UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

EMANUEL OLIVEIRA ANDRADE

DESENVOLVIMENTO DE UMA FERRAMENTA PARA O ENSINO DE LÓGICA DE PROGRAMAÇÃO POR MEIO DE DIAGRAMAS DE BLOCOS

GUARAPUAVA

2026

EMANUEL OLIVEIRA ANDRADE

DESENVOLVIMENTO DE UMA FERRAMENTA PARA O ENSINO DE LÓGICA DE PROGRAMAÇÃO POR MEIO DE DIAGRAMAS DE BLOCOS

Development of a Tool for Teaching Programming Logic Through Block Diagrams

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado como requisito parcial para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Profª. Drª Renata Luiza Stange

Coorientador: Prof. Dr. Diego Marczal

GUARAPUAVA

2026

RESUMO

O ensino de lógica de programação apresenta desafios significativos em disciplinas introdutórias, contribuindo para elevados índices de reprovação. Dados históricos do curso de Sistemas para Internet da UTFPR-Guarapuava indicam que a utilização de abordagens visuais baseadas em diagramas de blocos esteve associada ao aumento da mediana de aprovação de 22,2% (2016–2022) para 47,7% (2023–2025). Entretanto, ferramentas educacionais existentes apresentam limitações relacionadas à representação formal de algoritmos e à validação estrutural durante sua construção. Neste contexto, este trabalho propõe a modelagem e especificação de uma ferramenta web interativa para construção e validação estrutural de algoritmos utilizando diagramas de blocos, modelados internamente como grafos direcionados. O sistema permitirá criar algoritmos por meio de blocos visuais conectados, incorporando mecanismos capazes de identificar inconsistências estruturais durante a construção, como ausência de blocos obrigatórios, conexões inválidas e fluxos incompletos. A modelagem proposta será analisada por meio de algoritmos de referência representando estruturas sequenciais, condicionais e de repetição. Espera-se que a solução proposta contribua como ferramenta de apoio ao ensino introdutório de lógica de programação, auxiliando a compreensão da estrutura dos algoritmos e da organização do fluxo lógico.

Palavras-chave: ensino de programação; diagramas de blocos; validação estrutural; algoritmos; pensamento computacional.

ABSTRACT

Teaching programming logic presents significant challenges in introductory courses, contributing to high failure rates. Historical data from the Systems for Internet Technology program at UTFPR-Guarapuava indicate that the use of visual approaches based on block diagrams was associated with an increase in the median approval rate from 22.2% (2016–2022) to 47.7% (2023–2025). However, existing educational tools present limitations regarding the formal representation of algorithms and structural validation during their construction. In this context, this work proposes the modeling and specification of an interactive web-based tool for algorithm construction and structural validation using block diagrams internally modeled as directed graphs. The system will allow users to create algorithms through connected visual blocks, incorporating mechanisms capable of identifying structural inconsistencies during construction, such as missing required blocks, invalid connections, and incomplete flows. The proposed model will be analyzed through reference algorithms representing sequential, conditional, and repetition structures. It is expected that the proposed solution will contribute as a supporting tool for introductory programming logic education, assisting in the understanding of algorithm structure and logical flow organization.

Keywords: programming education; block diagrams; structural validation; algorithms; computational thinking.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

AL	Algoritmos
FP	Fundamentos de Programação
IOO	Introdução à Orientação a Objetos
LP1	Linguagem de Programação 1
PCLPV	Pensamento Computacional e Linguagem de Programação Visual
PCPF	Pensamento Computacional e Fundamentos de Programação
SI	Sistemas para Internet
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

A lógica de programação constitui um dos pilares fundamentais na formação de profissionais da área da computação, estando relacionada ao desenvolvimento do raciocínio lógico e da capacidade de resolução de problemas. Essas competências estão diretamente associadas ao pensamento computacional, que envolve habilidades como decomposição, abstração, reconhecimento de padrões e construção de algoritmos para representação estruturada de soluções (MARCZAL, 2007).

No contexto do ensino superior, disciplinas introdutórias de programação têm como objetivo apresentar esses conceitos a estudantes iniciantes. Entretanto, estudos apontam que esse processo de aprendizagem é frequentemente acompanhado por dificuldades relacionadas à compreensão de problemas, construção de soluções algorítmicas e assimilação da sintaxe de linguagens de programação (MEDEIROS; RAMALHO; FALCÃO, 2019). Essas dificuldades podem impactar diretamente o desempenho acadêmico dos alunos, contribuindo para índices elevados de reprovação e evasão em cursos da área de computação (SOUZA; FRANÇA, 2013).

Um dos fatores associados a esse cenário é a introdução precoce de linguagens de programação textuais, como C e Java. Nesse contexto, o estudante precisa lidar simultaneamente com dois desafios: desenvolver a lógica da solução e compreender a sintaxe da linguagem utilizada. Erros simples de escrita podem impedir a execução do programa, gerando frustração e dificultando o processo de aprendizagem.

Diante disso, abordagens visuais têm sido utilizadas como alternativa para auxiliar o ensino introdutório de programação, permitindo que estudantes concentrem sua atenção inicialmente na organização lógica do algoritmo antes de lidar diretamente com aspectos sintáticos de linguagens textuais.

