muito desenvolvimento para um funcionamento correto, isso é, levando em conta a tradução dos conteúdos estudados na primeira fase do curso de ciência da computação que envolve desde de os conceitos básicos de algoritmos até o conteúdo de ponteiros.

No Brasil o “C” é utilizado em várias universidades como uma linguagem de ensino a algoritmos, por ser uma das linguagens base da programação além de seu alto índice de uso no mercado. A ausência da linguagem C no Blockly faz com que a biblioteca não seja usada em muitos dos cursos de algoritmos. Como forma de solução para esse problema esta pesquisa elaborou um meio de tradução da Blockly para a linguagem C, desenvolvendo para isso um módulo que converte a saída de código de ambiente com a Blockly, para a linguagem de programação C.

4 TRABALHOS CORRELATOS

Nos subcapítulos a seguir serão abordados os usos de ferramentas de programação em blocos como forma de auxílio ao ensino. Os quatro subcapítulos tratam, respectivamente, do uso na Blockly Games como recurso educacional (SILVEIRA JÚNIOR et al., 2015), o desenvolvimento da ferramenta IVProg com uso educacional (KAMIYA; BRANDRÃO, 2009), o estudo de padrões de alunos ao usar uma ferramenta educacional de blocos (SHIH, 2017, tradução nossa) e o desenvolvimento de uma ferramenta para o uso de blocos com a linguagem C (LIANG; PENG; LI, 2016, tradução nossa).

4.1 ANÁLISE DA FERRAMENTA DE PROGRAMAÇÃO VISUAL BLOCKLY COMO RECURSO EDUCACIONAL NO ENSINO DE PROGRAMAÇÃO

O artigo de Garibaldi da Silveira Júnior, Fábio Diniz Rossi, Patric Lincoln Ramires Izolan e Jader Renan da Silva Almeida, Análise da ferramenta de programação visual blockly como recurso educacional no ensino de programação foi publicado no III Seminário Argentina-Brasil de Tecnologias da Informação e da comunicação no ano de 2015.

No artigo, os autores aplicam a ferramenta de programação visual Blockly como uma forma de auxílio ao ensino e aprendizagem de programação. Para isso o artigo conta com a análise inicial e superficial de duas ferramentas de programação visual, a Blockly e o Scratch. Devido aos critérios de escolha serem baseados na facilidade de acesso e facilidade para a implementação, a ferramenta escolhida para ter sua análise como um recurso no ensino a programação foi a Blockly. Para sua avaliação foi proposta uma pesquisa de cunho quanti-qualitativo, em que consistia na realização de um roteiro de atividades por parte dos estudantes com o objetivo de introduzi-los na plataforma Blockly e desenvolver o raciocínio lógico (SILVEIRA JÚNIOR et al., 2015).

A pesquisa foi aplicada com alunos, adolescentes que participam do curso integrado ao ensino médio na área de tecnologia de informação, porém não tiveram nenhum contato com programação anteriormente. Os resultados da pesquisa foram divididos em três fatores principais, sendo eles: o nível de satisfação no uso do

ambiente, o nível de desafio oferecido para o aluno e o nível de facilidade no aprendizado com a ferramenta.

O resultado da pesquisa apontou que a ferramenta tem capacidade de aumentar o raciocínio lógico e desenvolver o conhecimento de programação nos alunos, além disso ela obteve níveis altos de interesse na ferramenta por parte dos estudantes, o que reforça a ideia do uso de formas lúdicas no ensino.

4.2 IVPROG – UM SISTEMA PARA A INTRODUÇÃO À PROGRAMAÇÃO ATRAVÉS DE UM MODELO VISUAL NA INTERNET

O artigo de Reginaldo Rideaki Kamiya e Leônidas de Oliveira Brandão iVProg – Um sistema para a introdução à Programação através de um modelo visual na internet foi um projeto de mestrado de Reginaldo e sob supervisão de Leônidas, publicado no XX Simpósio Brasileiro de informática na educação que aconteceu em Florianópolis nos dias 17 a 20 de novembro de 2009.

No artigo o autor apresenta um sistema para auxiliar o ensino de algoritmos e programação, para uso do professor dentro da sala de aula. O iVProg foi desenvolvido como uma ferramenta que possa de fato ser usada dentro da sala de aula utilizando a abordagem de programação em blocos no ensino introdutório da matéria de algoritmos. Para isso, buscou-se como características ser de software livre para o uso de professores e alunos, além do funcionamento Web e a disponibilidade para elaboração de exercícios no repositório Web (KAMIYA; BRANDÃO, 2009).

Para o desenvolvimento do projeto foi feita uma pesquisa em cima das ferramentas para o ensino de programação, e optou-se por usar como base o sistema Alice, que é um sistema para ensino com base no uso de blocos para a programação, e que possuía já implementados alguns dos requisitos da pesquisa. Apesar do sistema Alice já abranger alguns dos pontos requisitados no projeto, o sistema não compreende todas as características buscadas, então tomou-se como base o sistema para o desenvolvimento de uma ferramenta nova que acolha todos os requisitos da pesquisa.

Como resultado do projeto, obteve-se a ferramenta IVProg que possui as características de blocos, o funcionamento Web e repositório para a elaboração de exercícios. Além disso, para comprovar sua eficácia a ferramenta foi aplicada em um

treinamento de introdução a programação e algoritmos, onde obteve resultados satisfatórios.

4.3 MINING LEARNERS' BEHAVIORAL SEQUENTIAL PATTERNS IN A BLOCKLY VISUAL PROGRAMMING EDUCATIONAL GAME

O artigo, Wen-chung Shih 2017, *Mining Learners' Behavioral Sequential Patterns In A Blockly Visual Programming Educational Game* (Descobrindo padrões de comportamento de estudantes em um jogo educacional de programação visual com Blockly, tradução nossa) foi publicado *na International Conference on Industrial Engineering, Management Science and Application* (ICIMSA) que aconteceu entre 13 e 15 de junho na capital da Coréia, Seul, no ano de 2017.

No artigo Wen-chung Shih faz uma análise sobre padrões de aprendizado de estudantes através de algoritmos de descobrimento de padrões sequenciais. O objetivo da pesquisa tem seu foco na descoberta de padrões na aprendizagem de programação de alunos com altas performances e de alunos com baixas performances, ao se jogar um jogo visual e educacional de blocos, o “Blockly Games”.

Como forma de analisar o comportamento dos estudantes, a pesquisa usou como método uma análise composta de combinação de dois padrões normalmente utilizados na literatura, em que consistem em análise sequencial e mineração de padrões sequenciais, no artigo foram combinados para a geração de um método de análise de mineração de dois níveis. Os níveis foram divididos conforme o tipo de ação realizada pelo estudante. O primeiro composto de padrões de operações sequenciais e o segundo de ações sequenciais. Ao funcionamento da ferramenta de captura dos padrões, ambos eram interligados, e obedeciam a ordem, primeiro o descobrimento de padrões operacionais, onde era avaliado conforme o banco de padrões operacionais, e após era pré-processado e enviado para a análise de padrão de ações sequenciais (SHIH, 2017, tradução nossa).

A pesquisa contou com a análise de 45 estudantes, que utilizaram a ferramenta por aproximadamente uma hora. Com os dados obtidos foi possível observar no início o aprendizado com a ferramenta e, logo após a forma em que os alunos realizavam os problemas lógicos.

O padrão encontrado nos alunos que conseguiram uma alta performance, ou seja, que conseguiram completar mais de sete níveis do jogo foi composto da sequência de passos: Programar os blocos, rodar o programa e responder à questão. Os alunos com baixa performance e que responderam menos de 3 questões obtiveram o padrão de sequência: Programar os blocos, rodar o programa e escolher outro nível. Com esse resultado a pesquisa conseguiu identificar que os alunos com baixa performance ao falhar em uma questão optam pela desistência, ou seja, desistem no nível e vão para próximo sem resolver o problema.

4.4 A BLOCK-ORIENTED C PROGRAMMING ENVIRONMENT

O artigo de Tyng-Yeu Liang, Hao-Ting Peng e Hung-Fu Li, *A block-oriented C programming environment* foi publicado na *International Conference on Applied System Innovation*, ICASI (conferência internacional de sistemas aplicados) de 2016.

Liang, Peng e Li (2016, tradução nossa) propuseram o desenvolvimento de um ambiente de desenvolvimento baseado em blocos para o uso da linguagem C, utilizando para isso a biblioteca Blockly como base para seu ambiente. O foco da pesquisa consistiu na extensão da biblioteca Blockly para a linguagem C

Para o desenvolvimento do projeto, o principal foco dos autores foi a implementação de um tradutor de XML para a linguagem C. Além da implementação do tradutor, também foi necessária a integração da ferramenta com um tradutor através de C# WinForms. Para isso elaborou-se uma interface que fazia o recebimento dos blocos do Blockly, incorporava sua referência de XML, e após, fazia a tradução do XML para o C.

Como forma de avaliação do projeto[2] foram realizadas duas formas de análise. A primeira foi uma comparação de eficiência em tempo de programação comparando a ferramenta desenvolvida e o DevC++. Foram utilizados 4 algoritmos no experimento, calcular um número padrão, chamar uma função de soma, selecionar uma operação aritmética para dois casos diferentes e a função *Bubble Sort*. Sete estudantes participaram da pesquisa, e os dados obtidos foram medidos em

[2] Código do projeto disponível em: https://github.com/cra16/cake-core acesso em 02 Dez. 2018. Projeto disponível em: https://cra16.github.io/cake-core/index.html acesso em 02 Dez. 2018.

segundos. Os resultados obtidos com a ferramenta foram todos melhores que com o DevC++, principalmente em métodos usando de loops e switch cases, pois com a Blockly é possível apenas copiar e colar o número de blocos, enquanto que com o DevC++ é necessário copiar e colar uma sequência de código e altera-los.

A segunda avaliação elaborada pela pesquisa, consistiu em questionários para avaliar a experiência de uso com a ferramenta. Os questionários foram criados com o Google Questionários. Foram recebidos 165 resultados do questionário e 135 resultados válidos. As questões foram criadas referenciando o Davis Technology *Acceptance Model* (em português, Modelo de Aceitação de Tecnologia de Davis) e foram divididas em cinco direções: Usabilidade, Facilidade de uso, uso contínuo, hábito e influência. Como forma de obtenção dos dados foram utilizadas cinco opções, apresentadas em ordem: Discordo fortemente, Discordo, Sem Comentários, Concordo, Concordo Fortemente. Os resultados obtidos pela pesquisa obtiveram pontos altos em Usabilidade e Facilidade de uso, enquanto hábito e influência obtiveram resultados baixos.

5 A BIBLIOTECA BLOCKLY COM SAÍDA EM LINGUAGEM C APLICADA AO ENSINO DE ALGORITMOS E PROGRAMAÇÃO

Este trabalho disponibilizou e aplicou a ferramenta de auxílio no ensino e aprendizado na disciplina de algoritmos para estudantes da área da computação do curso de Ciência da Computação da UNESC, usando para isso a biblioteca de linguagem de programação em blocos Blockly.

A implementação do projeto foi dada com a biblioteca Blockly[3] em versão web e desenvolvido a saída de código para a linguagem C. Para o desenvolvimento foi adotada a linguagem de programação javascript, juntamente ao ambiente de desenvolvimento integrado, ou IDE, Visual Studio Code. Foi desenvolvido um módulo de tradução com o propósito de traduzir a saída da linguagem disponibilizada pelo Blockly para a linguagem de programação C, e então implementada ao ambiente da plataforma Blockly. Por última etapa do desenvolvimento foi realizado testes na ferramenta para a correção de possíveis erros.

Ao término do desenvolvimento, a ferramenta teve como saída de código a linguagem C com correta sintaxe e compilável, só então foi iniciada a segunda etapa da pesquisa que consistiu na aplicação da ferramenta em sala de aula com alunos do curso de Ciência da Computação da Universidade do Extremo Sul Catarinense (UNESC). Os procedimentos propostos foram dados em uma única etapa que compreende a aplicação da ferramenta juntamente com exercícios de algoritmos, e posteriormente a avaliação da ferramenta por meio de questionários. As questões aplicadas basearam-se em exercícios dados comumente na disciplina pelo professor e então resolvidos usando a ferramenta de blocos. Ao fim da aula foi distribuído o questionário que se baseia em métricas de plataforma e aprendizado, para medir critérios tanto da usabilidade e experiência de usuários com a plataforma quanto sobre o aprendizado dos alunos com ela. Por fim, foi efetuada a análise dos resultados obtidos através dos questionários.

Os conteúdos abordados tanto para os exercícios com a ferramenta quanto para os questionários de avaliação serão baseados nos conteúdos normais da disciplina da primeira fase do curso de ciência da computação da UNESC, o que

---

[3] Disponível em: https://developers.google.com/blockly/ acesso em 04 Nov. 2017

envolve desde as noções básicas de programação até vetores, porém foi preferido conteúdos que estavam sendo estudados pela turma durante a época da aplicação.

Essa pesquisa foi aprovada pelo Comitê de Ética em Pesquisa da UNESC pelo parecer número 2.923.843 (Anexo A).

5.1 METODOLOGIA

Para obter os resultados levou-se em conta as seguintes etapas metodológicas: Levantamento bibliográfico; modelagem UML; implementação do ambiente de linguagem de blocos com a biblioteca Blockly; desenvolvimento do módulo de conversão da linguagem; implementação do módulo de conversão de linguagem com a ambiente de blocos; realização de teste para correção de possíveis erros; elaboração do questionário; aplicação da ferramenta utilizando de exercícios da disciplina; aplicação do questionário após o término dos exercícios e análise do resultado obtidos.

Na etapa de levantamento bibliográfico foi compreendido e fundamentado os assuntos referentes a biblioteca Blockly, com ênfase na compreensão das suas funcionalidades e deficiências; o estudo referente ao processo de ensino e aprendizagem na disciplina de algoritmos e programação, e por último, técnicas de tradução de linguagens.

5.1.1 Modelagem em Unified Modeling Language

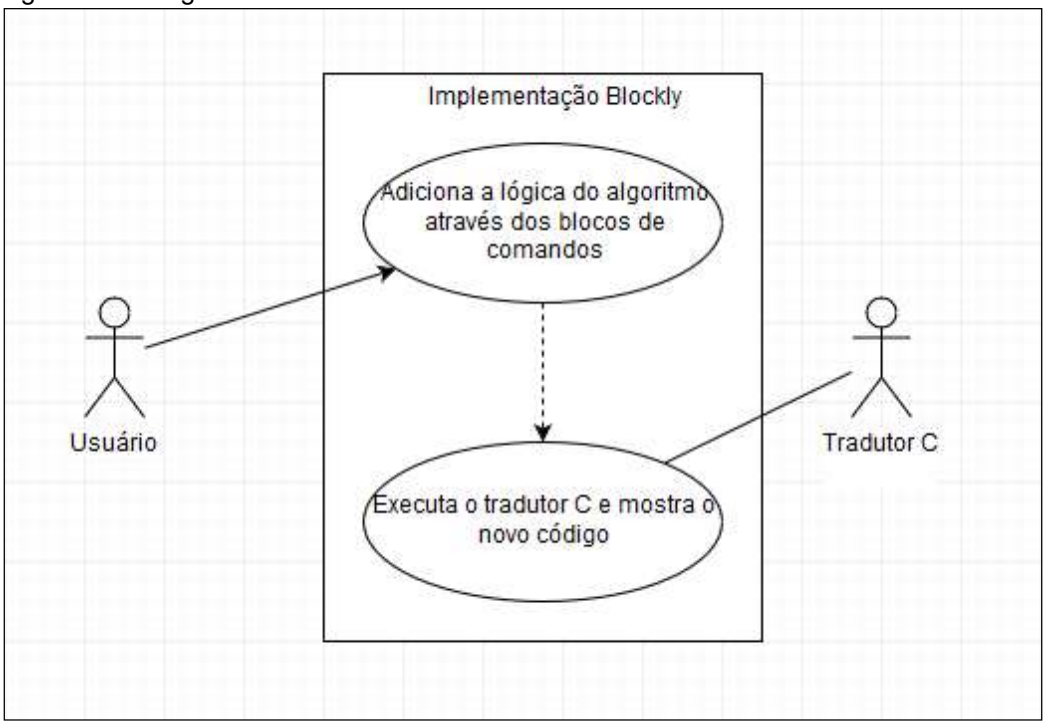
A *Unified Modeling Language* do inglês, Linguagem de Modelagem Unificada (UML) é uma linguagem visual com foco em modelagem de projetos que tem como base o paradigma de programação a objetos. Nos últimos anos tornou-se a linguagem mais adotada para modelagem na área de engenharia de software, seu objetivo principal envolve fatores complexos como a tempo de desenvolvimento, análise de requisitos, prototipação, tamanho do projeto, documentação, manutenção, custos, entre outros (GUEDES, 2011).

