UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CAMPUS FRANCISCO BELTRÃO

CURSO DE LICENCIATURA EM INFORMÁTICA

Fernando Antunes de Carvalho

# Lógica de Programação em Linguagem Visual

Francisco Beltrão, Paraná

2018

Fernando Antunes de Carvalho

# Lógica de Programação em Linguagem Visual

Trabalho de Conclusão de Curso, apresentado a Universidade Tecnológica Federal do Paraná – Campus Francisco Beltrão, como parte das exigências para a obtenção do título de Licenciado em Informática.

Orientador: Doutor. Eng. Francisco A. F. Reinaldo

Coorientador: Ms. Marlon Henrique Scalabrin

Francisco Beltrão, Paraná

2018

Fernando Antunes de Carvalho Lógica de Programação em Linguagem Visual/ Fernando Antunes de Carvalho. – Francisco Beltrão, Paraná, 2018- 40 p. : il. (alguma cor.) ; 30 cm.
Orientador: Doutor. Eng. Francisco A. F. Reinaldo
monografia – , 2018.
1. Lógica de Programação. 2. Linguagem de Programação Visual. I. Orientador. II. Universidade Tecnológica Federal do Paraná. III. Campus Francisco Beltrão. IV. Título
CDU 02:141:005.7

Fernando Antunes de Carvalho

# Lógica de Programação em Linguagem Visual

Trabalho de Conclusão de Curso, apresentado a Universidade Tecnológica Federal do Paraná – Campus Francisco Beltrão, como parte das exigências para a obtenção do título de Licenciado em Informática.

Trabalho aprovado. Francisco Beltrão, Paraná, 05 de Novenbro de 2018

Doutor. Eng. Francisco A. F. Reinaldo
UTFPR (Orientador)

Ms. Marlon Henrique Scalabrin
UTFPR (Co-orientador)

Doutor. Eng. Francisco A. F. Reinaldo
UTFPR (Presidente da Banca)

Gustavo Yuji Sato
UTFPR (Membro1 Banca)

Maicon Felipe Malacarne
UTFPR (Membro2 Banca)

Folha de Aprovação assinada encontra-se arquivada na Coordenação do Curso.

# RESUMO

Os processos de ensino-aprendizagem nas disciplinas de algoritmos e lógica de programação vem se demostrando uma tarefa árdua e desafiadora. Muitas vezes são pontos de partida para desistência. Tendo em vista tal dificuldade e os altos índices de evasão em cursos superiores logo nos anos iniciais, objetiva-se, construir um software, com foco em desenvolvimento do raciocínio lógico. Com auxilio de uma linguagem de programação visual, contendo funções e códigos pré-definidos que são representados por blocos arrastáveis. Desta maneira oferecer uma experiência pratica e visual. Disponibilizada via *Web* com intuito de facilitar o acesso e dispensar a instalação.

**Palavras-chave**: Lógica de Programação. Informática. Ensino-Aprendizagem. Linguagem de Programação Visual.

# ABSTRACT

The teaching-learning processes in the disciplines of algorithms and programming logic are proving an arduous and challenging task. They are often starting points for attention. The high index of avoidance in the upper levels early in the year, aims to build software, focusing on the development of logical reasoning. With the aid of a visual programming language, with pre-elected functions and codes that are represented by draggable blocks. In this way we offer a practical and visual experience. Available via  textit Web in order to facilitate access and to dispense with an installation.



**Keywords**:Programming logic. Computing. Teaching-Learning. Visual Programming Language.

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| CSS | *Cascading Style Sheets* |
| URL | *Uniform Resource Locator* |
| XML | *Extensible Markup Language* |
| VPL | Linguagem de Programação Visual |

# LISTA DE EXCERTOS DE CÓDIGO-FONTE

# SUMÁRIO

# 1 INTRODUÇÃO

O processo de ensino-aprendizagem de lógica de programação é uma tarefa desafiadora tanto para o professor quanto para o estudante. Absorver os conceitos de algoritmos e lógica, principalmente nos anos iniciais é uma tarefa complexa e árdua. Pode-se tonar desmotivadora, dependendo das frustrações encontradas durante o processo de inserção e adaptação no meio informacional. Contudo, professores e estudantes de diversas áreas vêm demonstrando interesse em ferramentas que possam facilitar e maximizar o processo de ensino-aprendizagem.

Segundo Santos e Costa (2006, p. 2),

> Professores de disciplinas relacionadas à programação e coordenadores de curso sentem a grande responsabilidade de buscar e aperfeiçoar sua maneira de conduzir a estrutura disciplinar e curricular dos graduandos em seus estágios iniciais da universidade. (SANTOS; COSTA, 2006).

Um algoritmo é a descrição de uma sequência de passos que devem ser seguidos com a finalidade de atingir um objetivo. A programação é a arte ou a técnica de escrever vários algoritmos de forma sistemática. Segundo Santos e Costa (2006, p. 2), "Com o intuito de produzir melhores resultados no processo de aprendizagem nessas áreas, faz-se constante a necessidade de atualização das didáticas de ensino de forma geral".

Com isso novas tecnologias proporcionam novas experiências que vem ganhando espaço no meio acadêmico. A linguagem de programação visual ou programação por demonstração, de acordo com Ferreira, Gonzaga e Santos (2010, p. 982), "[...] é uma técnica que visa aproximar o usuário cada vez mais do ambiente de programação, sem que seja necessário aprender uma linguagem específica. "Esta técnica permite que alunos aprendam a desenvolver programas ou rotinas por meio de uma ferramenta baseada em interface gráfica com exemplos passo-a-passo de como deverá ser feito, sem se preocupar tanto com a sintaxe.

De acordo com Pereira, Medeiros e Menezes (2012, p. 4),

> É necessário que os professores que trabalham com as disciplinas de algoritmos busquem soluções para minimizar o número de reprovações ou abandonos. Uma das formas de sanar esse problema é usar ferramentas ou ambientes facilitadores que provoquem um aprendizado substancial das atividades didáticas. (PEREIRA; MEDEIROS; MENEZES, 2012)

Com a expansão dos recursos tecnológicos e o aumento significativo do número de programadores, surge a necessidade de novas técnicas e instrumentos para manter atualizado o processo de inserção ao mundo da computação. A programação visual se

apresenta como um novo conceito de aprendizagem. Neste cenário um *software* educacional pode se tornar um grande aliado. Ele se destaca como uma possibilidade de inserção na prática complementando as linguagens textuais já conhecidas.

Com o auxílio de uma ferramenta visual, o aluno pode de maneira intuitiva aprender os conceitos de lógica pela manipulação de blocos que representam funções e estruturas de códigos pré-definidos. Também são apresentadas em forma textual seguindo a notação de uma sintaxe. Dessa maneira o estudante pode estar atento a implementação lógica, diminuindo a barreira entre aprender e desenvolver programas.

Para Santos e Costa (2006, p. 4), "A elaboração de uma ferramenta computacional didática deve tornar o ensino do conteúdo abordado mais prático e abrangente, de forma a despertar o interesse do aluno. "Possibilitando a formação de um profissional crítico, melhorando seu desempenho ao longo do curso influenciando na pesquisa e inovação.

## 1.1 OBJETIVOS

### 1.1.1 Objetivo Geral

O objetivo deste trabalho é facilitar o ensino e aprendizagem da lógica de programação com auxílio de linguagens visuais por meio de arrastar e encaixar blocos.

### 1.1.2 Objetivos Específicos

Os objetivos específicos são:

a) Implementar um protótipo de *software* com as seguintes funcionalidades: Arrastar e encaixar blocos que representam estruturas de códigos;

b) Apresentar o código em cinco linguagens de programação ao lado dos blocos;

c) Permitir que o programa seja executado e testado;

d) Disponibilizar via *Web*, dispensando a instalação.

## 1.2 JUSTIFICATIVA

A dificuldade na construção de um pensamento lógico computacional pode ser um obstáculo que acaba resultando em altos índices de evasão escolar. Por meio do uso de uma ferramenta visual é possível despertar o aluno a desenvolver rotinas lógicas arrastando e encaixando blocos, contendo o avanço da evasão.