1.1 Motivação e Contextualização

No curso Superior de Tecnologia em Sistemas para Internet (SI) da Universidade Tecnológica Federal do Paraná (UTFPR) – Campus Guarapuava, esse cenário também pode ser observado nas disciplinas introdutórias relacionadas à programação e algoritmos. A análise histórica das matrizes curriculares do curso evidencia diferentes abordagens pedagógicas adotadas desde 2011.

Entre 2011 e 2015, a matriz curricular era composta pelas disciplinas de Linguagem de Programação 1 (LP1), voltada à elaboração de programas em linguagem de programação, e Algoritmos (AL), direcionada à construção de algoritmos para resolução de problemas computacionais. Nesse período, o percentual agregado de aprovação apresentava mediana de 26,4%, sendo 25,4% em LP1 e 31,4% em AL, conforme ilustrado na Figura 1.

Entre 2016 e 2022, ocorreu uma reestruturação curricular com a introdução das disciplinas Pensamento Computacional e Fundamentos de Programação (PCPF), voltada à resolução

de problemas utilizando linguagem de alto nível, e Introdução à Orientação a Objetos (IOO), focada nos conceitos básicos de orientação a objetos. Apesar da alteração de nomenclatura e abordagem, a utilização predominante de linguagens textuais permaneceu presente, e os índices de aprovação apresentaram leve redução, alcançando mediana agregada de 22,2%, sendo 21,6% em IOO e 25,7% em PCPF.

A partir de 2023, uma mudança metodológica mais significativa foi implementada com a introdução das disciplinas Pensamento Computacional e Linguagem de Programação Visual (PCLPV) e Fundamentos de Programação (FP). Nessa abordagem, a lógica de programação passou a ser ensinada inicialmente por meio de uma linguagem visual baseada em diagramas de blocos, priorizando os fundamentos do pensamento computacional antes da introdução de linguagens textuais.

Os dados mais recentes indicam aumento nos índices de aprovação, apresentando mediana agregada de 47,7%, sendo 76,2% na disciplina FP e 43,1% em PCLPV. Esses resultados podem ser observados nas Figuras 1 e 2, sugerindo que abordagens visuais podem contribuir positivamente para o processo de aprendizagem em disciplinas introdutórias de programação, embora não seja possível estabelecer relação causal direta apenas com base nos dados analisados.

Apesar desses avanços, ainda existem limitações relacionadas às ferramentas utilizadas no ensino de lógica de programação. Em muitos casos, os algoritmos são construídos em papel ou utilizando ferramentas genéricas de desenho, que não foram desenvolvidas especificamente para fins educacionais relacionados à organização estrutural de algoritmos.

Essa limitação pode dificultar a visualização do fluxo de execução, a organização estrutural das conexões e a identificação de inconsistências durante a construção do algoritmo.

Diante desse contexto, este trabalho parte da necessidade de uma ferramenta computacional web voltada especificamente à construção estruturada de algoritmos por meio de diagramas de blocos, incorporando mecanismos de validação estrutural durante o processo de construção.

Assim, este trabalho propõe o desenvolvimento de uma ferramenta web interativa voltada ao apoio do ensino de lógica de programação, permitindo a construção visual de algoritmos por meio de diagramas de blocos conectados entre si. A proposta busca auxiliar estudantes iniciantes na organização do fluxo lógico dos algoritmos, reduzindo dificuldades relacionadas à sintaxe de linguagens textuais e favorecendo a compreensão estrutural do fluxo de execução.