De acordo com Rumbaugh, Jacobson e Booch (2005, tradução nossa) a UML é uma linguagem especializada para domínios específicos, e para abranger tal extensão a linguagem é armada de diversos diagramas com propósitos diferentes.

Nesta pesquisa foi aplicado o uso dos seguintes diagramas: diagrama de caso de uso, diagrama de atividade e diagrama de sequência, e foram desenvolvidos utilizando a ferramenta gratuita draw.io[4].

O diagrama de caso de uso aborda a interação do sistema com um agente externo, que é demonstrado pelos bonecos chamados de atores e um determinado sistema. Seu propósito é listar os atores e suas possíveis interações, e mostrar qual ator participa de cada interação (RUMBAUGH; JACOBSON; BOOCH, 2005, tradução nossa).

Figura 12 – Diagrama de caso de uso



Fonte: Do autor.

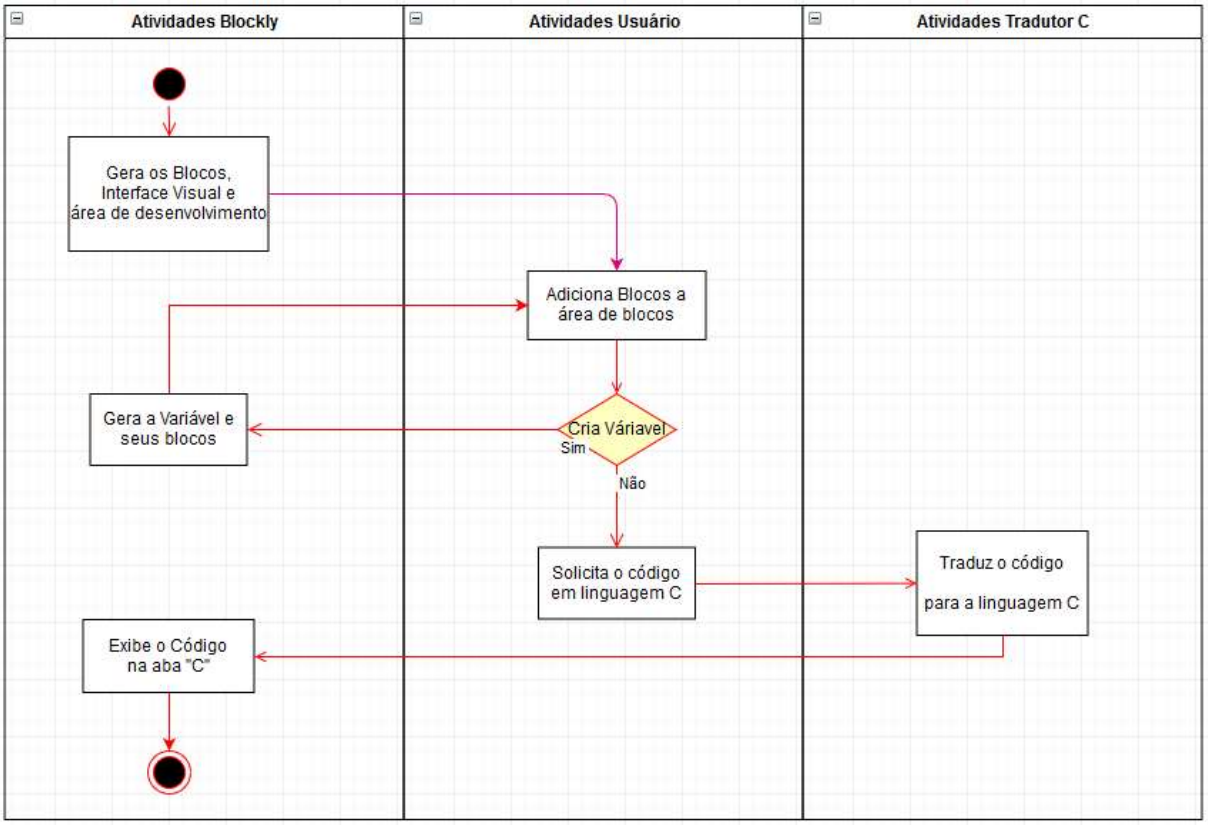
A figura 12 mostra um diagrama de caso de uso, tendo um ator simbolizando o usuário que interage com a parte visual da aplicação desenvolvendo a lógica de programação através dos blocos de comandos, que faz a chamada da execução do tradutor de código C que traduz e envia o novo código para a implementação mostrar em tela.

O diagrama de atividade descreve os passos necessários para a conclusão de uma atividade proposta dentro do sistema, sendo seu objetivo principal exemplificar

[4] Versão online disponível em: https://www.draw.io acesso em 29 out. 2018

o fluxo de controle das atividades (GUEDES, 2015). O diagrama de atividade possui círculos que representam o início e fim da aplicação, e setas que descrevem o fluxo de atividade.

Figura 13 – Diagrama de atividade



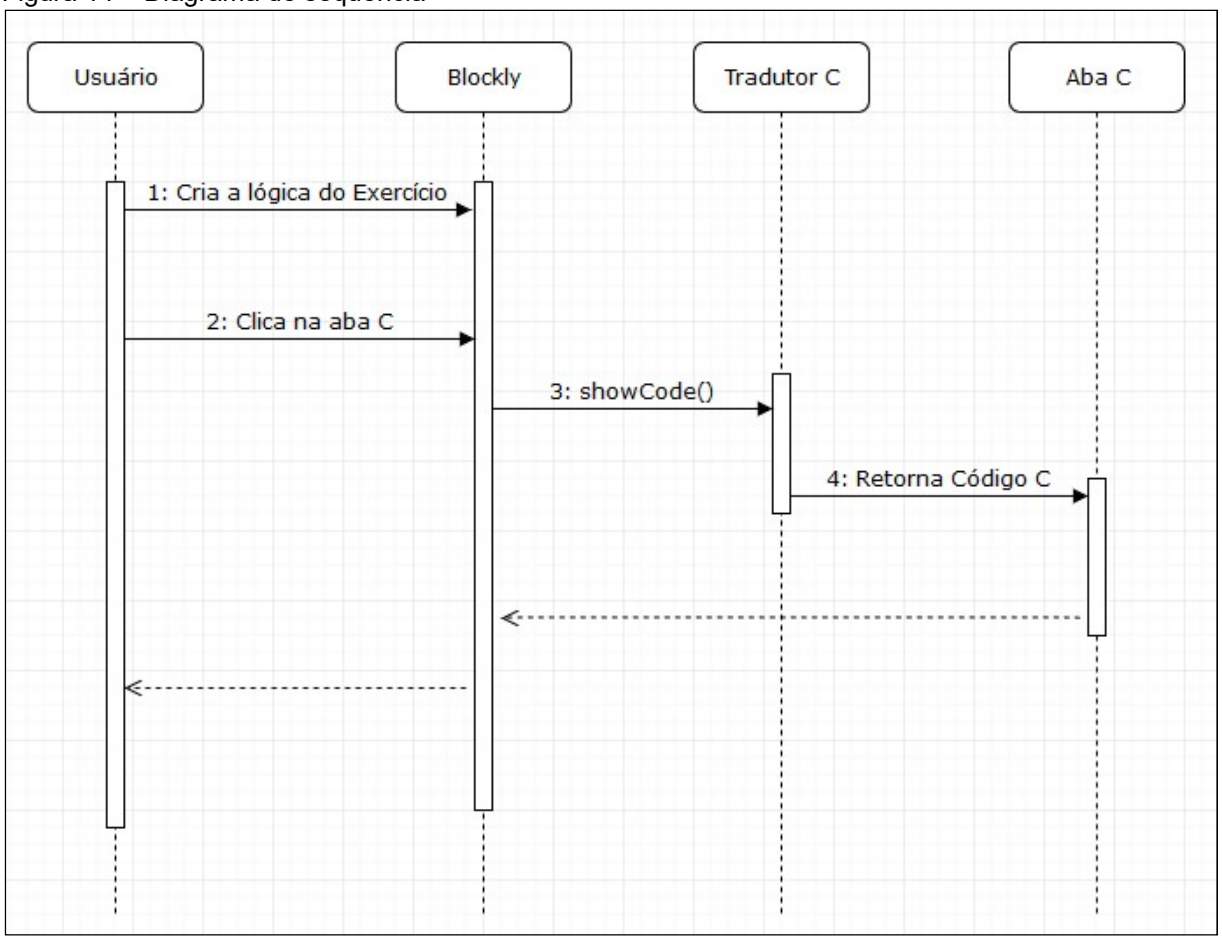
Fonte: Do autor.

A figura 13 exibe a atividade de uso da ferramenta pelo usuário, e aborda as ações padrão do sistema, com a atividade primária de geração do ambiente e blocos, retornando ao usuário o ambiente visual já montado. O usuário tem a opção de criar ou não uma variável, que se caso positivo afetará o ambiente visual necessitando o retorno para as atividades da Blockly montar o bloco de variável. Em última ação do usuário, tem-se a solicitação do código em linguagem C, de onde o fluxo ruma ao tradutor que retorna o código para o Blockly exibir.

O diagrama de sequência trata uma atividade do sistema através de troca de mensagens entre os objetos envolvidos, tendo o processo em uma linha de temporal que mostra o fluxo das ações tomadas para o início e fim do processo definido. O diagrama tem como objetivo identificar o evento gerador, bem como o ator responsável pelo seu início, os métodos envolvidos e o desenrolar de troca de

informações entre os objetos envolvidos para o cumprimento do processo descrito (GUEDES, 2015).

Figura 14 – Diagrama de sequência



Fonte: do Autor.

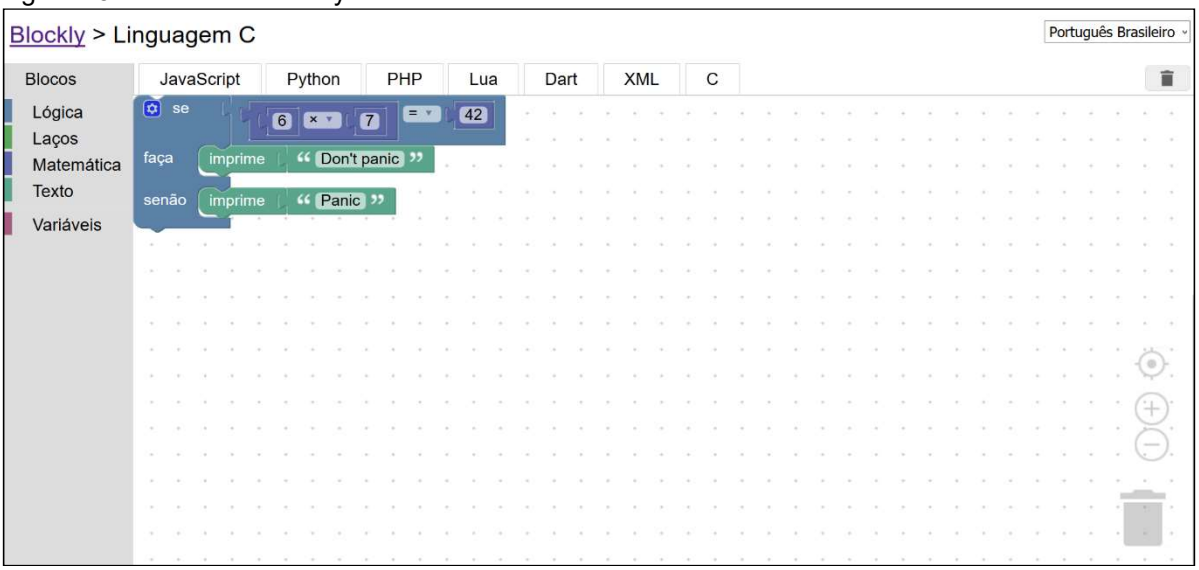
Na figura 14 é exibido os quatro objetos envolvidos no processo de elaboração da lógica e tradução para a linguagem C, bem como o ciclo de vida que cada objeto possui dentro da aplicação. A troca de mensagens principal por parte do sistema acontece com a chamada e retorno do método *showCode()* que chama o tradutor e retorna o código traduzido para o ambiente Blockly.

5.1.2 Implementação do ambiente de linguagem de blocos com a biblioteca Blockly

Para a implementação do ambiente de linguagem de blocos foi utilizado a Blockly em versão Web implementando sua interface com o uso da linguagem Javascript integrado em uma página HTML5.

A interface da Blockly consiste em uma região definida que armazena os blocos por tipos e uma área de trabalho para o organizar a lógica desenvolvida, além disso, uma região que exiba o resultado, em código, dos blocos utilizados (GOOGLE, 2016, tradução nossa).

Figura 15 – Ambiente Blockly



Fonte: Do autor.

A uma implementação completa da biblioteca Blockly é exibida na figura 15, tendo do lado esquerdo a área para seleção dos blocos, no centro a área de trabalho para o desenvolvimento da lógica, abas para as diferentes traduções da linguagem de blocos, além dos botões de exclusão, ampliar e diminuir tamanho, centralizar e o menu de idiomas disponíveis.

Os componentes de funcionamento da biblioteca Blockly são dados pela Interface de Programação de Aplicação, do inglês *Application Programming Interface* (API): o *namespace Blockly*, com as classes *Workspace*, *Toolbox*, *Block* e *Generator*. O namespace Blockly é apenas o nome dado para a chamada da biblioteca, tendo

com intuito evitar conflitos entre diretórios semelhantes (GOOGLE, 2016, tradução nossa).

A classe *Workspace* é a estrutura principal da biblioteca, é responsável por todo movimento dos blocos dentro da aplicação, ela gerencia o movimento dos blocos na tela, arrastar e soltar, excluir, além de fazer as chamadas das funções para os métodos de exibição do código nas demais linguagens. O escopo de funções da classe *Workspace* envolve todas as mudanças possíveis para os blocos desde a criação e exclusão do bloco.

A classe *Toolbox* é responsável pelo bloco e sua caixa de seleção, como exemplificado na figura 15, onde apresenta a caixa de seleção dos blocos localizada a esquerda. O papel da *Toolbox* no sistema é definir os blocos disponíveis para o usuário e fazer suas chamadas.

A classe *Block* representa cada bloco de lógica individual dentro da aplicação. Ela específica diferentes propriedades no que se referem aos blocos, e de maneira geral, atua desde o evento de criação até exclusão do bloco, definindo durante esse ciclo, diversas funções importante para o funcionamento da aplicação. Alguns exemplos de suas funções são de definir as conexões possíveis ou não de cada bloco com os demais blocos, a localização na área de trabalho (*Workspace*), além de diversas outras funções em quesito de funcionamento.

O *Generator* é a classe que refere-se a tradução da lógica blocos para a programação nas demais linguagens disponíveis. Para cada linguagem disponível para a Blockly existem um tradutor diferente, que é ligado diretamente com cada bloco para assim ser possível gerar a linguagem.