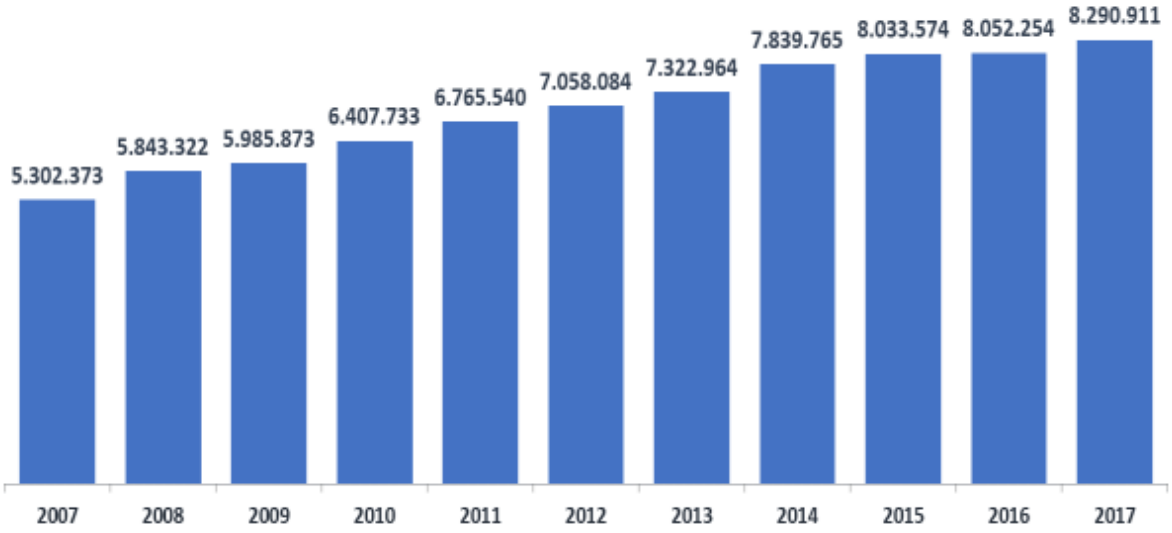
Este trabalho também contribui com várias metodologias ativas para o ensino-aprendizagem, tais como (a) ensino baseado em atividades, que permite ao professor incrementar o nível de dificuldade de acordo com a aceitação da classe, (b) em desafios,

no qual o aluno pode ser estimulado a resolver os exercícios em determinado tempo, (c) em problemas, que caracteriza um exercício a ser resolvido na prática, (d) em projetos, que divide a classe em equipes e cada equipe pode desenvolver uma parte de um projeto complexo e no final unir todas as respostas, com a mediação do professor.

# 2 FUNDAMENTAÇÃO TEÓRICA

O Plano Nacional da Educação (PNE), aprovado no Brasil em 2014 traça algumas metas que vão da educação infantil ao ensino superior. Dentre elas, elevar a taxa bruta de matrícula na educação superior para 50% e a taxa líquida para 33% da população. Conforme pode ser visto na Figura 1, esse número vem registrando uma média de crescimento em torno de 3% anual a partir de 2016, chegando próximo dos 8,3 milhões de matriculas.

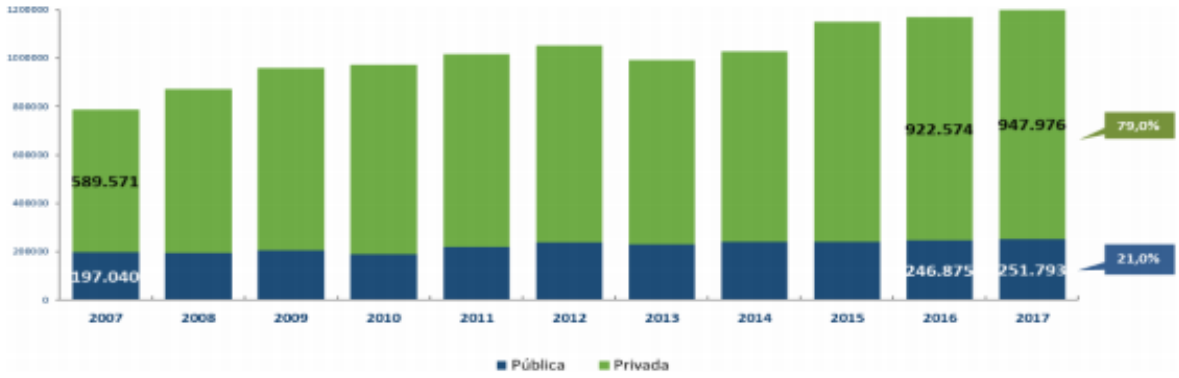
Figura 1 – Número de Matrículas na Educação Superior – 2007-2017




Fonte:DEED (2017)

Porém, observando a Figura 2, constata-se que, os índices de concluintes ainda são bastante inferiores se comparados com os ingressantes. Em 2016 foi registrado 8.052,254 de alunos matriculados e 1.169,449 alunos concluintes, gerando 6.882,805 de alunos em evasão. Segundo dados do DEED (2017, p. 22), "O percentual de concluintes nas instituições publicas é de 27,8%"se comparados com o numero de ingressantes. Esse problema de evasão no ensino superior não é uma exclusividade do Brasil, ela vem afetando os resultados de todo sistema educacional.

Figura 2 – Número de Concluintes em Cursos de Graduação – 2007-2017




Fonte:DEED (2017)

## 2.1 Desafios do ensino superior

O estudo da evasão escolar no ensino superior vem sendo uma tarefa complexa e desafiadora para pesquisadores de diversas áreas. Pois, são inúmeros os motivos que podem levar um aluno a tomar tal decisão. Segundo Ribeiro, Brandão e Brandão (2012, p. 1), "Existem diversos gatilhos para provocar a evasão, dentre eles as dificuldades encontradas em cursos de Computação, sejam elas relacionadas à lógica ou à sintaxe da linguagem associada". Também questões sócio econômicas podem influenciar.

De acordo com Principal (1988, p. 21).

> Grande parte da evidência empírica mostra que evasão escolar e pobreza são, intimamente, ligadas e que trabalho infantil prejudica a obtenção de melhores níveis educacionais. Pode-se argumentar que a indisponibilidade de serviços educacionais de qualidade e a falta de percepção acerca dos retornos futuros levem o aluno ao trabalho precoce e aos baixos níveis educacionais. Outros estudos privilegiam os efeitos de restrições de renda e de crédito como causas de desvios da trajetória individual ótima de longo prazo. (PRINCIPAL, 1988)

Segundo Rodrigues, Brackmann e Barone (2015, p. 98), “ Muitos deles chegam à graduação com pouca base em matemática, por exemplo, por conta de deficiências nos níveis fundamental e médio. Com o tempo, esse aspecto diminui o interesse do aluno pela área. ” O abandono sem conclusão do curso representa um grande desperdício de esforços, tempo e recursos financeiros, principalmente em instituições publicas onde constata-se maiores índices. Os quais causam impactos negativos para sociedade deixando uma sensação de perda.

De acordo com Giraffa e Móra (2013, p. 2)

> As perdas de estudantes que iniciam, mas não terminam seus cursos são desperdícios sociais, acadêmicos e econômicos. No setor público, são recursos públicos investidos sem o devido retorno. No setor privado, é uma importante perda de receitas. Em ambos os casos, a evasão pode ser tomada como fonte de ociosidade de professores, funcionários, equipamentos e espaço físico.(GIRAFFA; MÓRA, 2013)

Para Rodrigues, Brackmann e Barone (2015, p. 98), "Apesar da existência de um quadro extremamente favorável para os profissionais da área de Tecnologia da Informação, existe uma realidade desanimadora: a escassez da mão de obra no referido mercado." isso demonstra que cursos relacionados a informática não estão suprindo a demanda de profissionais e o crescimento do mercado.

De acordo com Rodrigues, Brackmann e Barone (2015, p. 98) "Sobre esse aspecto, Sergio Sgobbi, diretor de Educação e Recursos Humanos da Brasscom, assegura que a carência por profissionais de TI pode ser justificada pela desistência dos cursos superiores no Brasil."Com base nesses dados podem ser associados os números que se referem a evasão e a falta de profissionais formados no setor tecnológico, com algumas dificuldades encontradas no processo de ensino-aprendizado.

## 2.2 Dificuldades de Aprendizagem

Algumas das dificuldades encontradas estão relacionadas com o fato de alunos não compreenderem conceitos abstratos, devido a forma como os conteúdos são abordados ou até mesmo a ferramenta utilizada. Podendo também relacionar, dificuldades em resolução de problemas matemáticos, com dificuldades em programação Gomes et al. (2012, p. 439). Também destaca os "[. . . ] métodos de ensino desadequados à aprendizagem da programação e a conotação negativa associada a estas disciplinas".

De acordo com Neto, Castro e Júnior (2006, p. 527),

> A aprendizagem de programação pressupõe a transposição de alguns patamares de expertise até que o sujeito seja um programador experiente. Esses patamares, embora cogitados por muitos pesquisadores, ainda não são conhecidos, e por isso, ensinar programação não poderia seguir uma metodologia linear. (NETO; CASTRO; JÚNIOR, 2006)

Existem diversas opiniões sobre a causa de tal problema. Segundo Gomes, Henriques e Mendes (2008, p. 94), "[. . . ] Em função das quais, têm surgido diferentes ferramentas com o propósito de minimizar essas dificuldades. Embora a utilização dessas ferramentas mostre impactos positivos, as taxas de evasão e repetência continuam altos. "Contudo não basta apenas inserir ferramentas de apoio. É necessário quebrar a barreira do tradicionalismo, os quais faziam sentido quando o acesso a informação era restrito.

Esse problema pode se agravar ainda mais quando o professor tenta explicar um conteúdo sem exemplificar de alguma forma, seja utilizando uma linguagem de programação ou uma ferramenta que demonstre o conceito. Segundo Borges (2000, p. 3), "Não é motivador para os alunos desenvolver programas sem uma interface gráfica mais elaborada, uma vez que muitos deles já trabalham com um ambiente gráfico (como o Windows)".