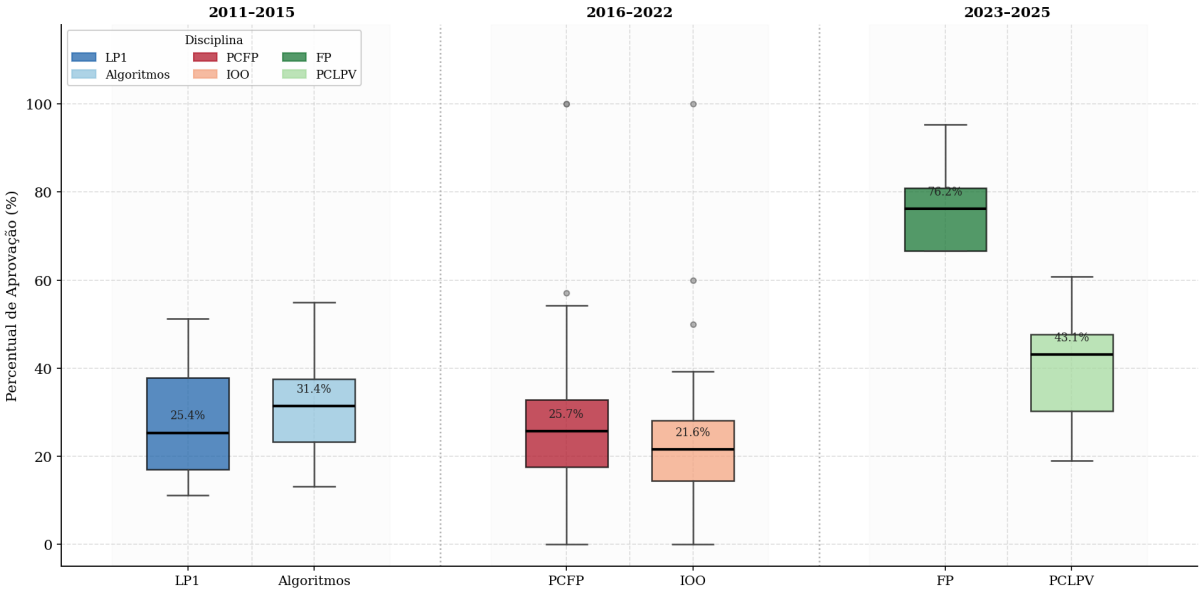


Figura 1 – Percentual de Aprovação por Disciplina e Período Curricular
Fonte: Relatório de Aprovações da Disciplina do Curso de Sistemas para Internet (2026).

[1]

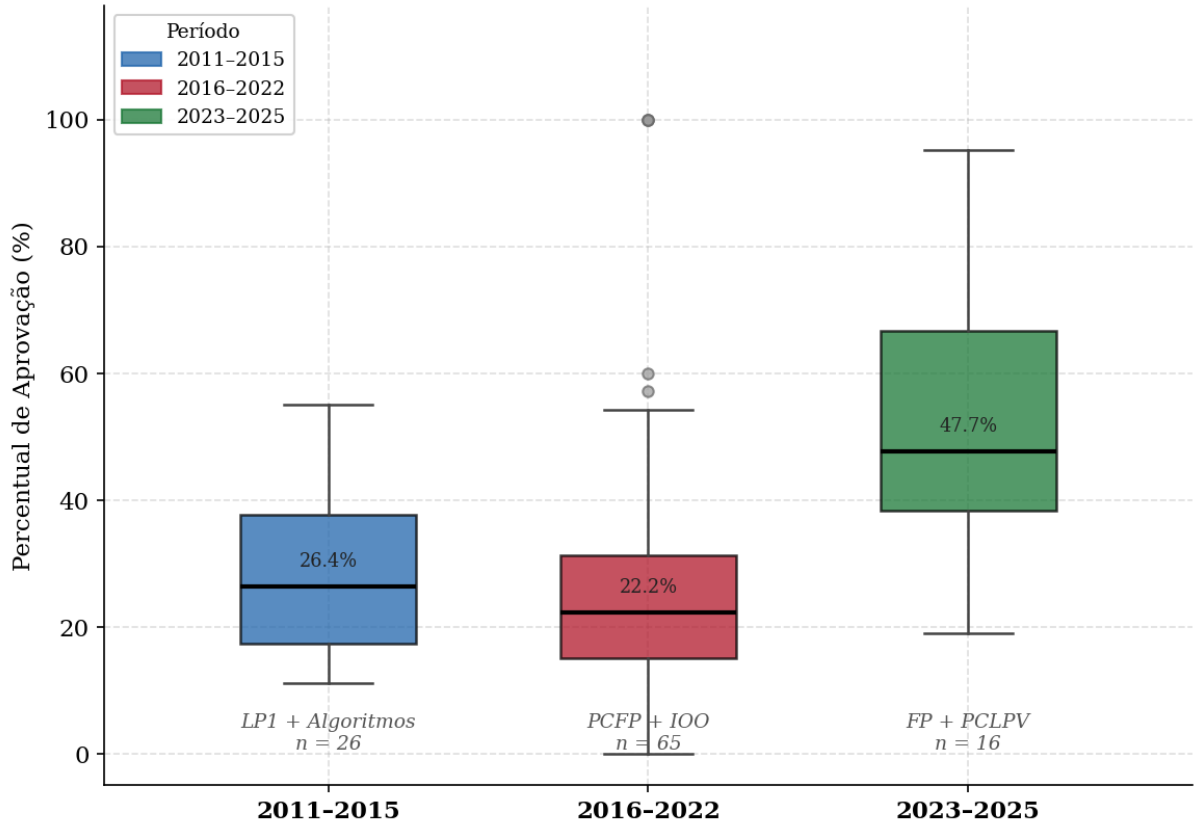


Figura 2 – Percentual de Aprovação Geral por Período Curricular
Fonte: Relatório de Aprovações da Disciplina do Curso de Sistemas para Internet (2026).

[1] A interpretação desses dados estatísticos pode ser realizada por meio de gráficos do tipo boxplot, que permitem analisar distribuição, dispersão e mediana dos valores (PERES, 2022).

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver uma ferramenta web interativa para construção e validação estrutural de algoritmos utilizando diagramas de blocos, com foco no apoio ao ensino de lógica de programação em disciplinas introdutórias.