Para a implementação da biblioteca neste projeto foi criado dois arquivos, um para abordar a parte visual desenvolvido em linguagem de marcação de hipertexto, do inglês, *HyperText Markup Language (*HTML*)* e com uso de Linguagem Extensível de Marcação, do inglês, *Extensible Markup Language (XML),* e outra que gerencia a chamada de funções e criação da área de trabalho, utilizando de linguagem javascript. Para o desenvolvimento do arquivo de gerenciamento de configurações foi utilizado de exemplos disponibilizados pelo *Get Started*[5] da própria biblioteca e

[5] *Get Started* disponível em: https://developers.google.com/blockly/guides/get-started/web/ acesso em 31 out. 2018.

seguido as especificações necessárias para cumprir o objetivo principal, uma implementação da biblioteca em página Web com a saída de código na linguagem C.

5.1.3 Desenvolvimento do módulo para a conversão da linguagem

O desenvolvimento do módulo tradutor da linguagem, foi implementado em linguagem javascript com o propósito de evitar a geração de dependências no projeto final. O funcionamento do módulo tem como premissa a entrada do código em linguagem javascript, sua análise para as diferenças entre as duas linguagens e ao final o retorno em linguagem C para a implementação da Blockly.

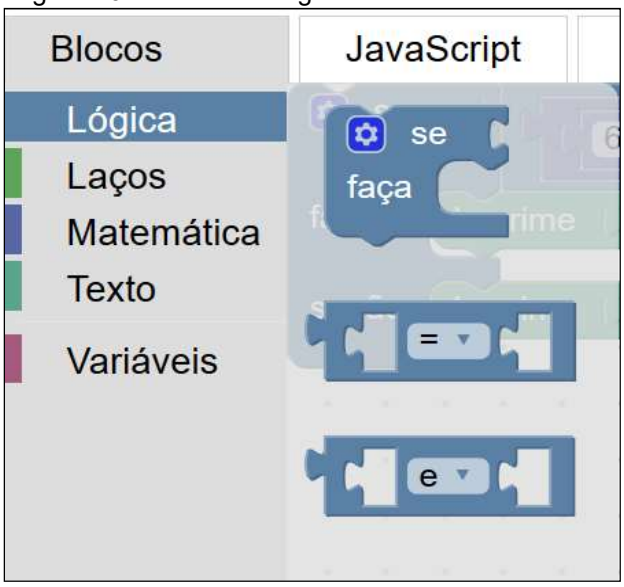
Para o desenvolvimento do módulo de conversão da linguagem foi empregue as seguintes etapas:

a) avaliação dos blocos necessários para o desenvolvimento dos exercícios propostos para os alunos;
b) retirada dos blocos desnecessários para a aplicação;
c) avaliação do código de cada bloco e sua saída;
d) tradução dos códigos *tokens*;
e) tratamento de diferenças nas linguagens.

Como primeira etapa para desenvolver o tradutor foi feita uma avaliação para encontrar os blocos necessários em conformidade com os exercícios propostos. O conteúdo abordado durante a aula da coleta dos dados foi definido de acordo com a matéria atual de estudo dos alunos, sendo ela: condicionais e tomada de decisões, laço de repetição, além dos comandos básicos de leitura e impressão de dados na tela e criação e atribuição de variáveis dos tipos numéricos e vetores de caracteres.

Conforme a etapa de avaliação, foi retirado diversos blocos desnecessários para a resolução do exercício, tendo como objetivo auxiliar na adaptação dos alunos com a ferramenta. Os blocos mantidos na aplicação foram organizados dentro da caixa de seleção em Lógica, Laços, Matemática, Texto e Variáveis, tendo ainda cada tipo de blocos uma cor diferente para facilitar a visualização.

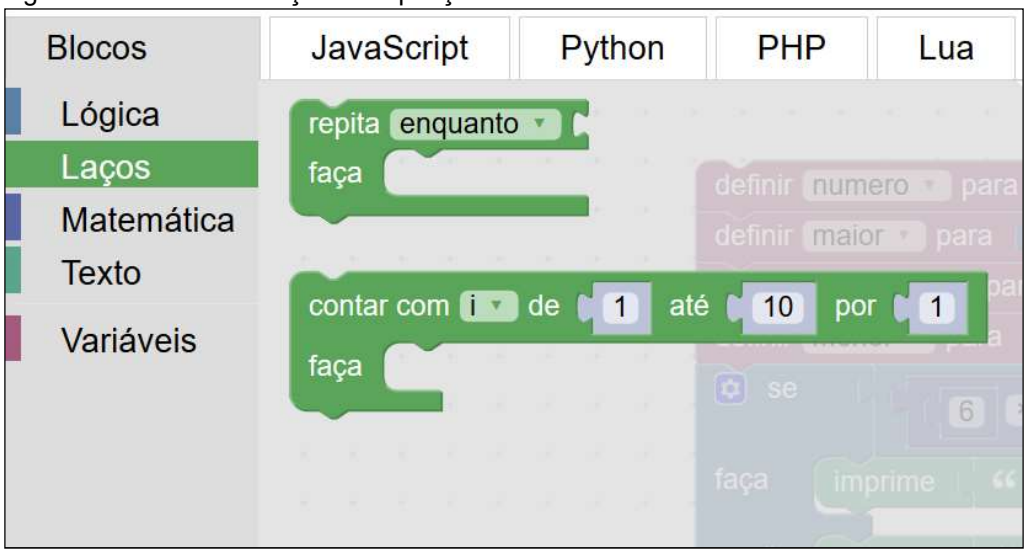
Figura 16 – Blocos de lógica



Fonte: Do autor.

A figura 16 axibe os três blocos de lógica mantidos na aplicação final. O primeiro representa um *if* sendo possiveis adicionar o *else* clicando no botão de engranagem. O segundo bloco é um condicional lógico de compração e possue os opções: igual, diferente, maior que, menor que, maior ou igual que, menor ou igual que. Sua saída em programação é respectivamente “==”, “*!*=”, “>”, “<”, “>=”, “<=”. O último bloco representa a condição lógica “*e*” e “*ou*” em código “*&&*” e “||”.

Figura 17 – Blocos de laços de repetição

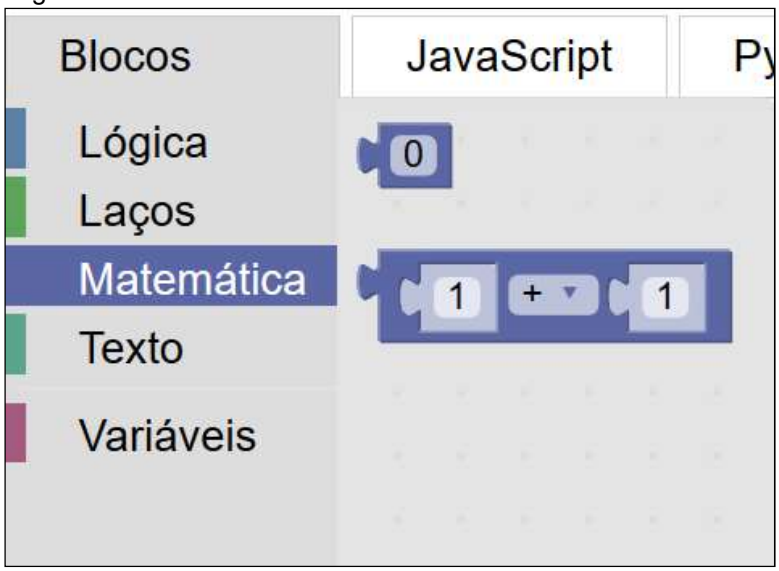


Fonte: Do autor.

A figura 17 apresenta os blocos do tipo laços de repetição. O primeiro representa um *while*, podendo alterar para a negação do *while*, o *!while*. O segundo

corresponde a um *for,* tendo a criação automática da variável de nome *i*, sendo possivel alterar essa variável para qualquer outra criada durante o densenvolvimento da lógica.

Figura 18 – Blocos de mátemática



Fonte: Do autor.

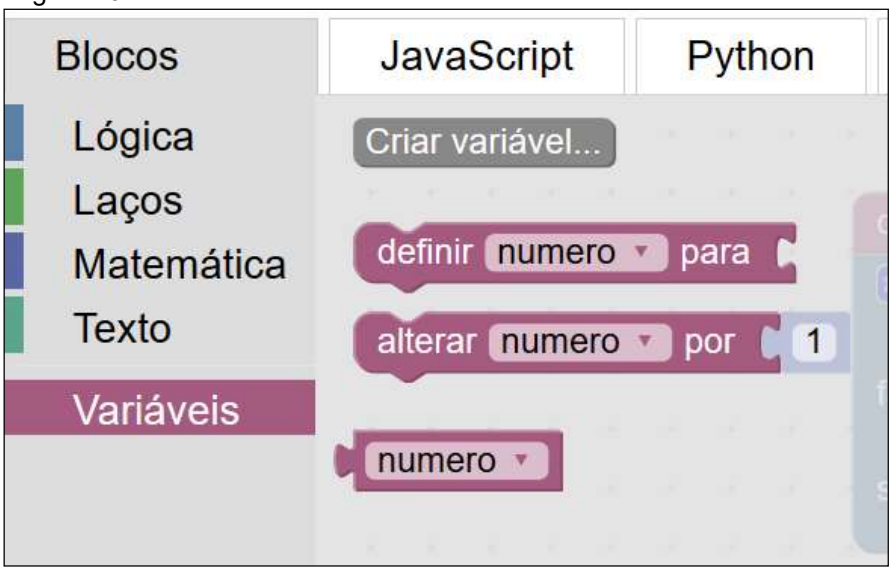
A figura 18 exibe a caixa de matemática com o blocos de numeral e funções aritméticas. O bloco numeral aceita tanto números inteiros como decimais. O segundo bloco contém as funções matemáticas de soma, subtração, divisão multiplicação e potenciação.

Figura 19 – Blocos de texto

Fonte: Do autor.

A figura 19 representa os blocos de texto, tendo as funções de imprimir e pedir valor, além de um auxiliar vazio para a escrita de texto. O bloco de texto vazio serve para a escrita de textos, sendo usado como encaixe em outros blocos. A função de imprimir, ou *printf* em linguagem C, é dado pelo bloco de imprimir. Este bloco suporta a impressão tanto de texto, como números e variáveis. No último bloco temos a função de recebimento de valor, ou *scanf* e *gets* na linguagem C. No próprio bloco é possível escolher o tipo de recebimento como texto ou numeral.

Figura 20 – Blocos de Variáveis



Fonte: Do autor.

Na figura 20 temos caixa de seleção para variáveis. A primeira opção é o botão para a criação de variável, sendo necessário um nome para a variável. Com a variável criada os blocos de definir, altera e variável serão criados pela ferramenta. O bloco de definir serve para a atribuição de valores à variável, e conforme o tipo de dados de entrada será definido sua tipagem. Para numeral, apenas números; decimal, números e ponto; e texto, com letras e símbolos. O bloco de função alterar serve para adição de um valor numérico a determinada variável, e só funciona com uma variável com tipo numérico ou decimal já definida. O último bloco, serve como auxiliar para referir-se a própria variável e encaixa com os demais blocos durante o desenvolvimento do código.

Conforme a avaliação dos blocos e suas saídas, foi desenvolvido o analisador léxico que leva em conta os nomes reservados de acordo com as saídas dos blocos. A primeira etapa para o desenvolvimento de um compilador é a análise

léxica, com a varredura dos *tokens*. O procedimento de varredura dos *tokens* acontece com a leitura do código fonte e identificação das palavras reservadas do sistema, que então são mantidas em memória e reconhecidas pelo analisador (RICARTE, 2008). O reconhecimento do *token* no sistema constitui o primeiro processo para a tradução e acontece a partir do uso de expressões regular e palavras reservadas, que quando encontradas retornam uma saída em forma de cadeia de tokens.

Como segunda etapa do desenvolvimento, aconteceria em um compilador a análise sintática, porém neste projeto o objetivo é a tradução de uma linguagem que já está regrada, pois a própria biblioteca Blockly já faz a análise sintática do código e evita os erros da compilação. Por este motivo, a próxima abordagem usada para o desenvolvimento foi a tradução dos comandos *tokens* da linguagem javascript para a linguagem C. As principais diferenças encontradas nas saídas de código entre as duas linguagens estavam nos comandos de imprimir e pedir valores de acordo com forma de tratamento da variável por tipagem.

Os comandos de impressão dos blocos na linguagem javascript seguem o padrão do comando *token window.alert()* com o valor entre parênteses, sem depender se for um numeral, um texto ou variável. No C para cada tipo, numeral e variável, existe uma forma diferente de escrita, sendo utilizado o comando *printf* que é o comando de escrita em terminal da linguagem C. A cada valor que se pretende imprimir, é necessário colocar a notação do tipo de valor dentro de aspas, sendo assim necessário a definição da tipagem da variável como primeiro processo da tradução. O comando de atribuição de valor segue o mesmo padrão e necessita de definição de tipos para o recebimento (SCHILDT, 1997).

Para viabilizar a implementação de um tradutor levando em conta a diferença de tratamento das variáveis em relação a seu tipo para cada linguagem, foi necessária uma avaliação extra no código, que tem por objetivo encontrar e definir previamente todas as variáveis presentes no código fonte. Para descobrir o tipo, foi avaliado o primeiro valor de entrada que a variável recebe, sendo assim, *int* para números inteiros, *float* para números decimais e vetor de *char* para letras ou caracteres. Em situações onde não é possível definir o tipo do numeral de entrada, como no bloco de pedir valor, foi definido *float* como padrão, pois aceita tanto números inteiros como decimais.

Com a definição de tipagem de variável o comando de impressão foi traduzido para a linguagem C, que então gerou os comandos *printf()* com a notação de tipo "*%d*", para números inteiros, "*%f*" para decimais e "*%s*" para vetor de *char*. O bloco comando para pedir valor segue o mesmo princípio, e foi traduzido pela sequência de um *printf* com o texto e *scanf* para a atribuição do valor da variável através das notações de tipo.

5.1.4 Implementação do módulo tradutor e testes para correção de possíveis erros

Para o processo de implementação do módulo tradutor com o ambiente da linguagem Blockly foi desenvolvido a função de chamada do tradutor e criação do espaço para o novo código. A função é responsável por receber o código da linguagem javascript e manda-lo por parâmetro na chamada do tradutor C, que por sua vez retorna-o em nova linguagem. Com o código traduzido a função o envia para a exibição ao usuário.

Para evitar erros com a ferramenta foi aplicada uma rotina de testes com a ferramenta, onde foram manualmente implementadas sequências de algoritmos para o formato de blocos, afim de testar a integridade do tradutor.

5.1.5 Elaboração do questionário

Para a etapa de coleta de dados foi desenvolvido um questionário a ser aplicado em sala de aula. Sua composição foi disposta em quinze perguntas das quais foram divididas em dez para questões referentes a usabilidade e experiência do usuário e cinco que abordavam sobre a resolução dos exercícios com a ferramenta (Apêndice A).

Para a elaboração das questões foi utilizado as medidas presentes na ISSO 9241-210 de 2011, onde atribuem métricas para interação humano e software, levando em conta critérios de eficácia, eficiência e satisfação (ISO 9241-210, 2011, tradução nossa). As questões desenvolvidas de acordo com a usabilidade e experiência basearam-se nos seguintes fatores principais:

a) Satisfação no uso da plataforma;
b) Facilidade de aprendizado da ferramenta;
c) Velocidade de aprendizado;
d) Redução da necessidade de suporte.

As perguntas referentes a satisfação no uso da plataforma foram três sendo elas:

a) Estou satisfeito(a) com o uso da ferramenta;
b) Prefiro utilizar a ferramenta do que outro software de programação;
c) Escrever o código com os blocos é mais simples.

As questões que abrangem a facilidade de aprendizado com a ferramenta foram duas:

a) É fácil aprender a utilizar a ferramenta;
b) É fácil compreender o que cada bloco representa.