Borges (2000, p. 3), "Define uma nova proposta para uso em um curso de introdução a programação, todo desenvolvido em ambiente gráfico e visual. Os primeiros problemas são relacionados com o mundo real e, pouco a pouco, vão sendo adaptados. "Desta forma a introdução não começa com aulas de sintaxe ou explicando a lógica verbalmente. Desde o início os alunos resolvem exercícios que apresentam resultados na tela. A sintaxe seja em uma linguagem de programação ou em *Portugol*, é explorada posteriormente.

Para Gomes, Henriques e Mendes (2008, p. 96), "A primeira dificuldade, frequentemente não percebida pelos alunos, diz respeito à compreensão do problema, pois muitas vezes os alunos 'saltam' para a fase de *codificação*".

> Como pensamos que o problema principal reside na incapacidade de os alunos resolverem problemas, criou-se uma aplicação, cuja preocupação principal era fornecer um ambiente onde os alunos não apenas compreendessem as diversas fases de um algoritmo já concebido, mas sobretudo que permitisse que o aluno concebesse, testasse, experimentasse, alterasse e corrigisse os seus próprios algoritmos. Contudo, as avaliações efetuadas demonstraram que a abordagem utilizada não é suficiente para contemplar todos os alunos. Por um lado, é um ambiente que não promove de igual forma todos os estilos de aprendizagem, mas antes favorece os alunos marcadamente visuais. (GOMES; HENRIQUES; MENDES, 2008)

Existem várias maneiras de se ensinar lógica de programação, a mais simples é com papel e caneta escrevendo um pseudocódigo. Porém impossibilita a visualização dos resultados e depende da correção do professor pois o aluno só consegue testar com teste de mesa. Uma maneira mais eficaz para explorar o desenvolvimento de atividades deste gênero são *software* didáticos desenvolvidos para tal finalidade.

## 2.3 Linguagem Visual

Tendo em vista a importância de um aprendizado significativo e as dificuldades citadas a cima, sobre lógica de programação, para torna-la um processo mais eficiente e produtivo, professores e alunos podem contar com apoio de algumas ferramentas baseadas em Linguagem Visual. As quais Segundo Ribeiro, Brandão e Brandão (2012, p. 2), "[...] permite aos alunos construir seus algoritmos interagindo com elementos visuais. Os elementos representam blocos de controle similares a estruturas do tipo “for”, “while”,

“if-then-else” e variáveis. "Com isso, quebra a barreira entre aprender a lógica e aprender a lógica com a sintaxe.

Para Trindade (2015, p. 66),

> A Linguagem de Programação Visual, é um novo conceito em aprendizagem de programação de computadores. Com a expansão do uso dos recursos tecnológicos, mais pessoas começarem a programar computadores e várias linguagens de programação foram e estão em desenvolvimento. A VPL se apresenta como uma iniciação ao ato de programar, não substituindo as linguagens de programação como já conhecida nas linguagens textuais, mas como um recurso simples e rápido para o desenvolvimento de programas. (TRINDADE, 2015)

Existem diversas ferramentas destinadas a facilitar o ensino de lógica e de programação, porém de acordo com Pereira, Medeiros e Menezes (2012, p. 3), "Mesmo com diversos modelos, não há um *software* ideal, pois seu uso depende da metodologia do professor e do nível de conhecimento e velocidade de aprendizado da turma. "Dentre as ferramentas com intuito de facilitar a aprendizagem com auxílio de VPL, pode ser citado o Scratch.

Segundo Pereira, Medeiros e Menezes (2012, p. 5),

> A programação do Scratch é feita através de blocos de comandos que são encaixados uns aos outros, formando a sequência de comandos que se deseja. Os blocos são concebidos para se encaixar apenas de uma única forma, fazendo sentido sintaticamente, não ocorrendo, assim, erros de sintaxe. É uma maneira de trazer conceitos de informática de alto-nível para seus usuários, pois os comandos presentes nos blocos são praticamente os mesmos. (PEREIRA; MEDEIROS; MENEZES, 2012)

A linguagem de programação visual aplicada em disciplinas iniciais, conseguem suprir a falta de resultados imediatos onde muitas vezes deixam duvidas sobre os conceitos explanados pelo professor. Podem ser utilizadas para alavancar e despertar o interesse pela programação. De acordo com Trindade (2015) "Deve-se observar ainda que pela facilidade em sua utilização até o público infantil pode utilizá- la como ferramenta de apoio para as demais disciplinas curriculares".

Geraldes (2014, p. 107) afirma que,

> O aluno, ao usar as linguagens de programação, transforma seu conhecimento em procedimentos, ou seja, descreve todos os passos necessários para atingir um certo objetivo, para atingir a resolução de um certo problema; em suma, está “ensinando” o computador a atingir um objetivo através de um programa. (GERALDES, 2014).

Segundo Moran (2015, p. 16), "Essa mescla, entre sala de aula e ambientes virtuais é fundamental para abrir a escola para o mundo e para trazer o mundo para dentro da escola. "É possível de prever e planejar as atividades de uma forma mais aberta, utilizar

uma linguagem mais familiar, proporcionar um ensino moderno, desta forma fomentar o interesse dos alunos para desenvolvimento com criatividade.

# 3 MATERIAIS E MÉTODOS

## 3.1 Materiais

Para desenvolvimento do *software* foi utilizado os materiais descritos na Tabela 1.

Tabela 1 – Materiais utilizados no desenvolvimento do sistema

| **Material** | **Disponível em** | **Aplicação** |
|---|---|---|
| Google for Education Blockly | https://developers.google. com/blockly/ | Biblioteca em JavaScript destinado a criação de editores de programação visual, utilizado para criar os blocos e a representação dos mesmos em códigos. |
| JavaScript | https://www.javascript. com/ | Linguagem de programação orientada a objetos para *Web* utilizada para representar os blocos e renderiza-los em códigos. |
| CSS | https://www.w3.org/css/ | Linguagem de estilização utilizada na aparência/estilo da página web por meio de folhas de estilo em cascata. |
| HTML | https://www.w3.org/ html/ | Linguagem de marcação de textos, utilizada para desenvolvimento da interfaces junto com CSS. |
| Sublime Text | https://www.sublimetext. com/ | Editor de texto utilizado para desenvolvimento *Web*. |

Continua na página seguinte

**Tabela 1 – na página anterior**

| Material | Disponível em | Aplicação |
|---|---|---|
| Apache HTTP Server | https://www.apache.org/ | Servidor HTTP utilizado para disponibilizar paginas e todos os recursos no navegador. |

O Software permite a visualização de códigos em cinco linguagens de programação distintas, conforme descritos na tabela 2.

Tabela 2 – Linguagens disponíveis para visualização

| Linguagem | Descrição |
|---|---|
| JavaScript | Nasceu em 1995 junto com o surgimento dos primeiros navegadores para internet, com intuito de executar *scripts* e interagir com o usuário. É uma linguagem orientada a objetos, com foco no desenvolvimento *web*, utilizada para criar um comportamento dinâmico e interativo com o usuário, através de animações, mensagens e manipulações de elementos na interface. |
| Python | Criado no ano de 1989 é considerada uma linguagem de alto nível, suporta o paradigma de orientação a objetos e também procedural, tornando ela uma linguagem adaptável para o ensino de programação, desde a introdução até aplicações mais avançadas. Voltada para o desenvolvimento *web* e *internet*, utilizada frequentemente para desenvolvimento de softwares. |
| PHP | Disponibilizada no ano de 1995, é utilizada para o desenvolvimento *back-end* (do lado do servidor), foi uma das primeiras linguagens a interagir com documentos em HTML dispensando arquivos externos para processamento de dados na *internet*. |
| Lua | Foi criada em 1993 por desenvolvedores da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), é uma linguagem clara e de fácil aprendizado, utilizada para unir códigos de diversas linguagens, softwares embarcados, desenvolvimento de jogos, controle de robôs e prototipagem rápida. |
| Continua na página seguinte | |

**Tabela 2 – na página anterior**

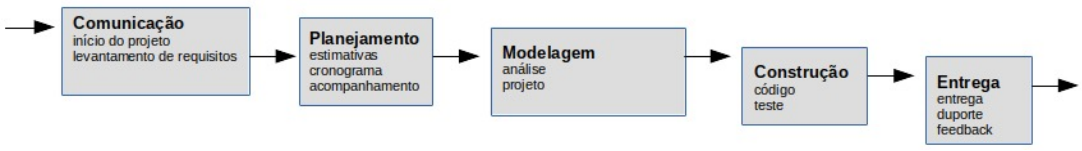
| Requisito | Descrição |
|---|---|
| Dart | Desenvolvida pela Google em 2011 com intuito de se tornar a linguagem principal dos navegadores. É uma linguagem de *script* com foco em desenvolvimento para *web*, permite otimizar a interação com cliente criando interfaces bonitas de alta qualidade. |
| XML | É uma linguagem de marcação utilizada para troca de dados. Neste software o XML é utilizado para representar a montagem dos blocos através de uma marcação, também permite a reconstrução dos blocos. |

## 3.2 Métodos

A proposta da ferramenta é utilizar linguagem de programação visual para contribuir com o processo de ensino-aprendizado em disciplinas de lógica de programação. Portanto, o desenvolvimento do *software* foi baseado em um modelo de ciclo de vida em cascata, o qual de acordo com Zucule (2017, p. 33), "[. . . ] os principais subprocessos são executados de acordo com uma sequencia , que permite demarcar pontos de controle bem definidos. Estes pontos de controle facilitam muito a gestão de projetos, "fazendo com que o processo se torne confiável.