1.2.2 Objetivos Específicos

- Definir um modelo estrutural para representação de algoritmos baseado em diagramas de blocos e modelagem conceitual por grafos direcionados;
- Especificar regras estruturais para organização e validação dos diagramas de blocos, incluindo restrições de conexão e definição de estruturas obrigatórias;
- Desenvolver uma interface web interativa que permita criação, edição, organização e conexão visual de blocos;
- Implementar mecanismos de validação estrutural capazes de identificar inconsistências no fluxo do algoritmo, como conexões inválidas, blocos desconectados, fluxos incompletos e ausência de elementos obrigatórios;
- Construir algoritmos de referência contendo estruturas sequenciais, condicionais e de repetição para verificação do funcionamento das regras estruturais implementadas.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Este capítulo apresenta os conceitos teóricos que fundamentam o desenvolvimento deste trabalho, abordando aspectos relacionados ao ensino introdutório de algoritmos, ao uso de diagramas de blocos como ferramenta educacional e à representação estrutural de fluxos por meio da teoria de grafos. Também são discutidas interfaces visuais interativas utilizadas no ensino de programação, bem como ferramentas relacionadas ao contexto deste trabalho, permitindo identificar limitações que motivam a proposta apresentada.

2.1 Ensino de Algoritmos e Pensamento Computacional

O ensino de programação em disciplinas introdutórias envolve o desenvolvimento do raciocínio lógico e da capacidade de resolução de problemas. Nesse contexto, estudantes frequentemente enfrentam dificuldades relacionadas à abstração de problemas e à construção de soluções algorítmicas.

O pensamento computacional é definido como um conjunto de habilidades relacionadas à decomposição de problemas, reconhecimento de padrões, abstração e construção de algoritmos (BOWER *et al.*, 2017). Essas habilidades são fundamentais no processo de aprendizagem de programação. A lógica de programação representa a aplicação prática desses conceitos na construção de soluções estruturadas (XAVIER, 2018). O desenvolvimento dessa competência é independente da linguagem de programação utilizada.

Abordagens construtivistas indicam que a aprendizagem é favorecida quando o estudante participa ativamente da construção do conhecimento (SILVA; SOUZA, 2018). Nesse sentido, representações visuais podem contribuir para reduzir a carga cognitiva inicial do ensino de algoritmos.

2.2 Representações Visuais de Algoritmos

Diagramas de blocos são amplamente utilizados no ensino de algoritmos por permitirem a representação visual do fluxo de execução. Essa abordagem facilita a compreensão de estruturas sequenciais, condicionais e de repetição, permitindo que o estudante visualize a organização lógica das instruções antes da implementação textual (MANZANO, 2019).

Estudos indicam que representações visuais podem reduzir dificuldades iniciais no aprendizado de lógica de programação (CREWS; ZIEGLER, 1998). Essas representações utilizam símbolos padronizados para representar operações como entrada, saída, decisão e processamento, contribuindo para a clareza na interpretação dos algoritmos.

2.3 Estruturas de Grafos na Representação de Fluxos

Um grafo é uma estrutura composta por vértices e arestas que representam relações entre elementos (DIESTEL, 2017). Quando essas relações possuem direção, define-se um grafo direcionado. Essa estrutura é amplamente utilizada para representar fluxos de execução em sistemas computacionais, devido à sua capacidade de modelar relações ordenadas entre etapas de processamento.

No contexto de algoritmos, diagramas de blocos podem ser interpretados como estruturas baseadas em grafos direcionados, onde os blocos representam elementos do fluxo e as conexões representam relações de precedência. Essa representação permite modelar estruturas como decisões, ciclos e sequências de execução de forma formal e estruturada.

2.4 Interfaces Visuais para Ensino de Programação

Ambientes baseados em interfaces visuais têm sido utilizados no ensino de programação como forma de reduzir a complexidade inicial do aprendizado. Essas interfaces geralmente utilizam mecanismos de interação direta, como arrastar e soltar (*drag-and-drop*), permitindo a construção de algoritmos de forma visual (SHNEIDERMAN, 1983).

Segundo Sun, Lin e Wu (2024), abordagens visuais podem reduzir a carga cognitiva associada ao ensino de programação textual, favorecendo a compreensão de conceitos fundamentais. Além disso, esses ambientes permitem representar estruturas de controle de fluxo de forma mais intuitiva, contribuindo para o aprendizado de lógica de programação.

2.5 Ferramentas Relacionadas

Diversas ferramentas têm sido utilizadas no ensino introdutório de programação e algoritmos, adotando diferentes abordagens para representação lógica e construção de soluções computacionais.

Esta seção apresenta uma análise de quatro ferramentas representativas de diferentes categorias: programação visual lúdica (Scratch, Blockly), diagramação genérica (Lucidchart) e execução de fluxogramas (Flowgorithm). A análise permitirá identificar as lacunas que motivam o desenvolvimento da ferramenta proposta neste trabalho.

2.5.1 Scratch

O **Scratch** é uma ferramenta desenvolvida pelo MIT Media Lab (2024) baseada em programação visual por blocos encaixáveis. Sua proposta é facilitar o ensino introdutório de programação por meio de uma abordagem lúdica e orientada a eventos.

A Figura 3 apresenta a interface do Scratch.