Duas perguntas mediram o critério de velocidade de aprendizado do aluno:

a) Escrever o código com a ferramenta é mais rápido;
b) Foi rápido encontrar os blocos que precisava;
c) Programar com a lógica visual é mais rápido.

Ao fim totalizando dez perguntas, duas questões que referem-se ao uso sem a necessidade de auxílio do professor ou pesquisador:

a) Consegui aprender sozinho(a) como utilizar a ferramenta;
b) Os botões são claros e objetivos em sua funcionalidade.

Para o recebimento das respostas das perguntas de usabilidade e experiência de usuário foi adotado o critério de medidas de questionário fechado, sendo as respostas possíveis: Discordo totalmente, discordo, sem opinião, concordo, concordo totalmente.

As perguntas referentes a resolução dos exercícios com a ferramenta totalizaram cinco e também utilizaram da metodologia de questionário fechado tendo apenas as respostas possíveis sim ou não. O objetivo dessas questões foram avaliar o resultado dos alunos em relação a resolução dos exercícios propostos com o uso da ferramenta. As cincos questões propostas foram:

a) Consegui resolver o primeiro exercício;
b) Consegui resolver o segundo exercício;
c) A ferramenta tornou mais fácil a resolução dos exercícios;
d) A ferramenta me auxiliou na escrita do código de programação;
e) A ferramenta me ajudou a desenvolver a lógica do exercício.

As perguntas *a)* e *b)* tem seu foco em avaliar a produtividade e o resultado do aluno com a aula. A questão *c)* se trata da eficiência geral da ferramenta em relação a outro software. As questões *d)* e *e)* focam-se respectivamente em medir o quesito de eficiência em auxilio para a escrita e lógica dos exercícios.

5.1.6 Aplicação da ferramenta e questionário utilizando de exercícios da disciplina

A aplicação da ferramenta foi dada em uma turma de alunos da disciplina de algoritmos da primeira fase do curso de ciência da computação da universidade do extremo sul catarinense – UNESC. Foi estimado uma turma com 15 alunos matriculados, 14 deles comparecem a aula no dia da aplicação da ferramenta.

No dia da pesquisa foi ministrada, ao começo da aula, a explanação sobre a pesquisa e a solicitação de consentimento dos alunos para a aplicação. Logo após foi explicado o funcionamento da ferramenta e suas pendências, abrangendo sobre o tipo de variável, forma de definição, impressão, lógicos e condicionais, entre outros. O questionário e questões foi entregue junto ao termo de consentimento para os alunos logo após o termino das explicações. Como forma de auxílio aos alunos, tanto

o pesquisador quanto o professor da disciplina estavam apostos para assistir as dúvidas dos alunos com a ferramenta e também com a lógica dos exercícios propostos.

As questões dadas para os alunos foram de exercícios já utilizados amplamente pelo professor da disciplina e consistiam em exercícios já resolvidos pela turma. O primeiro problema abrangia a matéria de laços condicionais e consistiam em gerar um algoritmo que leta a altura e sexo de uma pessoa e calcule seu peso ideal utilizando de fórmulas matemáticas de IMC. A segunda questão abrangeu o conteúdo de laços de repetição e consistiu em gerar um algoritmo para ler cinco números e ao final imprimir o maior, menor e a média dos números digitados (Apêndice A).

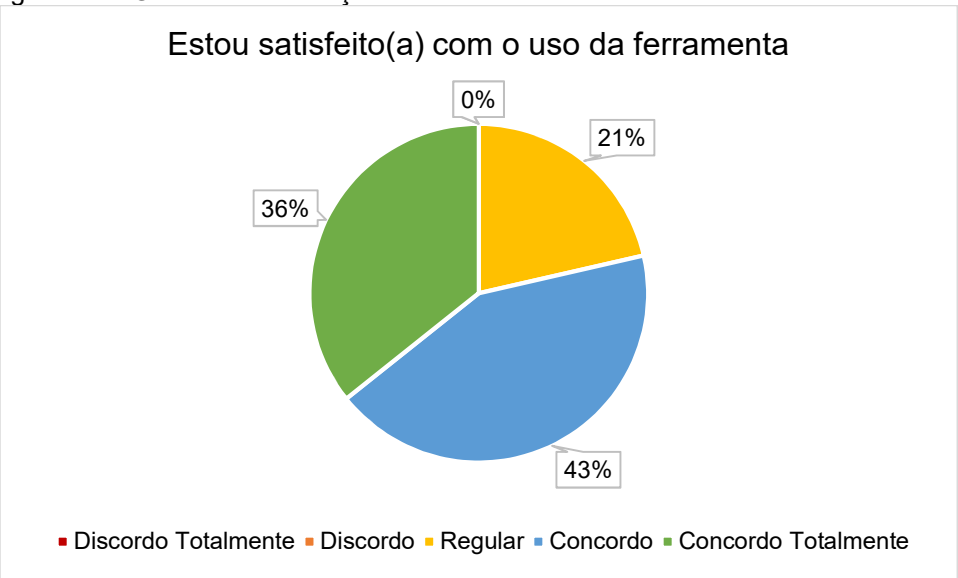
5.2 RESULTADOS OBTIDOS

Os resultados obtidos por meio deste projeto têm o objetivo de avaliar a possibilidade de aplicação de ferramenta de programação visuais como a biblioteca Blockly no aprendizado de algoritmos e programação. Tendo como critério o meio de observação, a aplicação da ferramenta e questionários, que exibem apenas a primeira impressão dos alunos com o uso do sistema, então deve-se levar em conta a que, dados mais precisos apenas serão possíveis com o uso a longo prazo, visto que é necessário tempo para adaptação com a ferramenta e também para o desenvolvimento lógico do aluno. A abordagem tem ainda a finalidade de observar fatores motivacionais dos alunos, dificuldades encontradas e avaliação da ferramenta como método educativo.

Para a análise dos dados foi utilizado das respostas do questionário e observações encontradas durante a aplicação na sala de aula. Por parte do questionário foi gerado gráficos que exemplificam as questões de caráter qualitativo sobre a usabilidade e experiencia de uso. Cada pergunta recebeu um gráfico que mostra o nível de aceitação da afirmação por parte dos alunos.

Para as perguntas referentes as figuras 21, 22 e 23 é observado o quesito de satisfação ao usar a ferramenta.

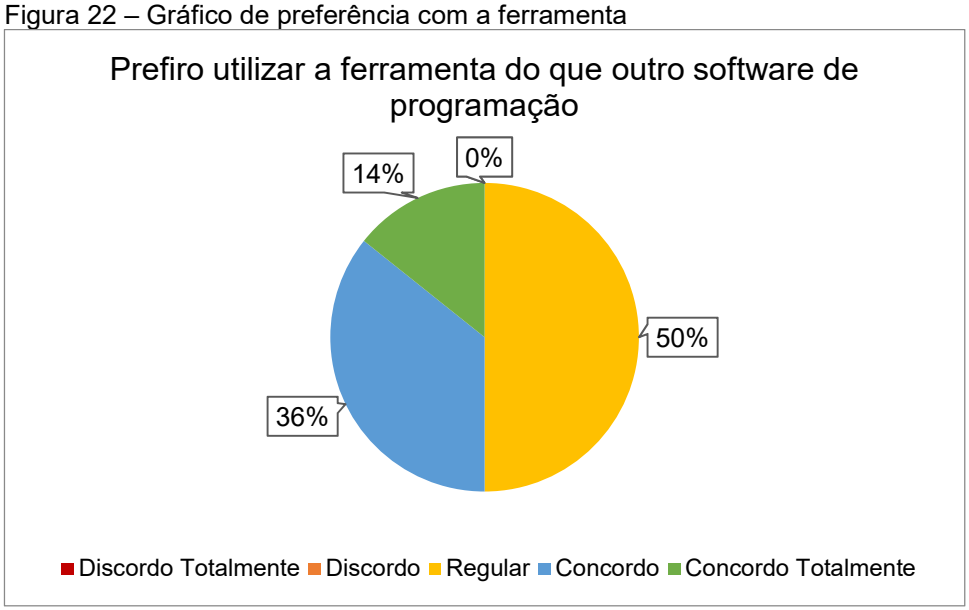
Figura 21 – Gráfico de satisfação com o uso da ferramenta



Fonte: Do autor.

O gráfico 21 aborda a afirmação “Estou satisfeito(a) com o uso da ferramenta” e não obteve nenhuma discordância parcial ou total. Assim adquirindo uma aprovação total de 36%, e parcial de 43%, com neutralidade de 21%. O quesito de satisfação positiva do gráfico, reflete-se na avaliação da aplicação em quesito de aceitação dos alunos com a ferramenta educativa.

Figura 22 – Gráfico de preferência com a ferramenta

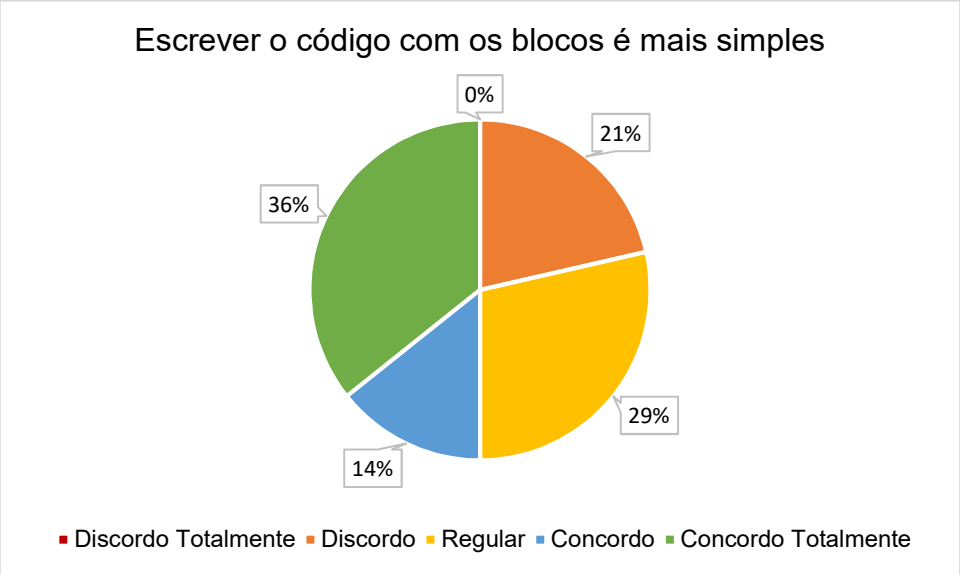


Fonte: do Autor.

No gráfico 22 é abordado a afirmação “Prefiro utilizar a ferramenta do que outro software de programação”, que por sua vez mede o quesito da satisfação em

relação as demais ferramentas já utilizadas pelos alunos. Com 50% dos alunos dando nota regular a afirmação, e concordando parcialmente, 36% e 14% concordando totalmente. Apesar de não haver índices negativos ainda devemos considerar a grande taxa de notas regulares, que se somaram metade dos avaliados, e que expressão uma possível falta de motivação, que é necessária em uma abordagem diferenciada como a de lógica visual por blocos.

Figura 23 – Gráfico de facilidade na escrita do código

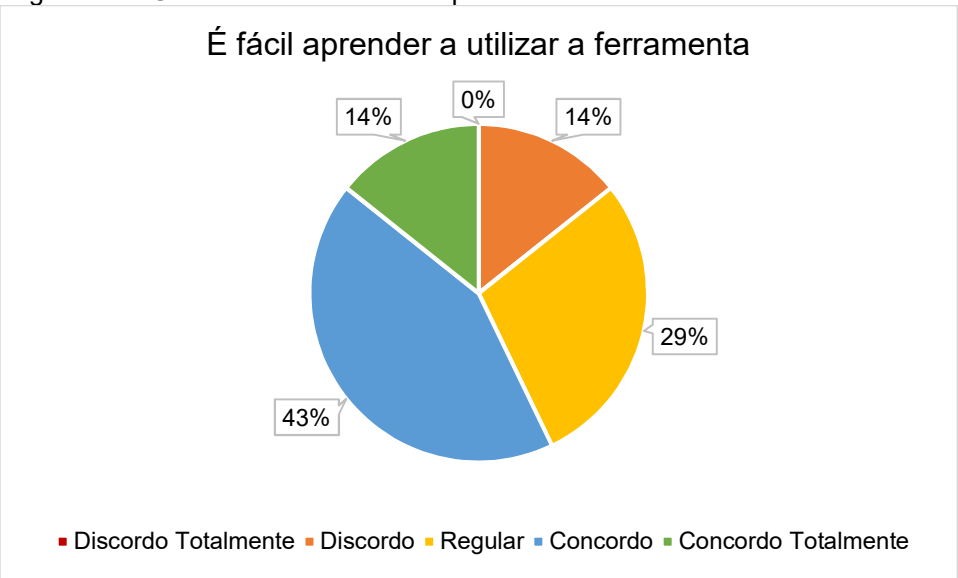


Fonte: Do autor.

O gráfico 23 mostra a afirmação “Escrever o código com os blocos é mais simples” e tem como objetivo medir a opinião do usuário em relação a facilidade de uso em comparação a outras ferramentas. O gráfico mostra opiniões divididas, uma taxa de discordância, opiniões regulares e outra de concordância parcial e total. O resultado do gráfico mostra que, certa quantidade de alunos não acharam a ferramenta simples ao discordar da afirmação, o que pode ser o indicio de dificuldades de adaptação se levarmos em conta o resultado positivo para os dois gráficos anteriores, além da dificuldade com tempo escasso para a aplicação, de apenas uma aula.

Outro quesito abordado nas perguntas refere a facilidade de aprendizado e adaptação com a ferramenta, que é abordado com os gráficos 24 e 25.

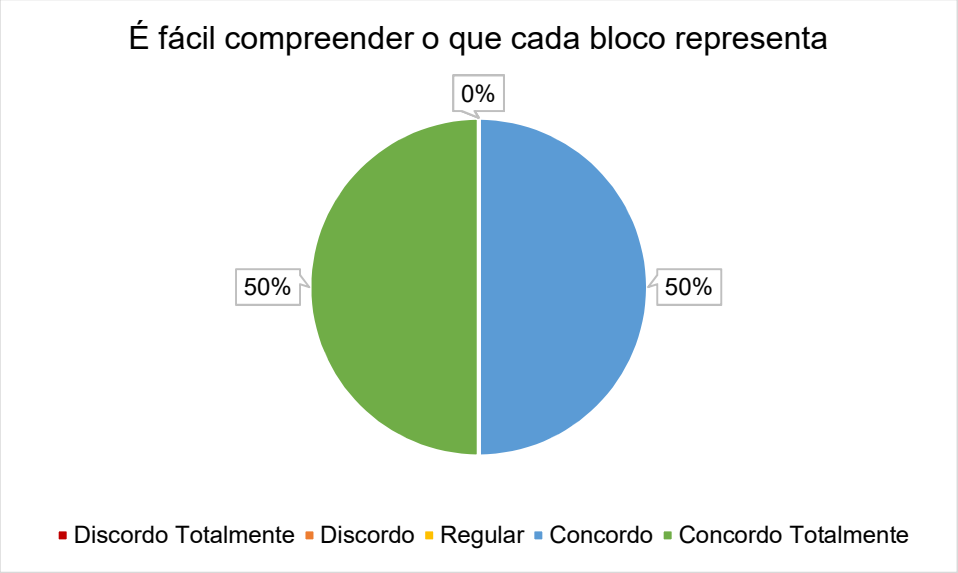
Figura 24 – Gráfico de facilidade de aprendizado



Fonte: Do autor.

Para a afirmação “É fácil aprender a utilizar a ferramenta” a maioria concordou parcialmente, seguido de opiniões regulares. Tais dados podem ser considerados muito positivos se for levado em conta o tempo limitado que os alunos passaram utilizando a ferramenta.