O modelo cascata também conhecido como modelo ciclo de vida clássico, sugere que seja utilizado um tratamento sequencial e sistemático, conforme pode ser visto na Figura 3. Este modelo é recomendado para projetos que tenham requisitos iniciais bem definidos e compreendidos, quando o desenvolvimento possa fluir de forma linear da comunicação até a entrega.

Figura 3 – Modelo de ciclo de vida em Cascata




Fonte: O Autor.

## 3.3 Ciclo de Vida

O ciclo de vida teve seu inicio com a formalização da ideia junto com o projeto de Trabalho de Conclusão de Curso (TCC), passando pelas etapas abaixo.

### 3.3.1 Comunicação

A comunicação inicialmente aconteceu entre o orientador e o acadêmico, nessa etapa o acadêmico apresentou as ideias e os objetivos para o orientador, foram discutidas as possibilidades e a viabilidade do projeto.

Ficou definido que seria desenvolvido uma ferramenta que gerasse códigos através de blocos, seriam arrastáveis e encaixados, também ficou definido que os usuários poderiam testar e visualizar seus códigos.

Com objetivo de produzir algoritmos sem precisar escrever os códigos, foi levantado os requisitos funcionais necessários para o desenvolvimento do sistema apresentados na Tabela 3 .

Tabela 3 – Levantamento de Requisitos

| **Requisito** | **Descrição** |
|---|---|
| *Software* | Desenvolver um programa que permita o usuário arrastar e encaixar blocos, gerar códigos a partir da montagem dos mesmos que serão apresentados em uma tela ao lado da tela de montagem. O sistema também deve permitir que esses códigos sejam executados disparando mensagens com os resultados. |
| Blocos | O conjunto de blocos deve ficar posicionado no canto direito da tela, os blocos devem ficar organizados dentro de um menu dividido em categorias sendo elas operações lógicas, laços de repetição, matemática, texto, listas, cores, variáveis e funções. |
| Representação dos blocos | Cada bloco deve representar uma estrutura de código pronta, sempre que selecionado e encaixado o sistema deve unir os dois códigos e apresentar na tela de visualização. |
| Encaixe dos blocos | Os blocos devem ser encaixados seguindo um conjunto de regras lógicas, não sendo permitido encaixar estruturas incorretas, como exemplo para um bloco que representa se (condição) então faça, essa condição deverá ser uma comparação (x > y), não será permitido encaixar um bloco de texto sozinho por exemplo. |

Continua na página seguinte

**Tabela 3 – na página anterior**

| Requisito | Descrição |
|---|---|
| Execução de Códigos | O sistema deve ter um botão que possibilite o usuário executar e testar o código gerado, como a plataforma é *Web*, os resultados devem ser apresentados via mensagem de alerta disparadas pelo navegador. |
| Visualização em Códigos | O sistema deve ter uma tela exclusiva para apresentação dos códigos, essa tela irá apresentar o resultado da montagem de acordo com a sintaxe da linguagem que o usuário escolher podendo ser alterado a qualquer momento. |

### 3.3.2 Planejamento

Nesta etapa definimos um cronograma com os prazos para desenvolvimento do projeto Tabela 4.

Tabela 4 – Cronograma

| **Etapas** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| Levantamento dos Requisitos | | | | | | | | | | |
| Configuração da Biblioteca | | | | | | | | | | |
| Visualização em Códigos | | | | | | | | | | |
| Regras de encaixe | | | | | | | | | | |
| Execução dos Códigos | | | | | | | | | | |
| Automatização na visualização | | | | | | | | | | |
| Revisão final no desenvolvimento | | | | | | | | | | |
| Entrega | | | | | | | | | | |

Fonte: O Autor.

### 3.3.3 Modelagem

Nesta etapa foi analisado os requisitos de sistema levantados anteriormente os quais serviram de embasamento para construção do *software*.

Foi definido algumas regras de encaixa afim de evitar a construção de um código incorreto, sendo elas:

a) Não será permitido encaixar um bloco de operações matemáticas (somar, subtrair, multiplicar, dividir e exponencial) com o bloco de condição lógica (se condição então faça).

b) Blocos da aba matemática não poderão ser encaixados diretamente sem a presença de um bloco de lógica, em blocos de laços de repetição.

c) Só será permitido encaixar o bloco Imprime referente a aba texto nos blocos das demais abas, o restante dos blocos da aba texto devem ser encaixados no Imprime pois o mesmo cria a estrutura de impressão.

d) Um bloco de cor só será permitido encaixar em um bloco de texto.

e) O bloco Encerra laço só poderá ser utilizado dentro de um laço de repetição.

### 3.3.4 Construção

Para construção deste projeto foi necessário realizar a instalação e configuração do editor de texto *Sublime* e servidor *HTTP Apache*. Possibilita assim escrever os códigos fontes e visualizar os resultados no navegador.

Para inciar o desenvolvimento da codificação foi necessário instalar e configurar a biblioteca Google Blockly para JavaScript, a partir dela foi possível fazer a chamada dos blocos bem como o processo de renderização em códigos. Essa biblioteca permite que os usuários arrastem blocos e gerem códigos.

A configuração do Google Blockly consiste em fazer a chamada e alocação dos objetos na tela do navegador, dessa forma inicialmente foi incluído o conjunto principal de blocos nativo desta biblioteca, fazendo sua chamada no arquivo *index.html* o qual representa a tela principal do *software*, conforme Cód. 3.1.

Também ficou definido em quais linguagens de programação será exibido os códigos, Cód. 3.2, ficando 6 linguagens de programação distintas, sendo elas JavaScript, Python, PHP, Lua, Dart e XML.

Código 3.1 – Chamada do conjunto de blocos principal

```
<head>
  <meta charset="utf-8">
  <meta name="google" value="notranslate">
  <title>Logica em Blocos</title>
  <link rel="stylesheet" href="style.css">
  <script src="/storage.js"></script>
  <script src="../../blockly_compressed.js"></script>
  <script src="../../blocks_compressed.js"></script>
```

```
  <script src="../../javascript_compressed.js"></script>
  <script src="../../python_compressed.js"></script>
  <script src="../../php_compressed.js"></script>
  <script src="../../lua_compressed.js"></script>
  <script src="../../dart_compressed.js"></script>
  <script src="code.js"></script>
</head>
```

Código 3.2 – Chamada das linguagens

```
<td id="tab_javascript" class="taboff">JavaScript</td>
            <td class="tabmin"> </td>
            <td id="tab_python" class="taboff">Python</td>
            <td class="tabmin"> </td>
            <td id="tab_php" class="taboff">PHP</td>
            <td class="tabmin"> </td>
            <td id="tab_lua" class="taboff">Lua</td>
            <td class="tabmin"> </td>
            <td id="tab_dart" class="taboff">Dart</td>
            <td class="tabmin"> </td>
            <td id="tab_xml" class="taboff">XML</td>
            ...
```

A tela principal do *software* foi dividida em três partes, sendo elas área dos blocos, montagem dos blocos e de visualização de códigos conforme Figura 4.

Figura 4 – Tela Principal do *software*


Fonte: O Autor.

O comportamento do sistema foi definido dentro do arquivo code.js também instanciado no *index.html*, neste aquivo foi importado uma lista de idiomas para tradução

da interface do sistema, ao iniciar o programa ficou definido o idioma português Brasileiro (pt-br), conforme Cód. 3.3, podendo ser redefinido pelo usuário.

Código 3.3 – Função retorna 'pt-br' quando o idioma está indefinido

```
Code.getLang = function() {
  var lang = Code.getStringParamFromUrl('lang', '');
  if (Code.LANGUAGE_NAME[lang] === undefined) {
    lang = 'pt-br';
  }
  return lang;
};
```

Ao abrir o programa também ficou inicializado a visualização de código na linguagem JavaScript Cód. 3.4, foi necessário esperar 100 milissegundos para que a biblioteca estivesse carregada e inicializada, após esse tempo a rotina chama o evento *click* da aba JavaScript, simulando que o usuário estivesse clicado em gerar código nessa linguagem:

Código 3.4 – Metodo setando

```
  Code.workspace.addChangeListener(Code.renderContent);
  window.setTimeout(Code.importPrettify, 1);
  window.setTimeout(Code.tabClick, 100, "javascript");
```

Para que os códigos ficassem visíveis e sem sobreposição na tela, foi necessário a cada evento de *click* na seleção de linguagem, passar uma propriedade para esconder as demais, conforme Cód. 3.5, quando o conteúdo selecionado for Blockly.Python, o método abaixo trata de esconder os conteúdos de JavaScript, PHP, Lua, Dart e XML ficando visível apenas o Python.