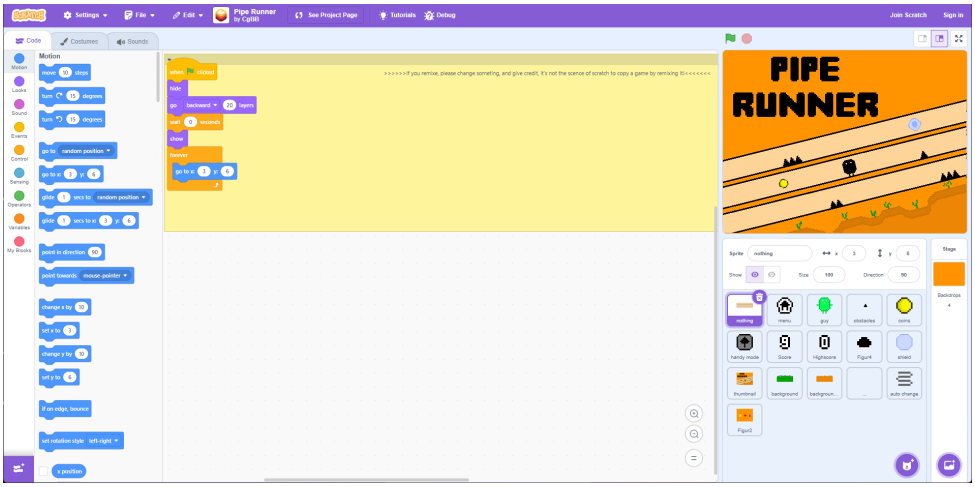


Figura 3 – Interface do Scratch
Fonte: MIT Media Lab (2024).

Apesar de favorecer o ensino introdutório, o Scratch possui foco em animações e programação orientada a eventos, não sendo voltado especificamente para representação formal de algoritmos utilizando diagramas.

2.5.2 Blockly

O **Blockly** é uma biblioteca de programação visual desenvolvida originalmente pelo Google LLC (2024). A ferramenta permite a construção de programas por meio de blocos manipuláveis utilizando interfaces *drag-and-drop*.

A Figura 4 apresenta a interface do Blockly.

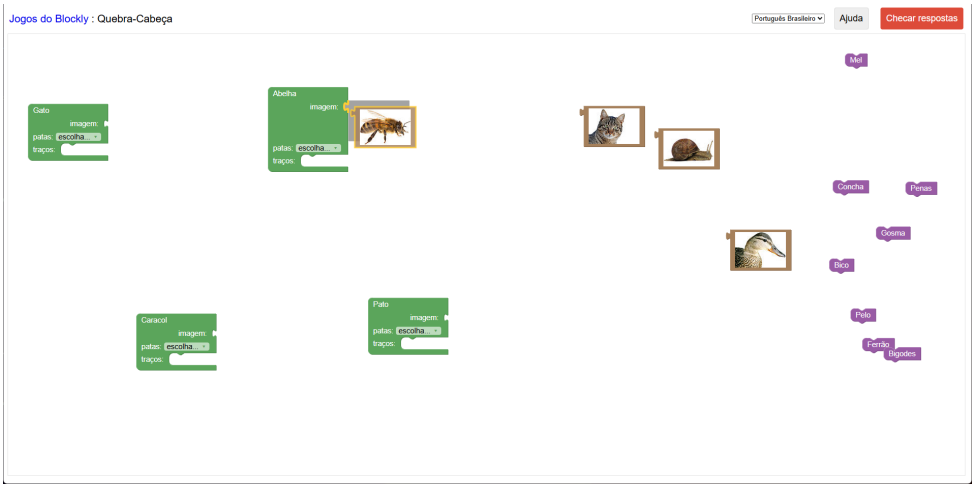


Figura 4 – Interface do Blockly
Fonte: Google LLC (2024).

Embora ofereça elevada flexibilidade para criação de ambientes educacionais, o Blockly não possui foco específico em diagramas de blocos baseados em fluxogramas.

2.5.3 Lucidchart

O **Lucidchart** é uma plataforma online para criação de diagramas e fluxogramas (Lucid Software Inc., 2024). A ferramenta permite representar visualmente estruturas e processos, sendo utilizada em diferentes contextos acadêmicos e profissionais.

A Figura 5 apresenta a interface da plataforma.

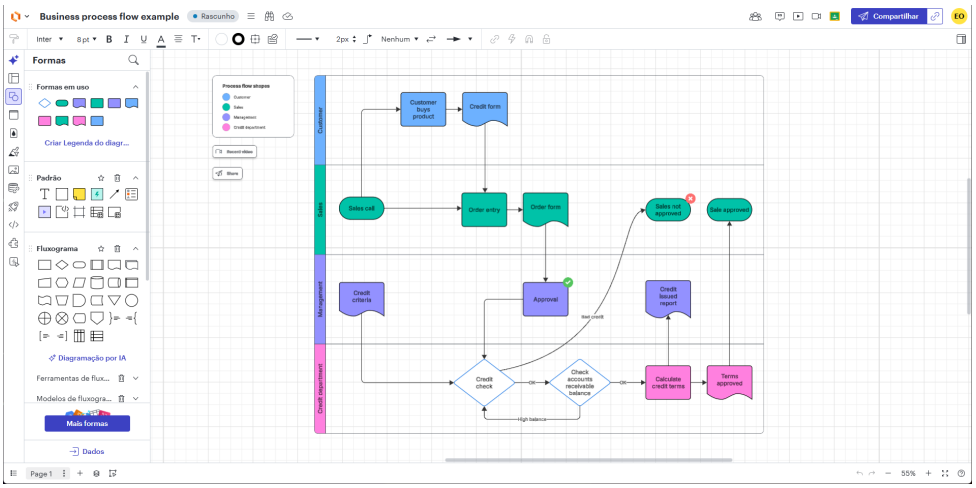


Figura 5 – Interface do Lucidchart
Fonte: Lucid Software Inc. (2024).