Figura 25 – Gráfico de facilidade de aprendizado dos blocos



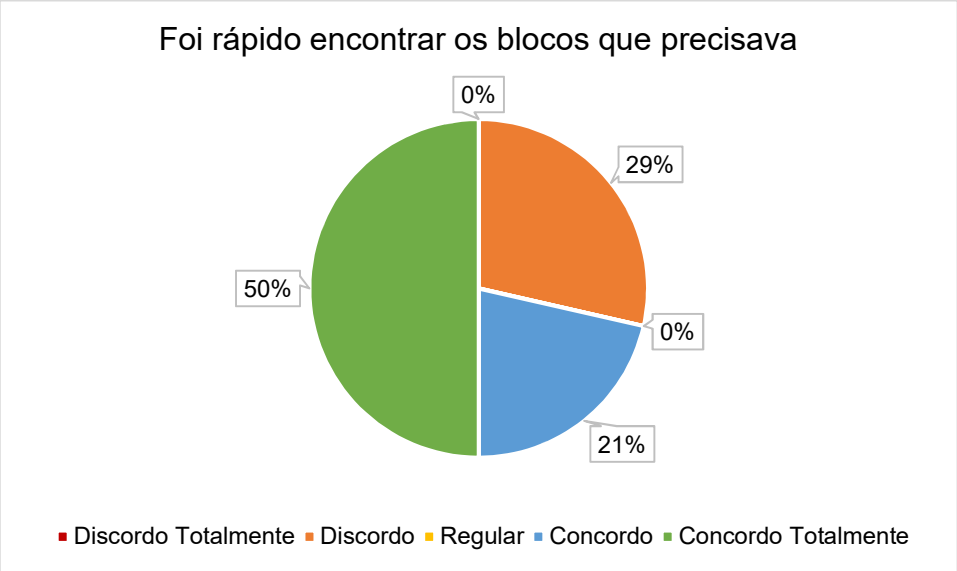
Fonte: Do autor.

Ainda em quesito de facilidade de aprendizado o gráfico 25 aborda a afirmação “É fácil compreender o que cada bloco representa” e recebeu 50% concordância total e 50% parcial. Levando também a avaliação do pesquisador

durante a aplicação, foi possível notar que poucas dificuldades foram encontradas em entender o que cada bloco representa, porém, houve dificuldades em questão de entender o funcionamento e mecanismos de encaixa para cada bloco específico, o que pode ser considerado normal, pois a lógica de bloco trabalha de uma forma totalmente diferente da programação convencional.

Em questão da facilidade de aprendizado geral do aluno com a ferramenta, foi possível notar por parte do avaliador, que as dificuldades encontradas pelos alunos foram nos quesitos que se diferem mais da linguagem de programação habitual, por exemplo a forma de criar e adicionar valores as variáveis.

Figura 26 – Gráfico de velocidade de aprendizado

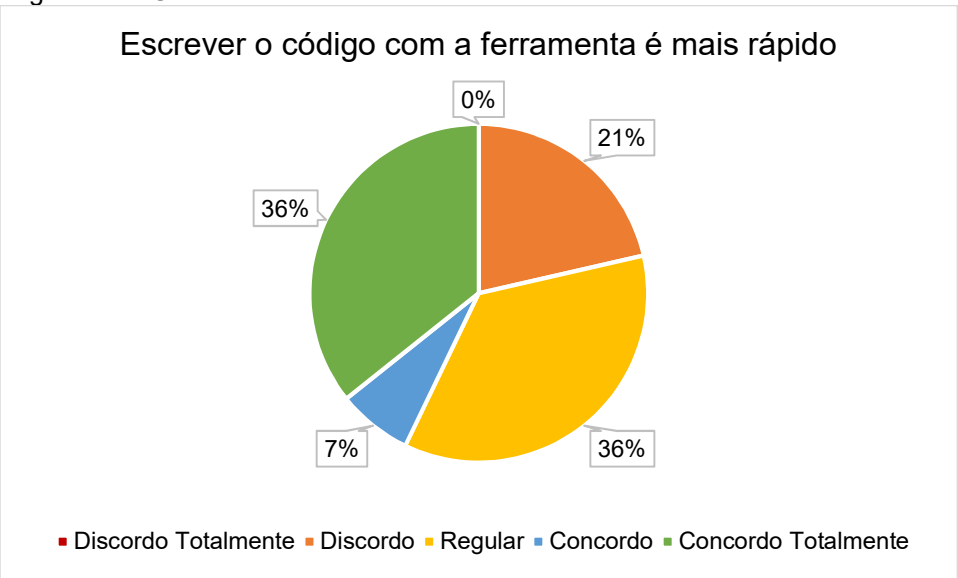


Fonte: Do autor.

A figura 26 mostra o gráfico da afirmação “Foi rápido encontrar os blocos que precisava” e aborda a opinião relativa do usuário com a velocidade para buscar cada bloco necessário para a lógica de seu programa. O resultado com maioria aceitando a afirmação é reflexo da metodologia proposta durante o desenvolvimento do projeto com a retirada de blocos inúteis a resolução dos exercícios.

No gráfico 27, temos em quesito de velocidade relativa a afirmação “Escrever o código com a ferramenta é mais rápido”, e busca a opinião do usuário comparando a aplicação Blockly com a sua ferramenta habitual.

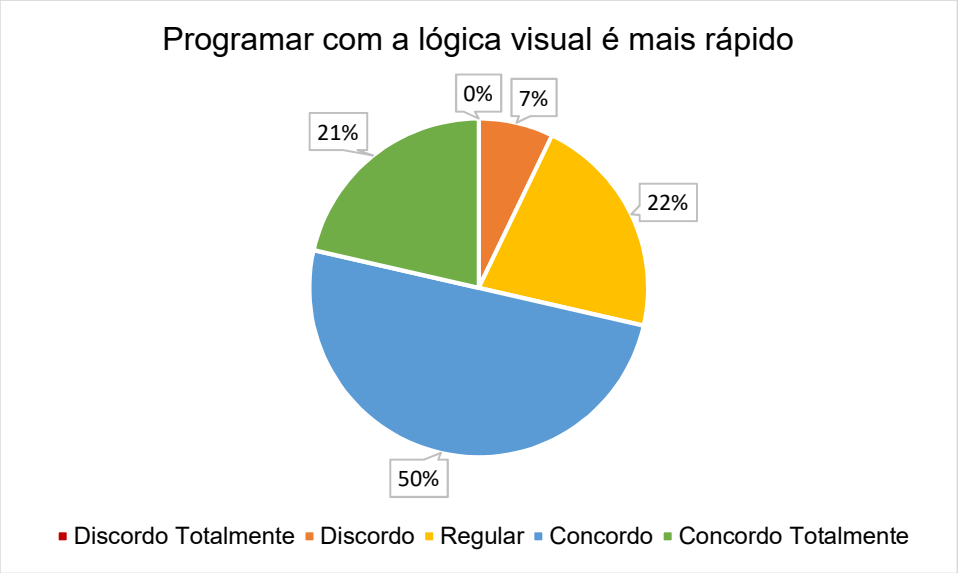
Figura 27 – Gráfico de velocidade com a ferramenta



Fonte: Do autor.

Em seguida temos o gráfico 28 com a afirmação “Programar com a lógica visual é mais rápido” que aborda a opinião do usuário em relação a velocidade em utilizar a lógica visual.

Figura 28 – Gráfico de velocidade de uso



Fonte: Do autor.

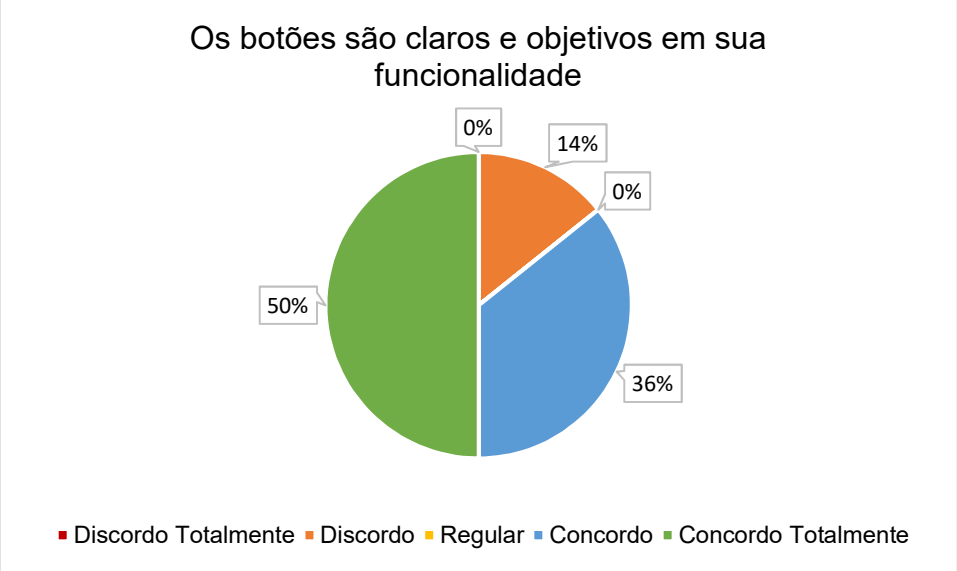
Em ambos os gráficos 27 e 28, o objetivo era obter a opinião dos alunos em relação a velocidade da ferramenta, tendo na primeira a comparação com sua ferramenta habitual e na segunda o paradigma da programação visual. Foi obtido uma porcentagem maior positiva, para a programação visual. Além disso, possível notar

por parte do avaliador que as dificuldades que atrasavam o desenvolvimento dos exercícios estavam em maioria por parte do desenvolvimento da lógica do próprio exercício e regras básicas da programação

Como último quesito do questionário observa-se os gráficos referentes a o aprendizado sem necessidade de suporte, quesito importante para uma ferramenta a ser usada em uma turma com grande número de alunos.

Primeiro temos o gráfico da afirmação “Os botões são claros e objetivos em sua funcionalidade”. A clareza e objetividade dos comandos na ferramenta foi um dos princípios durante o desenvolvimento da aplicação. Para facilitar o aprendizado sem a necessidade do professor foi dividido os comandos tipos e retirados blocos inúteis. Os resultados mostram uma aceitação maioritária deste quesito.

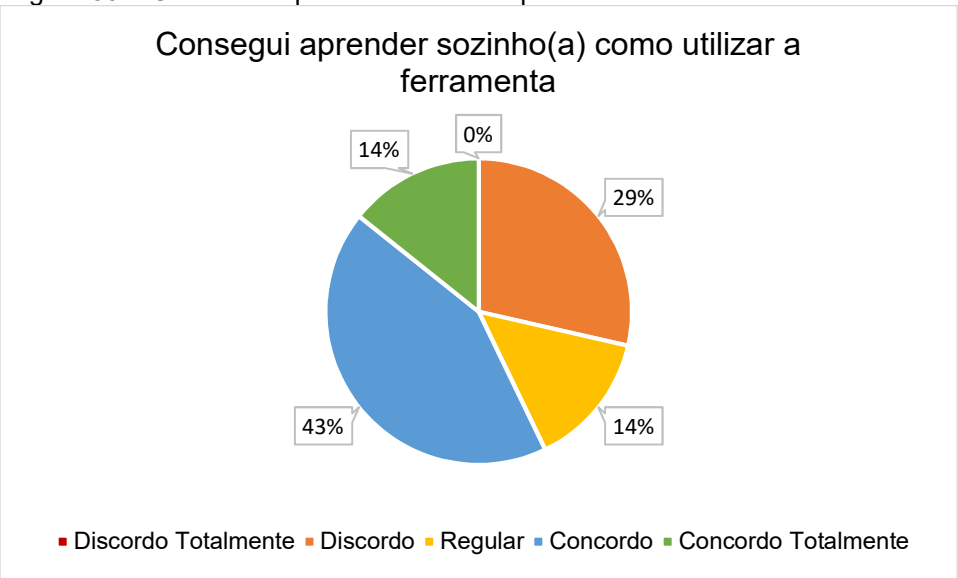
Figura 29 – Gráfico de avaliação visual da ferramenta



Fonte: Do autor.

O aprendizado sem suporte abordado no gráfico 26 reflete a dificuldade com o entendimento dos encaixes dos blocos. Durante a aplicação foi auxiliado alunos que tiveram dificuldades de entender o funcionamento de blocos que diferenciam-se da programação convencional. Como o tempo disponível para adaptação foi curto, ouve a necessidade do auxílio durante a resolução dos exercícios. Por este motivo temos os valores altos de 29% de discordância com a afirmação do gráfico.

Figura 30 – Gráfico de aprendizado sem suporte



Fonte: Do autor.

Para as perguntas referentes a resolução dos exercícios foi gerado a tabela 1 que aborda o total de respostas divididas em: perguntas, total, porcentagem positiva, não e sim.

Tabela 1 – Perguntas referentes a resolução dos exercícios com a ferramenta

Perguntas	Total	Porcentagem positiva	Não	Sim
Consegui resolver o primeiro exercício	14	0,9285	1	13
Consegui resolver o segundo exercício	14	0.7142	4	10
A ferramenta tornou mais fácil a resolução dos exercícios	14	0.7142	4	10
A ferramenta me auxiliou na escrita do código de programação	14	0.7142	4	10
A ferramenta me ajudou a desenvolver a lógica do exercício	14	0.7142	4	10

Fonte: Do autor.

Os resultados encontrados mostram que a maioria conseguiu resolver o primeiro exercício, porém a porcentagem cai para a resolução do segundo, o que pode ser justificado pelo tempo escasso para o uso da ferramenta.

As três perguntas sobre o auxílio na resolução, escrita e desenvolvimento da lógica com o uso da ferramenta obtiveram 10 respostas positivas e 4 negativas, totalizando uma porcentagem de 71% de provação.

Durante a aplicação da ferramenta também foi observado pelo pesquisador pontos fracos e fortes da ferramenta.

Os pontos fortes encontrados foram:

a) Curiosidade e motivação por parte dos alunos para o uso de resolução dos exercícios com a ferramenta;
b) Incentivo a colaboração entre os alunos para a resolução dos exercícios
c) Facilidade de adaptação a ferramenta mesmo com o curto período de tempo;
d) Aceitação positiva para a nova ferramenta;
e) Facilidade de implementação do sistema na sala de aula.

Os pontos fracos encontrados foram:

a) A aceitação, por parte da ferramenta, de nomes de variáveis não aceitados na linguagem C;
b) Um pequeno problema com a ferramenta, mas que pôde ser solucionado durante a aula;
c) Tempo escasso para a adaptação e resolução dos exercícios.

6 CONCLUSÃO

A biblioteca Blockly surgiu nos últimos anos como uma ferramenta de grande potencial educativo e tem seu foco em auxiliar o desenvolvimento da lógica de programação para quem inicia na área. Neste projeto foram estudadas modelos e ferramenta já utilizadas no processo de auxílio na aprendizagem, além de formas de ensino e linguagens adotadas, tendo como intuito, avaliar, desenvolver e aplicar uma ferramenta que auxilie no processo de aprendizagem dos alunos que iniciam na área da ciência da computação.

As dificuldades encontradas, foram durante o processo de desenvolvimento com a dificuldade de implementação de tradutores de linguagens com características distintas como a do javascript e C. A forma de adaptar uma linguagem como o javascript, que não possui tipagem de variável, para a linguagem “C” com a necessidade de tipos bem definidos, foi um dos principais problemas. Dificuldades que foram sanadas através do estudo e adaptação dos modelos de desenvolvimento de compiladores. Também houve dificuldades para encontrar bibliografias que tratam da forma de ensino em sala de aula da disciplina de algoritmo e programação nos cursos de ciência da computação.

Apesar de haver dificuldade durante o processo de desenvolvimento, o objetivo principal que consistia em aplicar a biblioteca Blockly para o auxílio no aprendizado de algoritmos para alunos de computação implementando a linguagem C como saída da programação em blocos, foi atingido com o funcionamento da ferramenta e sua aplicação em sala de aula.