Código 3.5 – Setando Visibilidade escondido no Click das linguagens

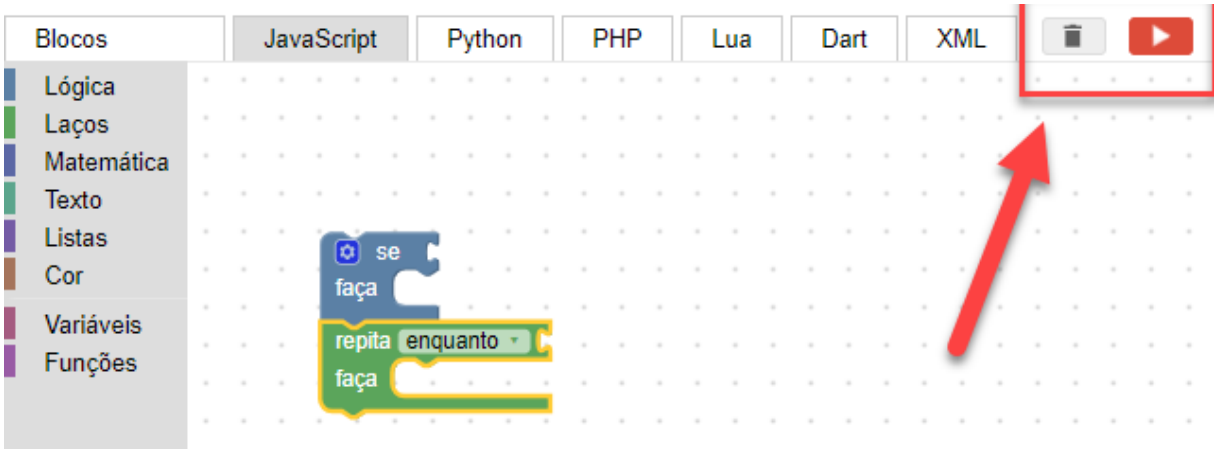
```
...
  }
    else if (content.id == 'content_python') {
    document.getElementById('content_php').style.visibility =
       'hidden';
    document.getElementById('content_dart').style.visibility =
       'hidden';
    document.getElementById('content_lua').style.visibility =
       'hidden';
    document.getElementById('content_javascript').style.visibility
       = 'hidden';
    document.getElementById('content_XML').style.visibility =
       'hidden';
```

```
    Code.attemptCodeGeneration(Blockly.Python, 'py');
...
```

O *software* conta com dois botões um para deletar com o formato de uma lixeira e outro para executar com formato de uma seta, conforme pode ser visto na Figura 5. Ao clicar em deletar o sistema exibe uma mensagem solicitando confirmação para deletar todos os blocos, Figura 6. Ao clicar em executar o *software* tenta executar a rotina caso esteja sem erros os resultados serão aprestados em forma de mensagem Figura 7, caso ocorra algum erro não terá diagnóstico.

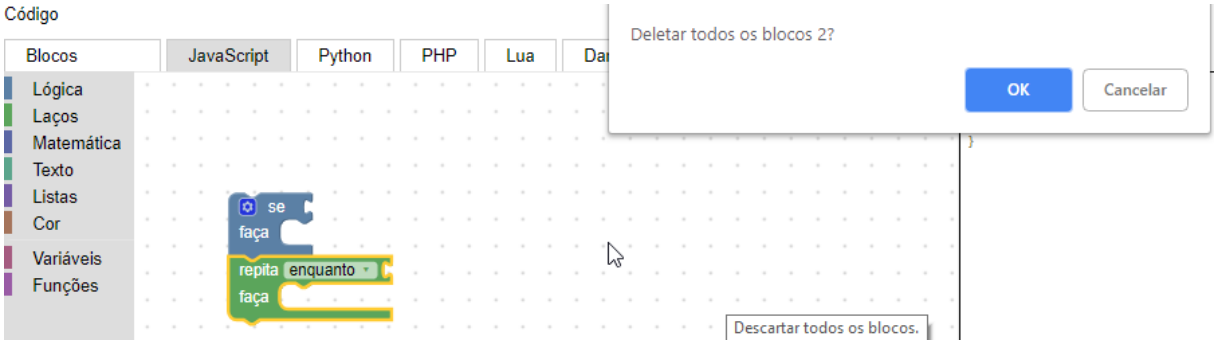
Para finalizar a construção do *software* foi desenvolvido um tratamento para que sempre ao editar um bloco o sistema já identifique a alteração e atualize a linguagem corrente na tela de visualização.

Figura 5 – Tela Principal do *software*, Botões deletar, executar



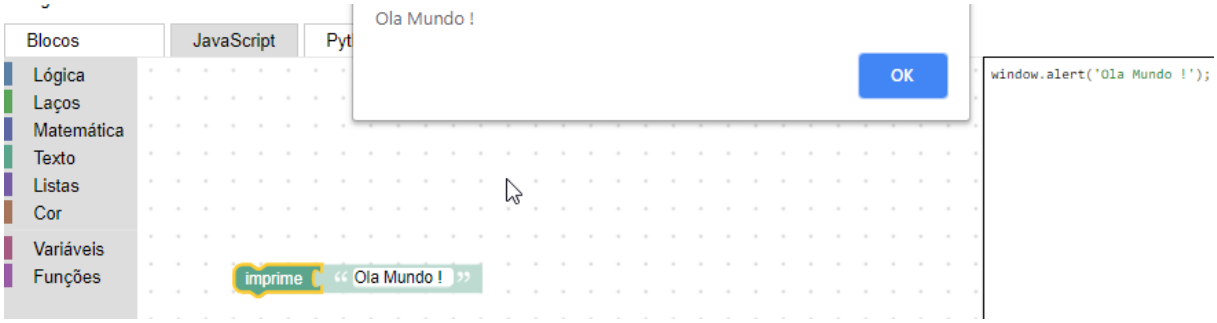

Fonte: O Autor.

Figura 6 – Tela Principal do *software*, Mensagem de confirmação antes de deletar




Fonte: O Autor.

Figura 7 – Tela Principal do *software*, Executando o código




Fonte: O Autor.

### 3.3.5 Entrega

Com intuito de facilitar o acesso e dispensar configuração de ambiente o *software* foi desenvolvido e entregue na plataforma *Web*.

# 4 RESULTADOS

Utilizando os materiais descritos na Tabela 1. Na seção 4.1, são apresentados os exercícios na qual a ferramenta foi testada

Na Figura 8 pode ser visto a tela principal do Software. No lado esquerdo da tela fica o conjunto de blocos encapsulado em um menu, no lado direito se encontra a tela de visualização dos códigos e no meio a tela de montagem, também um menu para selecionar em qual linguagem deseja visualizar os códigos, um botão para deletar todos os blocos montados e outro para executar e testar os resultados da montagem, mais uma caixa para seleção do idioma.

Figura 8 – Tela Principal do Software


Fonte: O Autor.

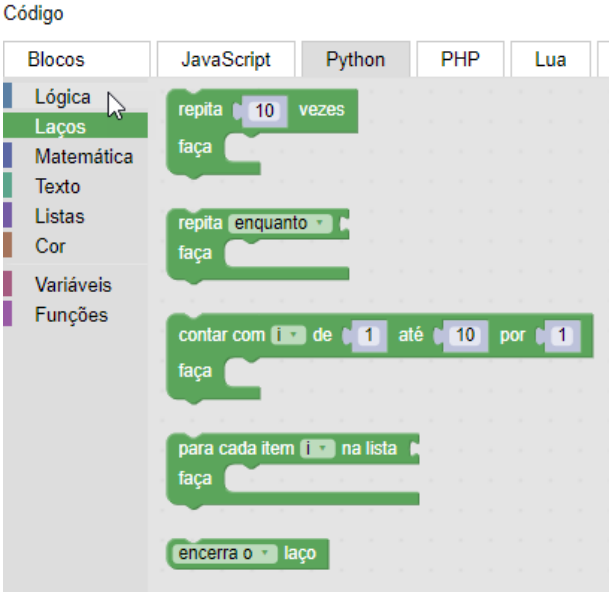
Os blocos da aba lógica são apresentados na Figura 9, responsáveis por fazer comparação e condições exemplo: se (condição) então faça se não faça.

Como apresentado na Figura 10, os blocos da aba Laços, responsáveis por produzir os códigos de laços de repetição exemplo: repita enquanto (condição).

Figura 9 – Tela Principal do Software Blocos de Lógica


Fonte: O Autor.