Apesar de permitir a criação de fluxogramas, o Lucidchart não possui mecanismos específicos de validação estrutural ou interpretação computacional dos algoritmos desenhados.

2.5.4 Flowgorithm

O **Flowgorithm** é uma ferramenta voltada ao ensino de lógica de programação utilizando fluxogramas executáveis (COOK, 2023). A ferramenta permite criar algoritmos visualmente e acompanhar sua execução.

A Figura 6 apresenta a interface da ferramenta.

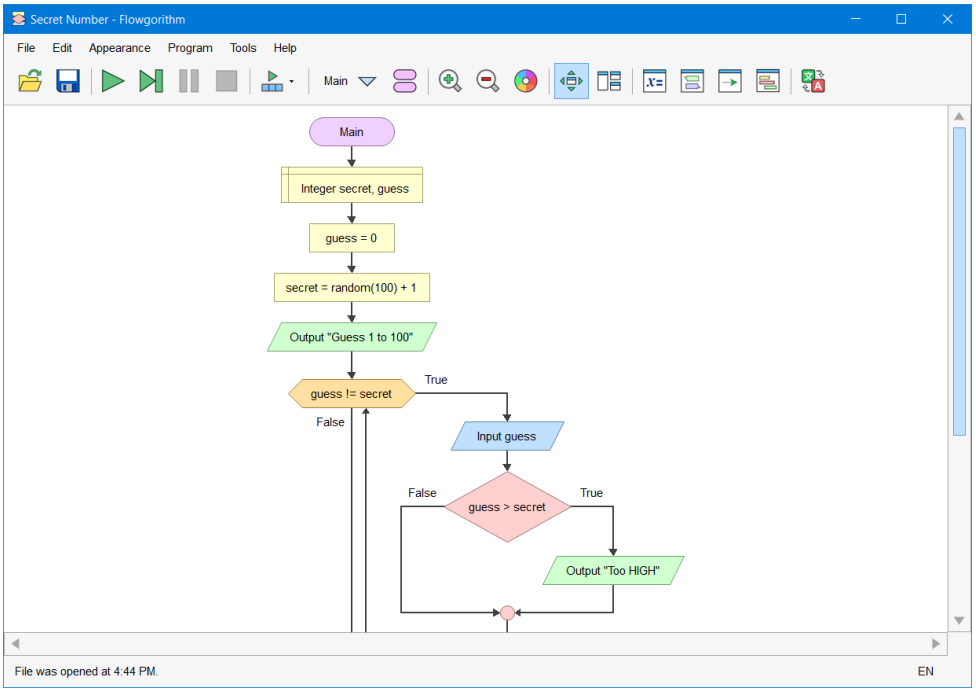


Figura 6 – Interface do Flowgorithm
Fonte: Cook (2023).

Embora o Flowgorithm possua recursos relacionados à execução e interpretação de algoritmos, sua proposta está centrada na simulação do fluxo por meio de uma arquitetura voltada à interpretação de comandos. Diferentemente dessa abordagem, o presente trabalho prioriza a modelagem estrutural e a construção visual do fluxo em ambiente web, com foco na organização das conexões entre blocos.

2.6 Análise Comparativa e Lacuna Identificada

As ferramentas analisadas apresentam abordagens distintas: ferramentas voltadas à programação visual lúdica (MIT Media Lab, 2024), (Google LLC, 2024) priorizam execução baseada em eventos e não seguem a representação formal de fluxogramas; ferramentas de diagramação genérica (Lucid Software Inc., 2024) não possuem validação estrutural ou interpretação computacional; e ferramentas voltadas à execução de fluxogramas (COOK, 2023) realizam validação apenas no momento da execução. Identifica-se, portanto, uma lacuna relacionada a ferramentas web que combinem: (a) representação formal de diagramas de blocos; (b) validação estrutural durante a construção; e (c) interface acessível via navegador, sem necessidade de instalação.

Para fins comparativos, foram definidos os seguintes critérios de análise:

- **Blocos Visuais**: utilização de componentes visuais encaixáveis para construção lógica;
- **Fluxograma**: suporte à representação algorítmica baseada em fluxos e símbolos;

- **Validação Estrutural**: capacidade de impedir ou alertar estruturas inválidas durante a construção;
- **Organização Estrutural**: mecanismos de separação, agrupamento ou hierarquização lógica dos elementos.

A Tabela 1 apresenta uma comparação entre ferramentas utilizadas no ensino de algoritmos e a ferramenta proposta neste trabalho.