Para os objetivos específicos, o primeiro foi o desenvolvimento do módulo de conversão para linguagem C da saída de código da Blockly, e foi concluído ao final do desenvolvimento com o tradutor C. O segundo e terceiro objetivos específicos consistiram no desenvolvimento do protótipo de ambiente de ensino e integração com o módulo de conversão, e foi completo com a avaliação da biblioteca e implementação do ambiente. O terceiro objetivo específico foi completo antes da aplicação com uma série de testes e correções de erros encontrados. O último objetivo específico foi a aplicação em sala de aula, da ferramenta e questionários, com a turma de quatorze alunos da disciplina de algoritmos e programação.

Os resultados obtidos concentraram-se na aplicação da ferramenta aos alunos e observação do pesquisador, de onde, obtiveram resultado positivos para a ferramenta como método de auxílio nos quesitos, satisfação e motivação do usuário, facilidade de adaptação e aprendizado da ferramenta, e por parte da resolução dos exercícios, foi observado que a maioria conseguiu resolver os exercícios, mesmo com apenas uma aula para o uso. Os resultados de satisfação e motivação, além da facilidade de aprendizado e adaptação, encontrados durante a análise dos resultados, condizem com as demais pesquisas abordando a programação visual. Além disso foram estudados alguns critérios que se referem a métricas da ferramenta, como a clareza dos botões e facilidade de encontrar os blocos, e que obtiveram resultados positivos contribuindo com o quesito de ferramenta para auxílio na aprendizagem.

Como pontos negativos da pesquisa pode ser destacado o tempo muito curto de uso com os alunos, visto que para avaliar se uma ferramenta de fato pode contribuir com o aprendizado de um aluno, é necessário uma série de aulas com a turma. Além disso, o baixo número de estudantes na turma, apenas quatorze, também prejudicou a coleta e análise dos dados, visto que com um número maior é possível obter uma média mais fiel para os gráficos.

Como forma de dar continuidade ao um estudo tão importante como o de ferramenta para o ensino e aprendizagem de algoritmos, este projeto deixa como sugestão para trabalhos futuros:

a) o uso da biblioteca Blockly aplicado com uma estratégia de ensino, como a exemplo de jogos sérios;
b) Um estudo comparando os resultados da Blockly com outra ferramenta de programação visual;
c) Aplicar outros modelos de medidas para avaliação da biblioteca Blockly;
d) Aplicação da biblioteca Blockly com estratégias educativas como a de aprendizado significativo.
e) Uma avaliação da ferramenta em relação a sua capacidade como ferramenta educativa, através da aplicação com duas turmas ao longo do semestre, uma com a ferramenta e outra sem ela.

REFERÊNCIAS

AGUILAR-SAVÉN, R. S. Business process modelling: Review and framework. **International Journal of Production Economics**, v. 90, n. 2, p. 129–149, 2004. Disponível em: <http://secure.com.sg/courses/ICT353/Session_Collateral/TOP_04_ART_03_ARTICLE_AGUILAR_Biz_Proc_Modelling.pdf>. Acesso em: 23 nov 2017.

ALEXANDROVA, S.; TATLOCK, Z.; CAKMAK, M. RoboFlow : A Flow-based Visual Programming Language for Mobile Manipulation Tasks. **Icra**, p. 5537–5544, 2015.

ALICE (1999) Learn Programming, Carnegie Mellon University- http://www.alice.org.

BARBOSA, L. DA S. **Aprendizado significativo aplicado ao ensino de algoritmos**, 2011. 57 f. Dissertação (Mestrado em Sistemas e Computação) Universidade Federal do Rio Grande do Norte. Rio Grande do Norte. 2011.

BRASIL, 2016. **Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação.** Ministério da Educação. CNE/CP 02/2012. 2016, 1–9.

BURNETT, M. M. Visual programming. **Computer**, v. 28, n. 3, p. 14, 1999. Disponível em: <http://search.proquest.com/docview/197417431?accountid=15533>. Acesso em: 23 nov 2017.

CORMEN, T. H. **Algoritmos:** teoria e prática. Rio de Janeiro: Elsevier, 2002.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. Cambridge, Massachusetts: The MIT Press, 2009.

EVARISTO, J. **Aprendendo a Programar Programando em C**. 2013. .

FALKEMBACH, G. A. M.; ARAUJO, F. V. DE. Aprendizagem de Algoritmos: Dificuldades na Resolução de Problemas. **Anais Congresso Sul Brasileiro de Computação**, p. 1–7, 2013.

FRASER, N. et al. **Blockly-A visual programming editor**, 2012. Disponível em: <https://code. google. com/p/blockly>. Acesso em: 23 abr 2017.

GUEDES, G. T. A. **UML 2:** Uma abordagem prática. 2º ed. São Paulo: Novatec, 2011.

GOOGLE. **Google for Education:** Blockly. 2016. Página Inicial. Disponível em: <https://developers.google.com/blockly/>. Acesso em: 24 nov. 2017.

SCHILDT, H. **C completo e total**. 3º ed. Makron Books do Brasil Editora Ltda, 1997.

KNUTH, D. E. **The Art of Computer Programming**. 3º ed. 1997.

KOWALTOWSKI, T. Von Neumann: suas contribuições à Computação. **Estudos Avançados**, v. 10, n. 26, p. 237–260, 1996. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-40141996000100022&nrm=iso>. Acesso em: 23 nov 2017.

KULKARNI, R.; CHAVAN, A.; HARDIKAR, A. Transpiler and it ' s Advantages. **International Journal of Computer Science and Information Technologies (IJCSIT)**, v.6, n. 2, p. 1629–1631, 2015. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.4640&rep=rep1&type=pdf>. Acesso em: 23 nov 2017.

KUK, K.; JOVANOVIC, D.; JOKANOVIC, D.; et al. Using a game-based learning model as a new teaching strategy for computer engineering. **Turkish Journal of Electrical Engineering and Computer Sciences**, v. 20, n. SUPPL.2, p. 1312–1331, 2012. Disponível em: <http://journals.tubitak.gov.tr/elektrik/issues/elk-12-20-sup.2/elk-20-sup.2-8-1101-962.pdf>. Acesso em: 22 nov 2017.

LIANG, T.-Y.; PENG, H.-T.; LI, H.-F. **A block-oriented C programming environment. 2016 International Conference on Applied System Innovation (ICASI)**. Anais... . p.1–4, 2016. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/7539741/>. Acesso em: 23 nov 2017.

LIFELONG KINDERGARTEN GROUP OF MIT MEDIA LAB. **Scratch Developers:** Scratch. 2017. Disponível em: <https://scratch.mit.edu/developers>. Acesso em: 24 nov. 2017.

LIMA JUNIOR, José Augusto Teixeira; VIEIRA, Carlos Eduardo Costa; VIEIRA, Priscila de Paula. Dificuldades no processo de aprendizagem de Algoritmos: uma análise dos resultados na disciplina de AL1 do Curso de Sistemas de Informação da FAETERJ – Campus Paracambi. **Cadernos UniFOA, Volta Redonda**, n. 27, p. 5-15, abr. 2015.

LUCRECIO, A. I. **Comparação e aplicação de diferentes ferramentas para ensino de programação para crianças**, 2016. Universidade Federal de Santa Catarina – UFSC.

MANZANO, J. A. N. G.; OLIVEIRA, J. F. DE. **Algoritmos: Lógica para desenvolvimento de programação de computadores**. 17º ed. Érica, 2005.

MATRIX TECHNOLOGY SOLUTIONS LIMITED. **What is Flowcode?** 2017. Disponível em: <https://www.matrixtsl.com/flowcode/>. Acesso em: 20 nov. 2017.

NOSCHANG, L. F.; PELZ, F.; ELIEZER, A. Portugol Studio: Uma IDE para Iniciantes em Programação. **XXXV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO – CSBC 2014**, p. 535–545, 2014. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2014/001.pdf>. Acesso em: 23 nov 2017.

OLIVEIRA, M. L. S. DE; SOUZA, A. A. DE; BARBOSA, A. F.; BARREIROS, E. F. S. Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência. **Congresso da Sociedade Brasileira de Computação – CSBC**, v. 12, p. 1–10, 2014. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2014/0022.pdf>. Acesso em: 23 nov 2017.

PAPERT, Seymour. **Mindstorms: Children, computers, and powerful ideas**. Basic Books, Inc., 1980.

PONTES, H. P. Desenvolvimento de jogos no processo de aprendizado em algoritmos e programação de computadores. **SBGames**, p. 220–228, 2013. Disponível em: <http://www.sbgames.org/sbgames2013/proceedings/cultura/Culture-28_full.pdf>. Acesso em: 23 nov 2017 .

PRICE, T. W.; BARNES, T. Comparing Textual and Block Interfaces in a Novice Programming Environment. **Proceedings of the eleventh annual International Conference on International Computing Education Research – ICER '15**. Anais... . p.91–99, 2015. Disponível em: <http://dl.acm.org/citation.cfm?doid=2787622.2787712>. Acesso em: 23 nov 2017 .

RAABE, A. L. A.; SILVA, J. M. C. DA S. Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos. XXV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, p. 2326–2337, **Anais...** 2005. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2005/003.pdf>. Acesso em: 23 nov 2017.

RIBEIRO, R. D. S.; BRANDÃO, L.; BRANDÃO, A. **Uma visão do cenário Nacional do Ensino de Algoritmos e Programação: uma proposta baseada no Paradigma de Programação Visual.** , n. Sbie, p. 26–30, 2012. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbie/2012/00127.pdf>. Acesso em: 23 mar 2017.

RODRIGUES JR, M. C. Experiências positivas para o ensino de algoritmos. **In: Workshop de Educação em Computação e Informática**, Salvador. 2004.

ROSEMANN, M. Potential pitfalls of process modeling: part A. **Business Process Management Journal**, v. 12, n. 2, p. 249–254, 2006. Disponível em: <http://www.emeraldinsight.com/doi/10.1108/14637150610657567>. Acesso em: 23 nov 2017.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. **Advanced Praise for The Unified Modeling Language Reference Manual**. 2º ed. Addison-Wesley Professional, 2005.

TIOBE. **TIOBE Index for April 2017:** April Headline: Hack programming language enters the top 50. 2017. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 22 nov. 2017.

WASIM, J.; SHARMA, S. K.; KHAN, I. A.; SIDDIQUI, J. Web Based Learning. **International Journal of Computer Science and Information Technologies**, v. 5, n. 1, p. 446–449, 2014. Disponível em: <https://pdfs.semanticscholar.org/814b/8ea740d22983e68bd269876e4a211aa45c54.pdf>. Acesso em: 23 nov 2017.

ZANINI, A. S.; RAABE, A. L. A. Análise dos enunciados utilizados nos problemas de programação introdutória em cursos de Ciência da Computação no Brasil. **Anais do XXXII Congresso da Sociedade Brasileira de Computação e XX Workshop de Educação em Informática**, 2012.

ANEXO A – PARECER CONSUBSTANCIADO DO CEP

UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: USO DA BIBLIOTECA DE PROGRAMAÇÃO EM BLOCOS BLOCKLY COMO FORMA DE AUXÍLIO AO APRENDIZADO DA DISCIPLINA DE ALGORITMOS E

Pesquisador: Luciano Antunes

Área Temática:

Versão: 2

CAAE: 96544718.2.0000.0119

Instituição Proponente: Universidade do Extremo Sul Catarinense

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 2.923.843

Apresentação do Projeto:

O projeto consiste no uso da biblioteca de programação em blocos Blockly como forma de auxílio ao aprendizado da disciplina de algoritmos e programação, pretendo auxiliar no ensino da disciplina de algoritmos no curso de computação da UNESC.

Objetivo da Pesquisa:

Aplicar a biblioteca Blockly para o auxílio no aprendizado de algoritmos para alunos de computação implementando a linguagem C como saída da programação em blocos.

Objetivo Secundário:

a) Desenvolver módulo de conversão para linguagem C da saída de código do Blockly

b) Desenvolver um protótipo de ambiente de ensino e integrar com o módulo de conversão

c) Realizar testes no protótipo desenvolvido a fim de validá-lo

d) Aplicar a plataforma desenvolvida e os instrumentos de coleta dos dados

Avaliação dos Riscos e Benefícios:

Riscos: estão diretamente ligados ao questionário para coleta das informações sendo pela perda da confiabilidade dos dados e este risco pode ser amenizado pela privacidade mantida, não sendo divulgado os dados pessoais dos alunos.

Benefícios: O auxílio no desenvolvimento do raciocínio lógico aplicado a programação e elaboração de aplicações e algoritmos.

Endereço: Avenida Universitária, 1.105
Bairro: Universitário **CEP:** 88.806-000
UF: SC **Município:** CRICIUMA
Telefone: (48)3431-2606 **E-mail:** cetica@unesc.net

UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

Plataforma Brasil

Continuação do Parecer: 2.923.843

Comentários e Considerações sobre a Pesquisa:

A pesquisa está de acordo em relação ao cumprimento das normas e orientações do CEP.

Considerações sobre os Termos de apresentação obrigatória:

Estão de acordo.

Conclusões ou Pendências e Lista de Inadequações:

O referido projeto não apresenta pendências ou lista de inadequações, estando de acordo.

Considerações Finais a critério do CEP:

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_1093609.pdf	19/09/2018 15:55:39		Aceito
Outros	Questionario.pdf	18/09/2018 21:25:31	RAUL PORTO DE SOUZA	Aceito
Parecer Anterior	PB_PARECER_CONSUBSTANCIADO_CEP_2857685_RAUL.pdf	17/09/2018 21:16:36	RAUL PORTO DE SOUZA	Aceito
Outros	ALTERACOES_DE_ACORDO_COM_O_PARECER_CONSUBSTANCIADO_DO_CEP.pdf	17/09/2018 21:15:46	RAUL PORTO DE SOUZA	Aceito
Projeto Detalhado / Brochura Investigador	Projeto_de_pesquisa.pdf	17/09/2018 20:27:39	RAUL PORTO DE SOUZA	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	Termo_de_Confiabilidade.pdf	17/09/2018 20:05:24	RAUL PORTO DE SOUZA	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TCLE.pdf	17/09/2018 20:04:29	RAUL PORTO DE SOUZA	Aceito
Declaração de Instituição e Infraestrutura	Carta_aceite.pdf	14/08/2018 22:02:36	RAUL PORTO DE SOUZA	Aceito
Folha de Rosto	Folha_de_Rosto.pdf	14/08/2018 21:58:14	RAUL PORTO DE SOUZA	Aceito

Situação do Parecer:

Aprovado

UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

Plataforma Brasil

Necessita Apreciação da CONEP:
Não

CRICIUMA, 27 de Setembro de 2018

Assinado por:
RENAN ANTONIO CERETTA
(Coordenador(a))

Endereço: Avenida Universitária, 1.105
Bairro: Universitário **CEP:** 88.806-000
UF: SC **Município:** CRICIUMA
Telefone: (48)3431-2606 **E-mail:** cetica@unesc.net

APÊNDICE A – ATIVIDADES E QUESTIONÁRIO

ATIVIDADE A SER REALIZADA E QUESTIONÁRIO

Atividade a ser realizada:

Os exercícios a seguir são dados pelo professor da disciplina e devem ser resolvidos utilizando a ferramenta de desenvolvimento em blocos Blockly adaptada para a linguagem C.

1) Segundo uma tabela médica, o peso ideal está relacionado com a altura e o sexo. Elabore um algoritmo que leia a altura e o sexo de uma pessoa, calcule e imprima seu peso ideal, utilizando as seguintes fórmulas.

Sexo	PI
1 – Masculino	**(72,7 * altura) – 58**
2 – Feminino	**(62,1 * altura) – 44,7**

2) Elabore um algoritmo que leia 5 números e imprima ao final, imprima o maior, menor, e a média dos números digitados.