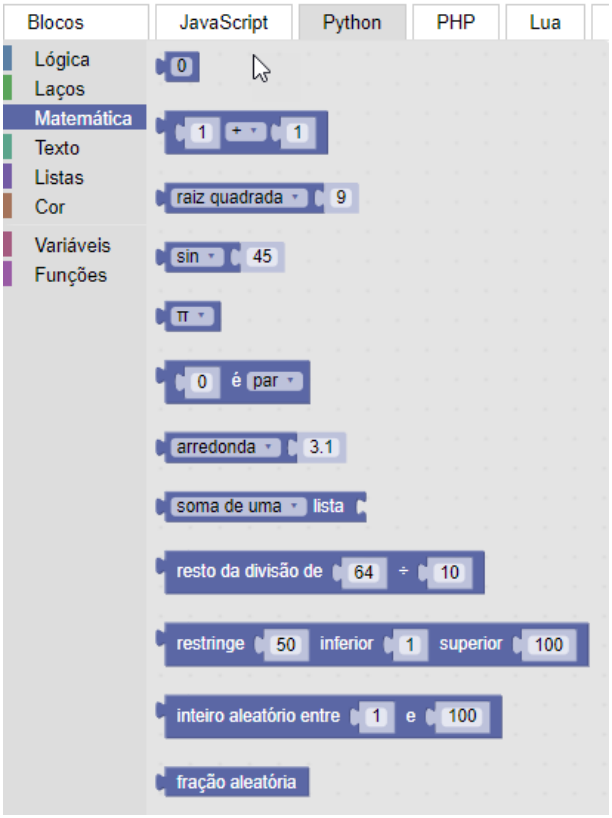
Figura 10 – Tela Principal do Software Blocos de Laços de repetição




Fonte: O Autor.

Blocos da aba Matemática (Figura 11), responsáveis por fazer operações de soma, subtração, raiz quadrada. Exemplo: variavelSoma recebe Variavel1 + Varialvel2.

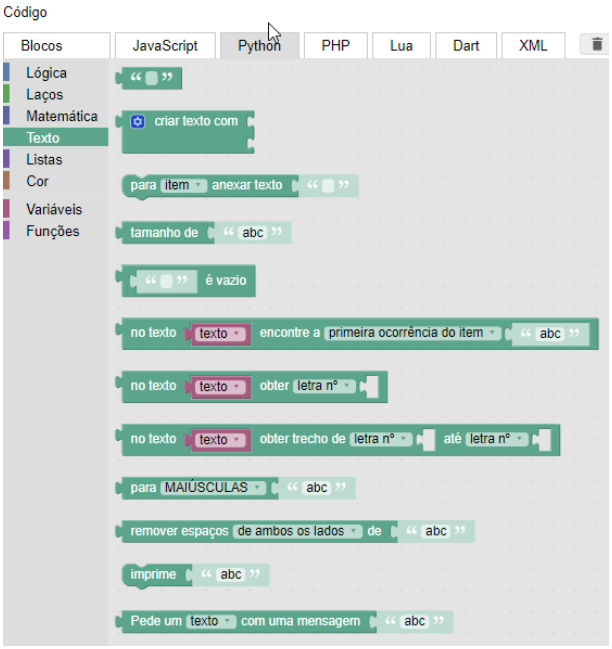
Figura 11 – Tela Principal do Software Blocos de Matemática




Fonte: O Autor.

Na Figura 12 está apresentado os blocos da aba Texto responsáveis por criar, concatenar e apresentar strings.

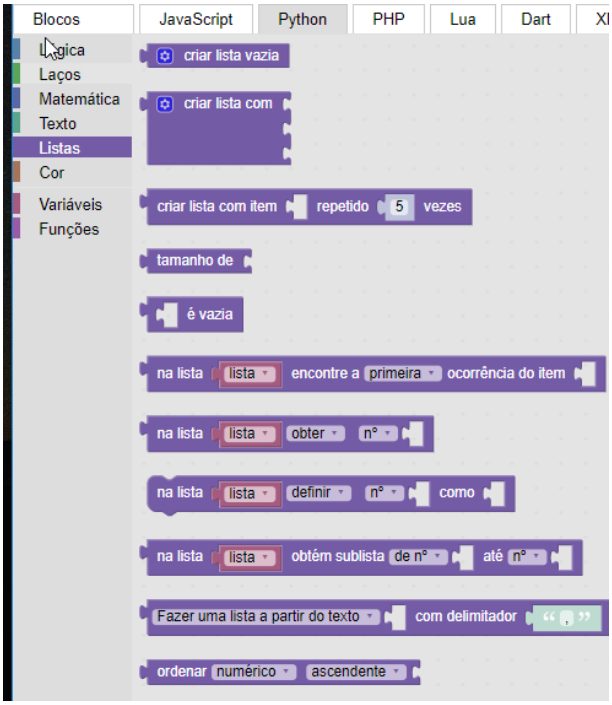
Figura 12 – Tela Principal do Software Blocos de Texto




Fonte: O Autor.

A Figura 13 apresenta os blocos da aba Lista, responsáveis por manipulação de arrays, com funcionalidades como criar array, verificar tamanho da array, buscar na array e ordenar array.

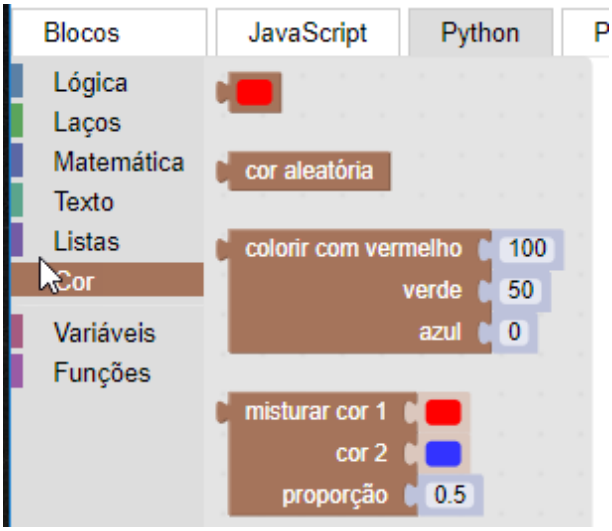
Figura 13 – Tela Principal do Software Blocos de Listas




Fonte: O Autor.

Os blocos da aba Cor estão apresentados na Figura 14 , responsáveis por definir ou randomizar (distribuir aleatoriamente) cores.

Figura 14 – Tela Principal do Software Blocos de Cor

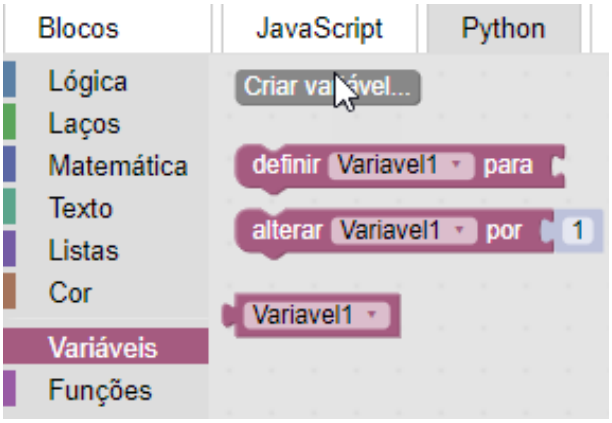



Fonte: O Autor.

Na aba variáveis Figura 15, encontra-se a funcionalidade responsável por criar

variáveis e os blocos encarregados de atribuir e editar valores para as mesmas. Sejam elas globais ou locais.

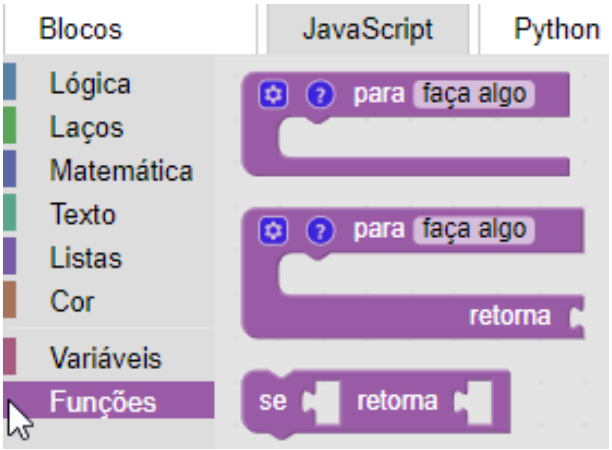
Figura 15 – Tela Principal do Software Blocos de Declaração de Variáveis




Fonte: O Autor.

Por fim na Figura 16 estão apresentados os blocos da aba Funções. Nessa aba é possível declarar métodos e funções bem como fazer suas chamadas e passar os parâmetros definidos.