Tabela 1 – Comparativo entre ferramentas de ensino de algoritmos

Ferramenta	Blocos Visuais	Fluxograma	Validação Estrutural	Organização Estrutural
Scratch	Sim	Não	Parcial	Parcial
Blockly	Sim	Não	Parcial	Parcial
Lucidchart	Não	Sim	Não	Parcial
Flowgorithm	Não	Sim	Parcial	Parcial
Ferramenta Proposta	Sim	Sim	Sim	Sim

Nota: O termo “Parcial” indica suporte limitado ou indireto ao critério avaliado.
Fonte: Elaborado pelo autor (2026).

Com base na análise realizada, observa-se que as ferramentas existentes normalmente concentram-se em programação visual lúdica, diagramação genérica ou execução de algoritmos. Entretanto, existe uma limitação relacionada à organização estrutural de diagramas de blocos associada à validação do fluxo construído pelo usuário.

Nesse contexto, o presente trabalho propõe o desenvolvimento de uma ferramenta web voltada à construção visual de algoritmos utilizando diagramas de blocos estruturados, permitindo representar o fluxo de execução por meio de conexões visuais entre blocos interativos.

3 METODOLOGIA DE DESENVOLVIMENTO

Este capítulo descreve a abordagem metodológica adotada para desenvolvimento da ferramenta proposta, detalhando as etapas do processo, as técnicas utilizadas para modelagem estrutural dos algoritmos, os recursos empregados na construção da interface visual e os procedimentos definidos para validação do sistema.

Este trabalho caracteriza-se como uma pesquisa aplicada de natureza tecnológica, voltada ao desenvolvimento de uma ferramenta web interativa para construção e validação estrutural de algoritmos utilizando diagramas de blocos.

A proposta busca oferecer um ambiente visual voltado ao apoio do ensino introdutório de lógica de programação, permitindo que estudantes construam algoritmos por meio da manipulação de blocos interconectados responsáveis pela representação das estruturas de execução.

3.1 Etapas de Desenvolvimento

O desenvolvimento do sistema será organizado nas seguintes etapas:

1. definição da modelagem estrutural utilizada para representação dos algoritmos;
2. especificação das regras estruturais associadas aos blocos visuais;
3. desenvolvimento da interface gráfica interativa;
4. implementação dos mecanismos de validação estrutural;
5. realização de testes utilizando algoritmos de referência.

Essa organização permitirá separar os aspectos relacionados à representação visual dos algoritmos das regras responsáveis pela integridade estrutural do fluxo construído pelo usuário.

3.2 Modelagem Estrutural Baseada em Grafos

A representação estrutural dos algoritmos adotada neste trabalho será baseada em diagramas de blocos organizados como grafos direcionados.

Cada bloco visual do algoritmo representará um vértice do grafo, enquanto as conexões estabelecidas entre os blocos representarão arestas responsáveis pela continuidade do fluxo de execução.

Essa abordagem permitirá interpretar estruturalmente os algoritmos construídos pelo usuário, possibilitando aplicar mecanismos de validação relacionados à conectividade do fluxo, existência de caminhos válidos, controle de bifurcações condicionais e formação de ciclos de repetição.

Os diferentes tipos de blocos utilizados no ambiente possuirão comportamento estrutural específico dentro do grafo, permitindo representar estruturas sequenciais, condicionais, iterativas e modularizadas.

A utilização de grafos direcionados também permitirá separar a representação visual da lógica estrutural do algoritmo, favorecendo futuras integrações com mecanismos de interpretação e execução passo a passo.

3.3 Interface Visual Interativa

A interface do sistema será desenvolvida com foco na manipulação visual dos elementos algorítmicos. O ambiente permitirá criar, mover, conectar e organizar blocos por meio de interações do tipo arrastar e soltar (*drag-and-drop*).

Além da construção visual dos algoritmos, a interface buscará facilitar a compreensão do fluxo de execução, permitindo ao estudante acompanhar visualmente a organização estrutural do algoritmo desenvolvido.

Também serão disponibilizados mecanismos visuais para auxiliar a organização do fluxo, incluindo conexões entre blocos, representação de bifurcações condicionais e visualização de estruturas de repetição.

3.4 Validação Estrutural

O sistema incorporará mecanismos de validação estrutural responsáveis por analisar a integridade das conexões estabelecidas entre os blocos do algoritmo.

A validação estrutural terá como objetivo identificar inconsistências relacionadas à organização formal do fluxo, incluindo conexões inválidas, ausência de elementos obrigatórios, interrupções no fluxo de execução e estruturas incompatíveis com as regras definidas pelo ambiente.

A proposta não possuirá como objetivo avaliar a correção lógica da solução implementada pelo estudante, concentrando-se exclusivamente na verificação estrutural do algoritmo representado visualmente.

3.5 Validação do Sistema

A validação do sistema buscará verificar tanto o funcionamento das funcionalidades implementadas quanto a capacidade da ferramenta em representar e analisar estruturalmente algoritmos construídos por meio de diagramas de blocos.

Para realização da validação do sistema, serão consideradas duas etapas complementares:

1. **Validação Funcional**

   Esta etapa consistirá na verificação das principais funcionalidades disponibilizadas pelo ambiente, buscando garantir o funcionamento adequado da interface e dos mecanismos estruturais implementados.

   Entre os aspectos avaliados, destacam-se:

   - criação e manipulação de blocos;
   - estabelecimento de conexões entre elementos;
   - organização visual do fluxo algorítmico;
   - funcionamento das regras de validação estrutural;
   - integração entre interface gráfica e mecanismos de validação.