O questionário a seguir possui 15 perguntas divididas em 10 questões referentes a usabilidade, experiência do usuário e 5 perguntas referentes a resolução dos exercícios.

Perguntas referentes a usabilidade e experiência de uso	Discordo totalmente	Discordo	Sem opinião	Concordo	Concordo totalmente
Estou satisfeito(a) com o uso da ferramenta	1	2	3	4	5
É fácil aprender a utilizar a ferramenta	1	2	3	4	5
Escrever o código com a ferramenta é mais rápido	1	2	3	4	5
Consegui aprender sozinho(a) como utilizar a ferramenta	1	2	3	4	5
Os botões são claros e objetivos em sua funcionalidade	1	2	3	4	5
Foi rápido encontrar os blocos que precisava	1	2	3	4	5
Prefiro utilizar a ferramenta do que outro software de programação	1	2	3	4	5
É fácil compreender o que cada bloco representa	1	2	3	4	5
Escrever o código com os blocos é mais simples	1	2	3	4	5
Programar com a lógica visual é mais rápido	1	2	3	4	5
Perguntas referentes a resolução dos exercícios com a ferramenta					
Consegui resolver o primeiro exercício				NÃO	SIM
Consegui resolver o segundo exercício				NÃO	SIM
A ferramenta tornou mais fácil a resolução dos exercícios				NÃO	SIM
A ferramenta me auxiliou na escrita do código de programação				NÃO	SIM
A ferramenta me ajudou a desenvolver a lógica do exercício				NÃO	SIM

APÊNDICE B – ARTIGO

USO DA BIBLIOTECA DE PROGRAMAÇÃO EM BLOCOS BLOCKLY COMO FORMA DE AUXÍLIO AO APRENDIZADO DA DISCIPLINA DE ALGORITMOS E PROGRAMAÇÃO UTILIZANDO A LINGUAGEM C

Raul Porto de Souza[1], Luciano Antunes[2], Louise Miron Roloff[3]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC – Brasil

[2]Professor do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC – Brasil

[3]Professora do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC – Brasil

raulporto95@hotmail.com, luc@unesc.net, professoralouise@gmail.com

Abstract. *The discipline of algorithms and programming is basic for all technological courses and still it has high failure rates. Blockly consists of a visual programming library in interleaved code blocks format that has been developed for educational purposes in the area of algorithms and programming. In the course of this project, an environment of teaching and learning programming was developed using the Blockly library and C language with metrics focused on teaching. Finally, the tool was applied to students of algorithms and programming of the computer Science course of UNESC, using systems evaluation criteria, such as usability and ease of adaptation, with the objective to explore and analyze the implementation of Blockly as an educational tool in the aid and learning of algorithms and programming.*

Resumo. *A disciplina de algoritmos e programação é básica para todos os cursos tecnológico e mesmo assim possui altas taxas de reprovação. A Blockly consiste em uma biblioteca de programação visual em formato de blocos de código interligados e que foi desenvolvida com fins educativos para área de algoritmos e programação. No decorrer deste projeto foi desenvolvido um ambiente de ensino e aprendizagem de programação utilizando da biblioteca Blockly e linguagem C com métricas voltadas para o ensino. Por fim foi aplicado a ferramenta a alunos de algoritmos e programação do curso de ciência da computação da UNESC, usando de critérios de avaliação de sistemas, como usabilidade e facilidade de adaptação, com o objetivo explorar e analisar a implementação da ferramenta Blockly como ferramenta educativa no auxílio e aprendizado de algoritmos e programação.*

1. Introdução

Como uma das matérias de maior importância nos cursos computacionais, a disciplina de algoritmos se mostra uma barreira para muitos dos alunos (BARBOSA, 2011). A dificuldade no aprendizado dessa matéria é clara em diversas pesquisas, como em Falkembach e Araujo (2013) e também em Raabe e Silva (2005) que apontam problemas, dentre os quais: os diferentes ritmos de aprendizado encontrados em grandes turmas; a impossibilidade do professor se adequar a cada aluno; a metodologia e o ambiente de realização das atividades utilizado; a resistência ao aprendizado devido a ser um conteúdo nunca visto antes e sem bases no ensino médio e básico; além da clara dificuldade na sintaxe das linguagens, apontado por Ribeiro, Brandão e Brandão (2012). Alguns estudos como o de Rodrigues (2004) citam também a questão da motivação dos alunos, que muitas das vezes foram atraídos para a área por projetos grandes como o de jogos digitais, e que então não se motivam no aprendizado dos pequenos algoritmos propostos na sala de aula.

Os diversos problemas já estudados por pesquisas em questão do aprendizado de algoritmos, levam a altos níveis de reprovação e evasão na disciplina; Barbosa (2011) nos mostra que as médias de reprovação e evasão variam de 23% a 40% em algumas das principais universidades públicas brasileiras; uma porcentagem muito alta para uma disciplina que é básica em todos os cursos de computação. De acordo com esses dados fica claro a ineficiência dos métodos tradicionais empregados no ensino de algoritmos.

Os problemas citados ainda são habitualmente encontrados nas soluções de ensino. Como forma de auxiliar na melhora dessas estatísticas este projeto, pretende usar da programação visual como uma forma de facilitar o entendimento e aprendizagem na disciplina, uma forma alternativa que abrange os problemas citados que são encontrados tanto pelos alunos, quando na dificuldade de uso e implementação. Uma alternativa de tecnologia ainda pouco explorada, mas que já vem tendo um desenvolvimento constante a décadas.

A Blockly é uma biblioteca desenvolvida pela Google em 2012 junto a equipe de desenvolvimento do instituto de tecnologia de Massachusetts, usando como base a fundamentação teórica de linguagem de blocos do MIT (PAPERT, 1980, tradução nossa). Ela disponibiliza recursos para se trabalhar com programação em conceito de blocos, tendo como entrada a programação de blocos interligados e de saída a sintaxe correta em linguagem de programação. A biblioteca Blockly é gratuita, e totalmente em Javascript. Por ser web, tem as propriedades de programação em nuvem o que a torna de fácil acesso, além de um navegador, não é necessário instalar qualquer recurso para seu correto funcionamento (GOOGLE INC, 2012, tradução nossa).

A biblioteca Bockly tem como princípios uma forma intuitiva e visual de programar (GOOGLE INC, 2012, tradução nossa). Dentre as vantagens da Blockly destacam-se ser gratuita e de código aberto, sem instalação ou *plug-in* para seu correto funcionamento; a própria linguagem visual que facilita o entendimento através da expressão sendo mostrada pelo bloco, principalmente nos códigos mais básicos. Já em códigos mais complexos a ferramenta possui a opção de exportar para uma linguagem de programação. Além dessas vantagens citadas, ela ainda tem o benefício de ser código aberto, o que a torna flexível, permitindo fazer uma melhor implementação para auxiliar o professor na sala de aula.

A escolha da biblioteca Blockly para este projeto, justifica-se por sua flexibilidade para implementações e vantagens da programação visual, que são mostradas

nas pesquisas desenvolvidas em relação ao seu potencial como ferramenta educativa, geralmente para o aprendizado de crianças, que analisam a ferramenta Blockly como forma de ensino de programação para alunos de ensino médio. Os resultados obtidos são geralmente a ampliação do raciocínio lógico e o desenvolvimento do conhecimento em programação.

A biblioteca Blockly não possui saída de código em linguagem C. Essa ausência da linguagem C na Blockly faz com que a ferramenta não possa ser usada em muitas disciplinas de algoritmos. Para solucionar esse problema, neste projeto foi adotada a linguagem C como foco da ferramenta, pois é a segunda linguagem mais utilizada de acordo com as pesquisas da TIOBE (TIOBE, 2017 tradução nossa) e é utilizada no ambiente onde foi aplicado os testes com alunos.

Como forma de avaliação do método foi aplicada a ferramenta na sala de aula com alunos de algoritmos, usando de exercícios para o ensino com a plataforma, e então coletado os dados através de questionário respondidos pelos estudantes.

2. Ensino e aprendizagem de algoritmos

A lógica de programação é a ligação direta do programador com capacidade de desenvolvimento, visto que é uma das habilidades mais importantes ao começar a trabalhar na área de programação. Algumas pesquisas afirmam ainda que seria de fundamental importância que o aprendizado de algoritmos começasse nos anos iniciais de estudo, ainda na escola, visto que os cursos exigem habilidades que não são aprendidas anteriormente e que impactam diretamente na habilidade do aluno em compreender e desenvolver algoritmos. Porém, essa disciplina ainda não é abordada na maioria das escolas, sendo um dos fatores que geram o problema no aprendizado de algoritmo. Na pesquisa realizada por Lima Junior, Vieira e Vieira (2015) foram avaliadas as dificuldades de aprendizagem dos alunos na disciplina de algoritmos dentro de unidades de aprendizagem, apontando o raciocínio lógico como o maior problema entre os estudantes (figura 1).

Gráficos de dificuldades de aprendizado
Fonte: Lima Junior, Vieira e Vieira (2015).

O raciocínio lógico é o maior problema dos estudantes de acordo com a pesquisa, os valores de problemas com capacidade de abstração e problemas de leitura e interpretação de textos também são semelhantes em porcentagem. Esses quatro pontos analisados pela pesquisa constituem a base da lógica de programação.

Ao se estudar as formas de ensino e aprendizagem de algoritmos, é difícil encontrar um padrão de ensino em todos os cursos que, apesar de ser apresentado na maioria por aulas expositivas elas ainda variam como nos mostra Raabe e Silva (2005) de acordo com as universidades e professores, em todas as formas de ensino é possível perceber sua base em propor problemas à resolução, que consiste no professor apresentando os problemas para os alunos resolverem em forma algorítmica.

De acordo com Leônidas (2011) em seu levantamento sobre o aproveitamento da disciplina de algoritmos, ele atribui dois principais problemas no aprendizado, o primeiro é o desenvolvimento do raciocínio lógico necessário para a disciplina, enquanto o segundo são subproblemas que incluem a falta de experiência semelhante com programação no ensino médio, metodologia usada pelos professores, a distância do mundo real da disciplina e dificuldades no aprendizado da linguagem de programação.

Ao falar dos cursos que ensinam algoritmos e programação, as linguagens normalmente utilizadas são C/C++, JAVA e Python, sendo base no ensino dos estudantes no início do curso, e utilizam para isso ferramentas como DEV C++ que requerem a correta sintaxe para o funcionamento do código necessitando que os estudantes lembrem da sintaxe da linguagem de programação ou eles não serão capazes de compilar seus programas em arquivos executáveis (LIANG; PENG; LI, 2016, tradução nossa).

Em alguns casos, a metodologia utilizada pelos professores no ensino de algoritmos se concentra muito em decorar a sintaxe de uma linguagem qualquer e não o próprio aprendizado da lógica com que funciona a programação. Em alguns cursos observa-se o uso de Plataformas de Desenvolvimento Integrado, do inglês *Integrated Development Environment* (IDE) como o Eclipse, Netbeans, VisualStudio, Code Blocks entre outras e que, de acordo com Noschang, Pelz e Eliezer (2014) o ensino inicial de programação deve ser focado na resolução de problemas e desenvolvimento da lógica de programação nos alunos. Tendo em vista o uso de IDEs de uso profissional que são focadas na produtividade e não na aprendizagem do usuário, o uso dessas ferramentas ao ensino nos estágios iniciais da programação são inadequadas e geram problemas ao desenvolvimento do estudante.

3. Blockly

A Blockly[6] é uma biblioteca produzida com o propósito de auxiliar o aprendizado de programação. Foi desenvolvida pela empresa Google em colaboração com o time de desenvolvimento do laboratório do MIT Media Lab (Laboratório do Instituto de Tecnologia de Massachusetts), que desenvolveu o Scratch.

A definição original da Blockly, de acordo sua criadora, é uma biblioteca que adiciona um editor de código visual para aplicações web e Android, onde utiliza blocos gráficos interligados para representar o código e conceitos como variáveis, expressões lógicas, loops entre outros, permitindo que usuários com pouco conhecimento na área apliquem os princípios da programação sem ter que se preocupar com a sintaxe das linguagens de programação (GOOGLE, 2016, tradução nossa).

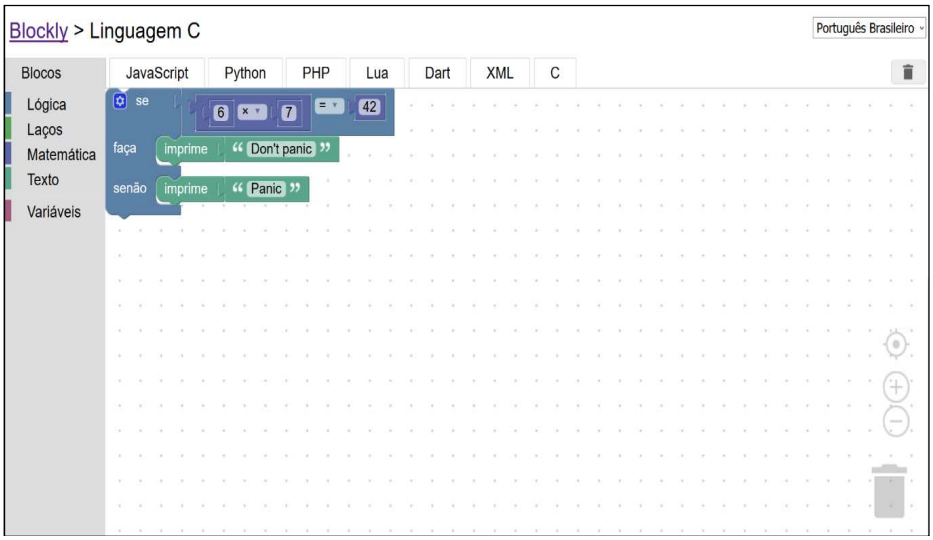
As características da biblioteca Blockly são:

f) possuir código livre;
g) possível implementação tanto para web Android e Ios;

[6] Disponível em: https://developers.google.com/blockly/ acesso em 04 Nov. 2017

h) funcionar 100% *client side*, não tendo nenhuma dependência de servidor;
i) desenvolvida totalmente em Javascript;
j) compatível com a maioria dos navegadores: Chrome, Firefox Safari, Opera e Internet Explorer.

A implementação da biblioteca consiste em uma área para arrastar os blocos e formar a sequência lógica, uma área para a seleção dos tipos de blocos disponível, os blocos de programação arrastáveis e modificáveis e um local para exibição da saída de código em uma das linguagens de programação disponíveis. A saída de código é necessária devido a Blockly não ser compilável, então ela oferece a saída nas linguagens JavaScript, Python, Php, Lua e Dart. Na abaixo tem-se o exemplo da implementação da biblioteca pelo Google. Ao lado direito da figura é possível observar os blocos separados por categorias, blocos lógicos, blocos de Loops, blocos matemáticos, blocos para textos, listas, cores, variáveis e funções. No meio da imagem fica a área para a elaboração da sequência lógica e logo abaixo a figura da lixeira para a remoção de blocos. Logo ao lado direito posiciona-se a saída do código tendo a escolha da linguagem compilável de saída, e abaixo o botão para geração do código.



Implementação da Blockly

Para a implementação do ambiente de linguagem de blocos foi utilizado a Blockly em versão Web implementando sua interface com o uso da linguagem Javascript integrado em uma página HTML5.

O desenvolvimento do módulo tradutor da linguagem, foi implementado em linguagem javascript com o propósito de evitar a geração de dependências no projeto final. O funcionamento do módulo tem como premissa a entrada do código em linguagem javascript, sua análise para as diferenças entre as duas linguagens e ao final o retorno em linguagem C para a implementação da Blockly.