Figura 16 – Tela Principal do Software Blocos para declaração de funções



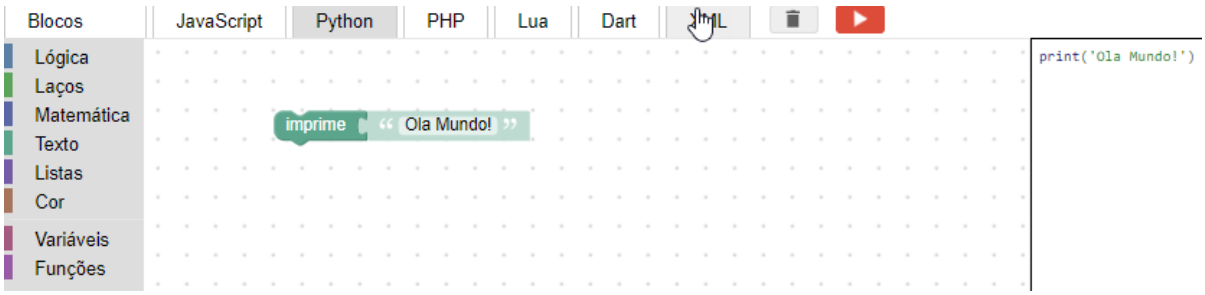

Fonte: O Autor.

## 4.1 Exercícios

Nesta seção será apresentado o escopo de alguns exercícios pertinentes da disciplina de algoritmos e suas resoluções.

a) Desenvolva um programa que ao ser executado apresente a mensagem Olá Mundo em uma mensagem.

Figura 17 – Resolução do exercício a)



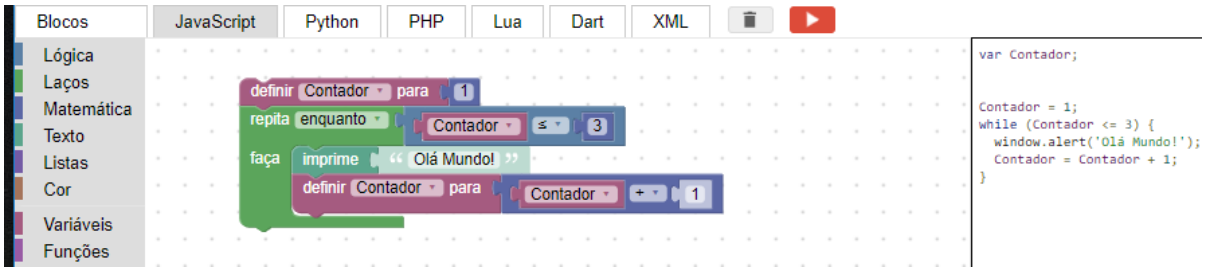

Fonte: O Autor.

Figura 18 – Execução do exercício a)


Fonte: O Autor.

b) Utilizando laço de repetição desenvolva um programa que apresente 3 vezes a mensagem Olá mundo.

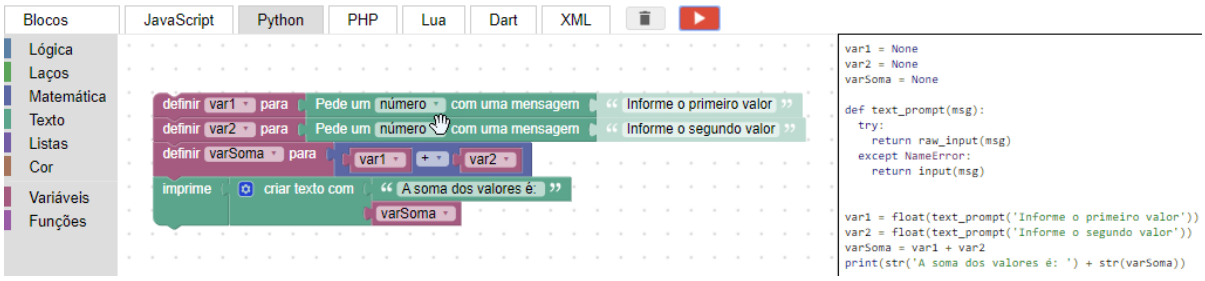
Figura 19 – Resolução do exercício b)




Fonte: O Autor.

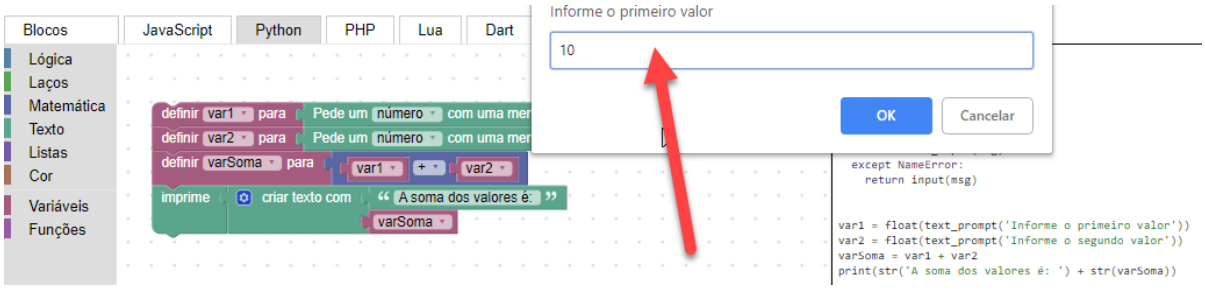
c) Desenvolva um programa que receba dois valores do teclado e apresente a soma dos dois.

Figura 20 – Resolução do exercício c)



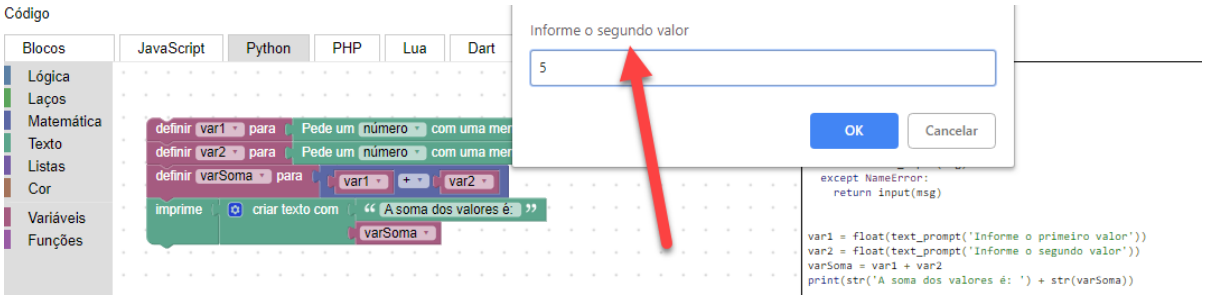

Fonte: O Autor.

Figura 21 – Resolução do exercício c) informando primeiro valor



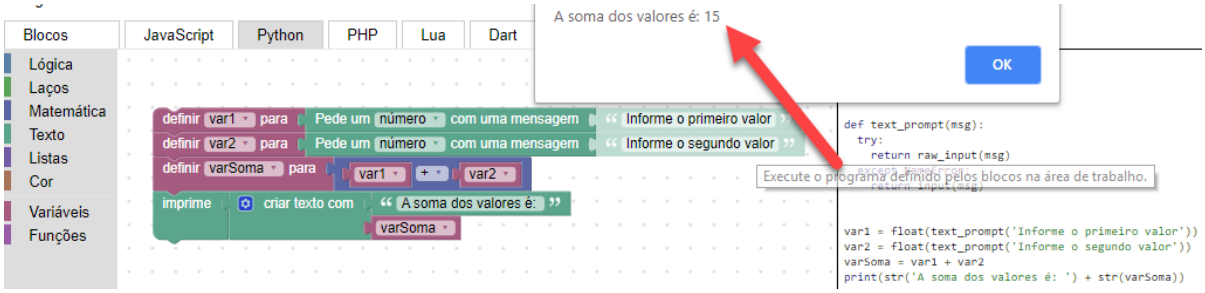

Fonte: O Autor.

Figura 22 – Resolução do exercício c) informando segundo valor




Fonte: O Autor.

Figura 23 – Resolução do exercício c) apresentando a soma dos valoes



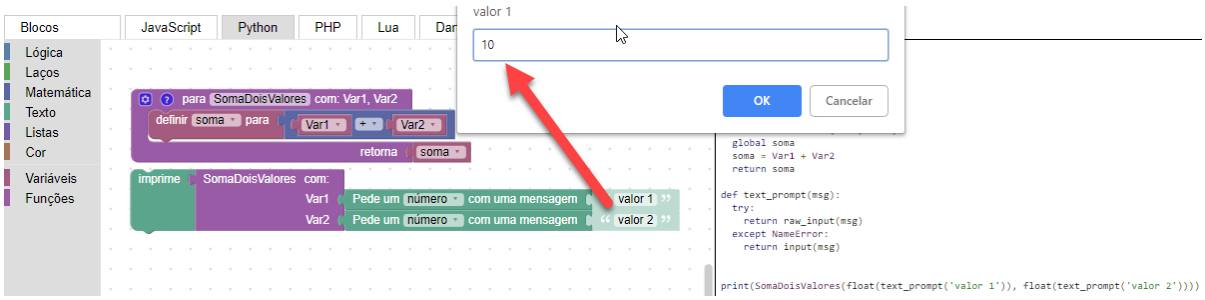

Fonte: O Autor.

d) Utilizando função desenvolva um programa que receba 2 valores e apresente a soma dos dois.