Para o desenvolvimento do módulo de conversão da linguagem foi empregue as seguintes etapas:

f) avaliação dos blocos necessários para o desenvolvimento dos exercícios propostos para os alunos;
g) retirada dos blocos desnecessários para a aplicação;
h) avaliação do código de cada bloco e sua saída;
i) tradução dos códigos *tokens*;

j) tratamento de diferenças nas linguagens.

Como primeira etapa para desenvolver o tradutor foi feita uma avaliação para encontrar os blocos necessários em conformidade com os exercícios propostos. O conteúdo abordado durante a aula da coleta dos dados foi definido de acordo com a matéria atual de estudo dos alunos, sendo ela: condicionais e tomada de decisões, laço de repetição, além dos comandos básicos de leitura e impressão de dados na tela e criação e atribuição de variáveis dos tipos numéricos e vetores de caracteres.

Conforme a etapa de avaliação, foi retirado diversos blocos desnecessários para a resolução do exercício, tendo como objetivo auxiliar na adaptação dos alunos com a ferramenta. Os blocos mantidos na aplicação foram organizados dentro da caixa de seleção em Lógica, Laços, Matemática, Texto e Variáveis, tendo ainda cada tipo de blocos uma cor diferente para facilitar a visualização.

Conforme a avaliação dos blocos e suas saídas, foi desenvolvido o analisador léxico que leva em conta os nomes reservados de acordo com as saídas dos blocos. A primeira etapa para o desenvolvimento de um compilador é a análise léxica, com a varredura dos *tokens*. O procedimento de varredura dos *tokens* acontece com a leitura do código fonte e identificação das palavras reservadas do sistema, que então são mantidas em memória e reconhecidas pelo analisador (RICARTE, 2008). O reconhecimento do *token* no sistema constitui o primeiro processo para a tradução e acontece a partir do uso de expressões regular e palavras reservadas, que quando encontradas retornam uma saída em forma de cadeia de tokens.

Como segunda etapa do desenvolvimento, aconteceria em um compilador a análise sintática, porém neste projeto o objetivo é a tradução de uma linguagem que já está regrada, pois a própria biblioteca Blockly já faz a análise sintática do código e evita os erros da compilação. Por este motivo, a próxima abordagem usada para o desenvolvimento foi a tradução dos comandos *tokens* da linguagem javascript para a linguagem C. As principais diferenças encontradas nas saídas de código entre as duas linguagens estavam nos comandos de imprimir e pedir valores de acordo com forma de tratamento da variável por tipagem.

Os comandos de impressão dos blocos na linguagem javascript seguem o padrão do comando *token window.alert()* com o valor entre parênteses, sem depender se for um numeral, um texto ou variável. No C para cada tipo, numeral e variável, existe uma forma diferente de escrita, sendo utilizado o comando *printf* que é o comando de escrita em terminal da linguagem C. A cada valor que se pretende imprimir, é necessário colocar a notação do tipo de valor dentro de aspas, sendo assim necessário a definição da tipagem da variável como primeiro processo da tradução. O comando de atribuição de valor segue o mesmo padrão e necessita de definição de tipos para o recebimento (SCHILDT, 1997).

Para viabilizar a implementação de um tradutor levando em conta a diferença de tratamento das variáveis em relação a seu tipo para cada linguagem, foi necessária uma avaliação extra no código, que tem por objetivo encontrar e definir previamente todas as variáveis presentes no código fonte. Para descobrir o tipo, foi avaliado o primeiro valor de entrada que a variável recebe, sendo assim, *int* para números inteiros, *float* para números decimais e vetor de *char* para letras ou caracteres. Em situações onde não é possível definir o tipo do numeral de entrada, como no bloco de pedir valor, foi definido *float* como padrão, pois aceita tanto números inteiros como decimais.

Com a definição de tipagem de variável o comando de impressão foi traduzido para a linguagem C, que então gerou os comandos *printf()* com a notação de tipo "*%d*", para números inteiros, "*%f*" para decimais e "*%s*" para vetor de *char*. O bloco comando para pedir valor segue o mesmo princípio, e foi traduzido pela sequência de um *printf* com o texto e *scanf* para a atribuição do valor da variável através das notações de tipo.

4. Questionário e Coleta de dados

Para a etapa de coleta de dados foi desenvolvido um questionário a ser aplicado em sala de aula. Sua composição foi disposta em quinze perguntas das quais foram divididas em dez para questões referentes a usabilidade e experiência do usuário e cinco que abordavam sobre a resolução dos exercícios com a ferramenta.

Para a elaboração das questões foi utilizado as medidas presentes na ISSO 9241-210 de 2011, onde atribuem métricas para interação humano e software, levando em conta critérios de eficácia, eficiência e satisfação (ISO 9241-210, 2011, tradução nossa).

A aplicação da ferramenta foi dada em uma turma de alunos da disciplina de algoritmos da primeira fase do curso de ciência da computação da universidade do extremo sul catarinense – UNESC. Foi estimado uma turma com 15 alunos matriculados, 14 deles comparecem a aula no dia da aplicação da ferramenta.

As questões dadas para os alunos foram de exercícios já utilizados amplamente pelo professor da disciplina e consistiam em exercícios já resolvidos pela turma. O primeiro problema abrangia a matéria de laços condicionais e consistiam em gerar um algoritmo que leta a altura e sexo de uma pessoa e calcule seu peso ideal utilizando de fórmulas matemáticas de IMC. A segunda questão abrangeu o conteúdo de laços de repetição e consistiu em gerar um algoritmo para ler cinco números e ao final imprimir o maior, menor e a média dos números digitados

5. Resultados obtidos

Para a análise dos dados foi utilizado das respostas do questionário e observações encontradas durante a aplicação na sala de aula. Por parte do questionário foi gerado gráficos que exemplificam as questões de caráter qualitativo sobre a usabilidade e experiencia de uso. Cada pergunta recebeu um gráfico que mostra o nível de aceitação da afirmação por parte dos alunos. Os quatro gráficos a seguir referem-se aos quesitos: satisfação no uso, facilidade de aprendizado, velocidade relativa de uso, aprendizado sem a necessidade de suporte.

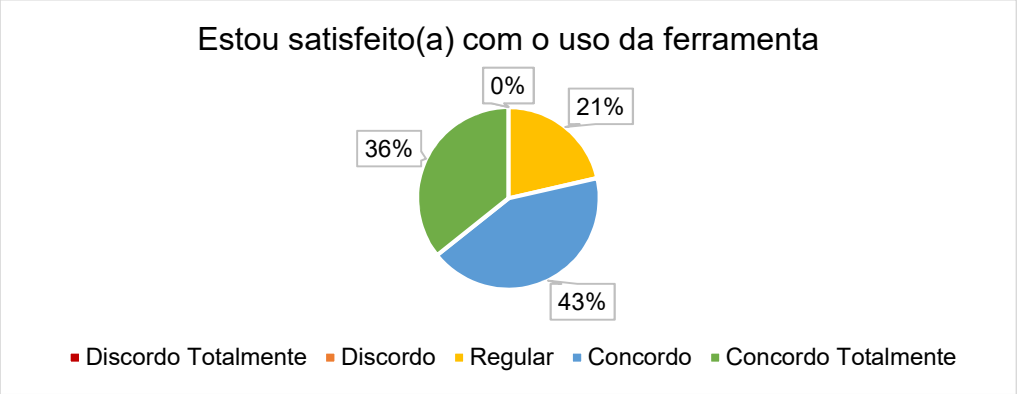


Gráfico que aborda a satisfação com o uso da ferramenta.

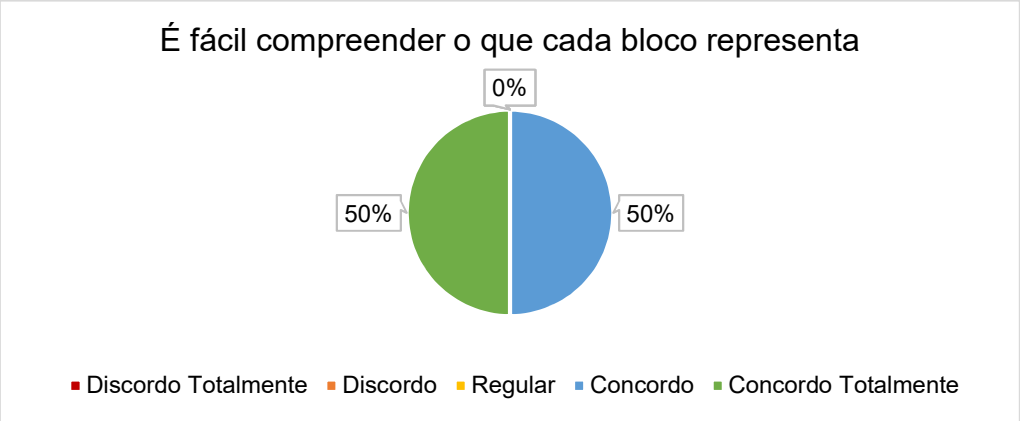


Gráfico referente a facilidade de aprendizado e adaptação com a ferramenta.

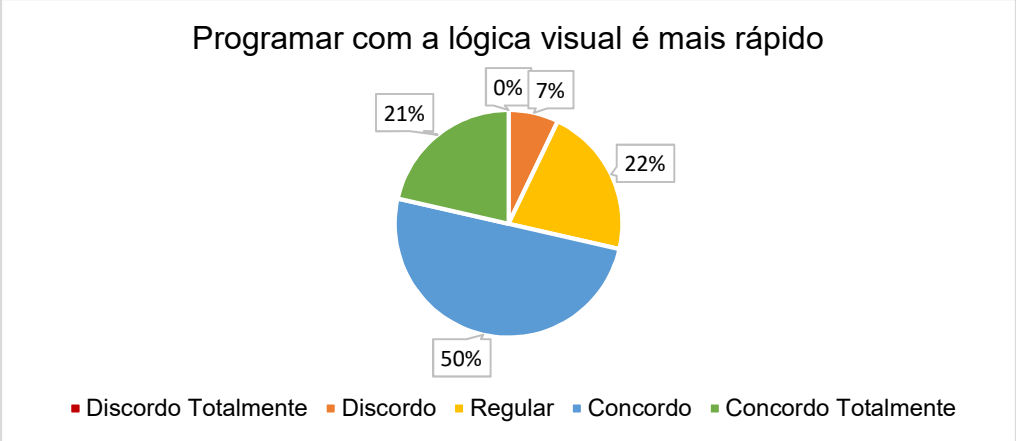


Gráfico referente a velocidade relativa do usuário com a ferramenta.

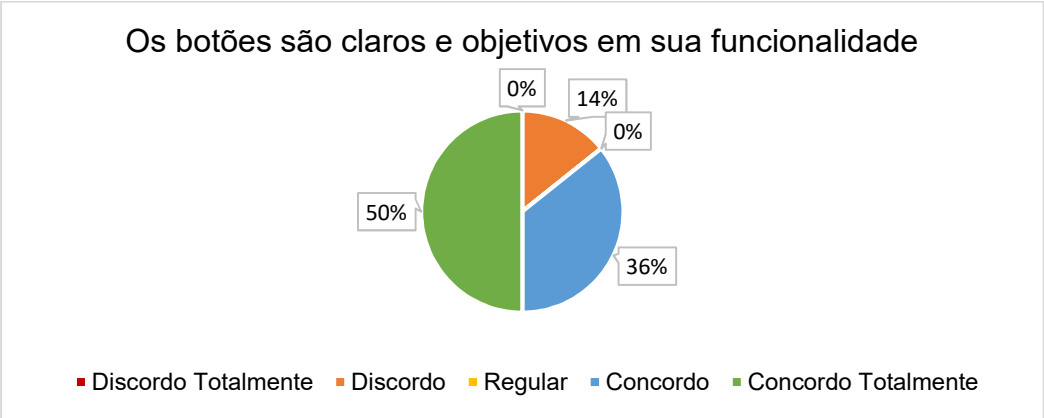


Gráfico referente ao aprendizado sem a necessidade de suporte.

Para as perguntas referentes a resolução dos exercícios foi gerado a tabela 1 que aborda o total de respostas divididas em: perguntas, total, porcentagem positiva, não e sim.

Tabela 1. Perguntas referentes a resolução dos exercícios com a ferramenta

Perguntas	Total	Porcentagem positiva	Não	Sim
Consegui resolver o primeiro exercício	14	0,9285	1	13
Consegui resolver o segundo exercício	14	0.7142	4	10
A ferramenta tornou mais fácil a resolução dos exercícios	14	0.7142	4	10
A ferramenta me auxiliou na escrita do código de programação	14	0.7142	4	10
A ferramenta me ajudou a desenvolver a lógica do exercício	14	0.7142	4	10

Durante a aplicação da ferramenta também foi observado pelo pesquisador pontos fracos e fortes da ferramenta.

Os pontos fortes encontrados foram:

f) Curiosidade e motivação por parte dos alunos para o uso de resolução dos exercícios com a ferramenta;
g) Incentivo a colaboração entre os alunos para a resolução dos exercícios
h) Facilidade de adaptação a ferramenta mesmo com o curto período de tempo;
i) Aceitação positiva para a nova ferramenta;
j) Facilidade de implementação do sistema na sala de aula.

Os pontos fracos encontrados foram:

d) A aceitação, por parte da ferramenta, de nomes de variáveis não aceitados na linguagem C;
e) Tempo escasso para a adaptação e resolução dos exercícios.

6. Considerações finais

Os resultados obtidos concentraram-se na aplicação da ferramenta aos alunos e observação do pesquisador, de onde, obtiveram resultado positivos para a ferramenta como método de auxílio nos quesitos, satisfação e motivação do usuário, facilidade de adaptação e aprendizado da ferramenta, e por parte da resolução dos exercícios, foi observado que a maioria conseguiu resolver os exercícios, mesmo com apenas uma aula para o uso. Os resultados de satisfação e motivação, além da facilidade de aprendizado e adaptação, encontrados durante a análise dos resultados, condizem com as demais pesquisas abordando a programação visual. Além disso foram estudados alguns critérios que se referem a métricas da ferramenta, como a clareza dos botões e facilidade de encontrar os blocos, e que obtiveram resultados positivos contribuindo com o quesito de ferramenta para auxílio na aprendizagem.

Como pontos negativos da pesquisa pode ser destacado o tempo muito curto de uso com os alunos, visto que para avaliar se uma ferramenta de fato pode contribuir com o aprendizado de um aluno, é necessário uma série de aulas com a turma. Além disso, o baixo número de estudantes na turma, apenas quatorze, também prejudicou a coleta e análise dos dados, visto que com um número maior é possível obter uma média mais fiel para os gráficos.

Referências

BARBOSA, L. DA S. **Aprendizado significativo aplicado ao ensino de algoritmos**, 2011. 57 f. Dissertação (Mestrado em Sistemas e Computação) Universidade Federal do Rio Grande do Norte. Rio Grande do Norte. 2011.

FALKEMBACH, G. A. M.; ARAUJO, F. V. DE. Aprendizagem de Algoritmos: Dificuldades na Resolução de Problemas. **Anais Congresso Sul Brasileiro de Computação**, p. 1–7, 2013.

GOOGLE. **Google for Education:** Blockly. 2016. Página Inicial. Disponível em: <https://developers.google.com/blockly/>. Acesso em: 24 nov. 2017.

LIANG, T.-Y.; PENG, H.-T.; LI, H.-F. **A block-oriented C programming environment. 2016 International Conference on Applied System Innovation (ICASI)**. Anais... . p.1–4, 2016. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/7539741/>. Acesso em: 23 nov 2017.

PAPERT, Seymour. **Mindstorms: Children, computers, and powerful ideas**. Basic Books, Inc., 1980.

RAABE, A. L. A.; SILVA, J. M. C. DA S. Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos. **XXV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO**, p. 2326–2337, Anais... 2005. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2005/003.pdf>. Acesso em: 23 nov 2017.

TIOBE. **TIOBE Index for April 2017:** April Headline: Hack programming language enters the top 50. 2017. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 22 nov. 2017.