Figura 24 – Resolução do exercício D) montagem dos blocos

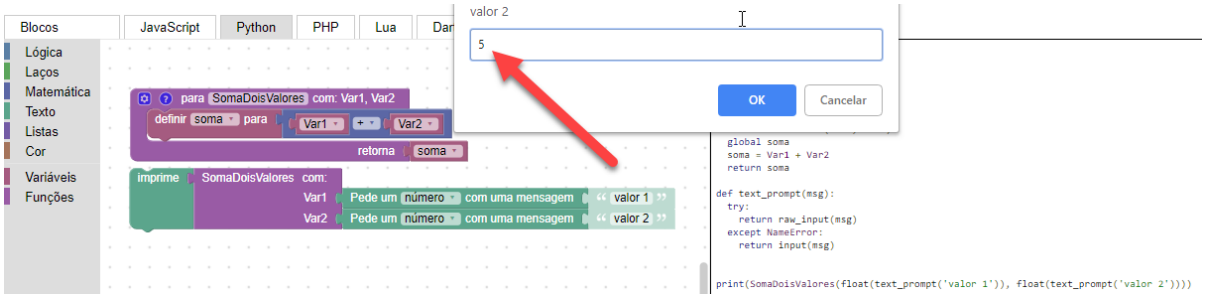

Fonte: O Autor.

Figura 25 – Resolução do exercício D) alimentando primeiro parâmetro



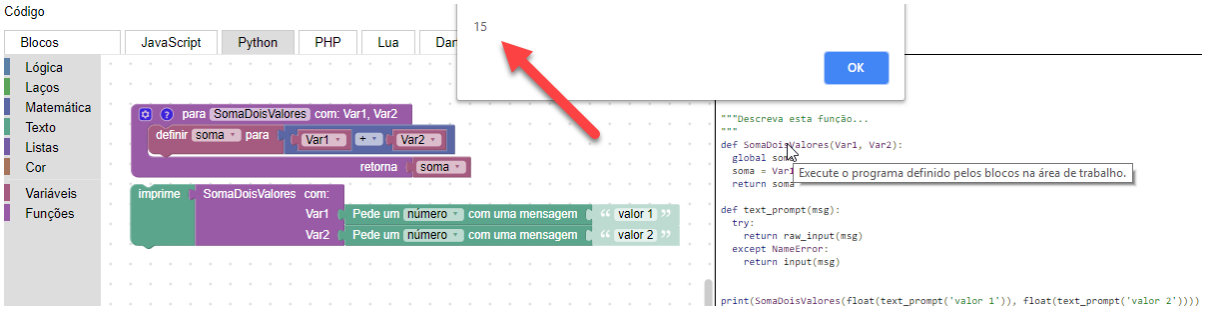

Fonte: O Autor.

Figura 26 – Resolução do exercício D) alimentando segundo parâmetro




Fonte: O Autor.

Figura 27 – Resolução do exercício D) Apresentando o resultado




Fonte: O Autor.

# 5 CONCLUSÃO

Tendo ciência das dificuldades encontradas por alunos e professores, inerentes a pratica de disciplinas envolvendo algoritmos e lógica de programação somados com altos índices de desistência no ensino superior. Surge a necessidade de um estudo aprofundado sobre possibilidades para superar essas problemáticas.

Foi optado por utilizar a linguagem visual para geração de códigos. Com a ideia de proporcionar uma prática em lógica e ao mesmo tempo demonstrar o algoritmo em cinco linguagens de programação distintas, com isso pretende-se ir aproximando o aluno de uma sintaxe textual, para que em um futuro não tão distante, já tenha uma noção sobre programação envolvida com linguagens textuais.

A ferramenta possibilita a criação de códigos independentes de uma linguagem de programação textual, deixando os alunos focados em aprender exclusivamente a lógica, e não mais a sintaxe.

Se aplicada no inicio do processo de aprendizagem, pode contextualizar conceitos abstratos e despertar o interesse pela programação e desenvolvimento de *softwares*, com isso pode contribuir significativamente para todo processo de formação acadêmica, onde as dificuldades são amenizadas durante o decorrer do processo.

Desta maneira procurou-se contribuir com o desempenho do sistema educacional nacional, reduzir os índices de desistência e as reprovações que acabam aumentando o tempo de formação. Suprir a demanda por profissionais formados com uma visão critica sobre desenvolvimento

# REFERÊNCIAS

BORGES, M. Avaliação de uma metodologia alternativa para a aprendizagem de programação. *VIII Workshop de Educação em Computação–WEI*, n. November, 2000. Disponível em: <http://www.niee.ufrgs.br/eventos/SBC/2000/pdf/wei/relatos/selecionados/wei006.pdf>. Citado na página 17.

DEED, D. d. E. E. *Censo da Educação Superior: Notas Estatísticas 2017*. Brasília, DF.: INEP. MEC., 2017. 28 p. Citado 2 vezes nas páginas 14 e 15.

FERREIRA, C.; GONZAGA, F.; SANTOS, R. Um Estudo sobre a Aprendizagem de Lógica de Programação Utilizando Programação por Demonstração. *Computer*, p. 981–990, 2010. Disponível em: <http://www.inf.pucminas.br/sbc2010/anais/pdf/wei/st06{_}03.> Citado na página 11.

GERALDES, W. Programar É Bom Para As Crianças? Uma Visão Crítica Sobre O Ensino De Programação Nas Escolas. *Texto Livre: Linguagem e Tecnologia*, p. 105–117, 2014. Disponível em: <http://periodicos.letras.ufmg.br/index.php/textolivre/article/view/6143>. Citado na página 18.

GIRAFFA, L. M. M.; MÓRA, M. C. Evasão e Disciplina de Algoritmo e Programação: Um Estudo a partir dos Fatores Intervenientes na Perspectiva do Aluno. *III Conferencia Latinoamericana sobre el abandono en la educación superior*, p. 1–10, 2013. ISSN 1098-6596. Citado na página 16.

GOMES, A.; HENRIQUES, J.; MENDES, A. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educação, Formação & Tecnologias*, v. 1, p. [93–103], 2008. ISSN 1646-933X. Disponível em: <http://www.eft.educom.pt/index.php/eft/article/view/23>. Citado 2 vezes nas páginas 16 e 17.

GOMES, G. et al. Dificuldades na aprendizagem da programação no ensino profissional – Perspetiva dos alunos. *Actas do II Congresso Internacional TIC e Educação*, n. January, p. 438–448, 2012. Disponível em: <http://www.researchgate.net/profile/Joana{_}Costa33/publication/267269351{_}DIFICULDADES{_}NA{_}APRENDIZAGEM{_}DA{_}PROGRAMAO{_}NO{_}ENSINO{_}PROFISSIONAL{_}-PERSPETIVA{_}DOS{_}ALUNO>. Citado na página 16.

MORAN, J. Mudando a educação com metodologias ativas. *Convergências Midiáticas, Educação e Cidadania: aproximações jovens*, II, p. 15–33, 2015. Disponível em: <http://www2.eca.usp.br/moran/wp-content/uploads/2013/12/mudando{_}moran.> Citado na página 18.

NETO, F. A. D. A.; CASTRO, T. H. C. D.; JÚNIOR, A. N. D. C. Utilizando o Método Clínico Piagetiano para Acompanhar a Aprendizagem de Programação. *SBIE - Simpósio Brasileiro de Informática na Educação*, p. 527–536, 2006. Citado na página 16.

PEREIRA, P. D. S.; MEDEIROS, M.; MENEZES, J. Análise Do Scratch Como Ferramenta De Auxílio Ao Ensino De Programação De Computadores. *Cobenge*, p. 9, 2012. Disponível em: <http://www.abenge.org.br/CobengeAnteriores/2012/artigos/104281.pdf>. Citado 2 vezes nas páginas 11 e 18.

PRINCIPAL, T. Motivos da Evasão Escolar Texto Principal. p. 1988, 1988. Citado na página 15.

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. *Lbd.Dcc.Ufmg.Br*, 2012. ISSN 2316-6533. Citado 2 vezes nas páginas 15 e 17.

RODRIGUES, F.; BRACKMANN, C. P.; BARONE, D. A. C. Estudo Da Evasão No Curso De Ciência Da Computação Da Ufrgs. *Revista Brasileira de Informática na Educação*, v. 23, n. 01, p. 97, 2015. ISSN 1414-5685. Disponível em: <http://www.br-ie.org/pub/index.php/rbie/article/view/2463>. Citado 2 vezes nas páginas 15 e 16.

SANTOS, R. P. dos; COSTA, H. A. X. Análise de Metodologias e Ambientes de Ensino para Algoritmos , Estruturas de Dados e Programação aos iniciantes em Computação e Informática. *Infocomp Journal of Computer Science*, v. 5, p. 41–50, 2006. ISSN 1982-3363. Citado 2 vezes nas páginas 11 e 12.

TRINDADE, A. G. LINGUAGEM DE PROGRAMAÇÃO VISUAL: UMA NOVA FORMA DE APRESENTAR A PROGRAMAÇÃO DE COMPUTADORES. *Processando o Saber*, p. 65–79, 2015. Citado na página 18.

ZUCULE, M. J. ENGENHARIA DE SOFTWARE. 2017. Citado na página 22.