   A validação funcional buscará verificar se os recursos implementados permitem construir algoritmos visualmente de maneira consistente e compatível com as regras estruturais definidas pelo sistema.

2. **Validação Estrutural com Algoritmos de Referência**

   Esta etapa será realizada por meio da construção de algoritmos representativos das principais estruturas utilizadas no ensino introdutório de lógica de programação.

   O objetivo será verificar a capacidade do sistema em representar diferentes organizações de fluxo e identificar inconsistências estruturais durante a construção dos algoritmos.

   Entre os algoritmos utilizados para validação estrutural, destacam-se:

   a) cálculo de média aritmética;
   b) verificação de número par ou ímpar;
   c) cálculo de fatorial;
   d) soma acumulada até condição de parada;
   e) verificação de aprovação por média e frequência.

   A construção desses algoritmos permitirá avaliar o comportamento das regras estruturais implementadas, verificando aspectos como continuidade do fluxo, organização das conexões, formação de estruturas condicionais e representação adequada de ciclos de repetição.

4 MODELAGEM E ESPECIFICAÇÃO DO SISTEMA

Este capítulo apresenta a modelagem e a especificação do sistema proposto, descrevendo os principais componentes da solução e as decisões adotadas durante o desenvolvimento da interface. Também são apresentados os elementos relacionados à organização visual do ambiente, à modelagem dos diagramas de blocos e aos mecanismos de validação estrutural utilizados pelo sistema.

4.1 Infraestrutura do Sistema

A solução será desenvolvida utilizando arquitetura cliente-servidor, separando os componentes responsáveis pela interface visual e pelo processamento das regras estruturais do sistema.

As principais tecnologias utilizadas serão:

- **React**[1]: biblioteca JavaScript voltada à construção de interfaces web interativas baseadas em componentes reutilizáveis. Sua utilização busca facilitar a criação de elementos dinâmicos da interface e o gerenciamento do estado visual da aplicação;
- **TypeScript**[2]: linguagem baseada em JavaScript com suporte à tipagem estática. Sua adoção busca aumentar a organização do código-fonte, reduzir erros durante o desenvolvimento e facilitar a manutenção da aplicação;
- **React Flow**[3]: biblioteca voltada à construção de interfaces baseadas em nós e conexões. Será utilizada para implementação da manipulação visual dos diagramas de blocos, conexões de fluxo e movimentação dos elementos do algoritmo;
- **Backend da aplicação**: componente responsável pelo processamento das regras de validação estrutural, gerenciamento das conexões entre blocos e controle das informações do sistema;
- **Git e GitHub**[4,5]: ferramentas utilizadas para controle de versão e gerenciamento do código-fonte, permitindo registro das modificações realizadas durante o desenvolvimento, manutenção do histórico de alterações e sincronização do projeto em repositórios remotos;

[1] https://react.dev/
[2] https://www.typescriptlang.org/
[3] https://reactflow.dev/
[4] https://git-scm.com/
[5] https://github.com/

- **Docker**[6]: plataforma utilizada para padronização do ambiente de desenvolvimento e execução da aplicação, facilitando a configuração e reprodução do sistema em diferentes máquinas.

4.2 Modelo de Representação dos Algoritmos

Esta seção apresenta o modelo de representação dos algoritmos adotado no sistema proposto, descrevendo tanto a notação visual utilizada para construção dos diagramas quanto a estrutura interna empregada para organização e validação do fluxo.

4.2.1 Notação dos Diagramas de Blocos

Os diagramas de blocos adotados neste trabalho são baseados na representação utilizada na disciplina de Pensamento Computacional e Linguagem de Programação Visual (PCLPV). Essa notação tem como objetivo representar algoritmos de forma visual, facilitando a compreensão do fluxo de execução e reduzindo a complexidade associada à programação textual.

Cada elemento do diagrama representa uma operação específica do algoritmo, enquanto as conexões indicam a ordem de execução das instruções. A Tabela 2 sintetiza os símbolos utilizados na ferramenta proposta.

[6] https://www.docker.com/

Tabela 2 – Notação dos Diagramas de Blocos

Símbolo	Representação	Descrição
Início/Fim	Elipse	Representa o início ou término do algoritmo ou sub-rotina, delimitando o fluxo de execução.
Entrada	Paralelogramo inclinado a direita com seta no canto superior esquerdo	Indica operações de entrada de dados fornecidos pelo usuário.
Saída	Paralelogramo inclinado a direita com seta no canto inferior direito	Representa a exibição de dados produzidos pelo algoritmo.
Memória	Retângulo com marcação interna	Representa armazenamento de valores em variáveis durante a execução.
Processo	Retângulo	Representa operações de processamento, como cálculos, atribuições e chamadas de funções.
Decisão	Losango	Representa estruturas condicionais com ramificação do fluxo de execução.
Conector	Círculo	Utilizado para ligação entre diferentes partes do diagrama.
Setas	Linhas direcionais	Indicam o sentido do fluxo de execução do algoritmo.
Sub-rotina	Retângulo duplo	Representa módulos reutilizáveis e blocos de decomposição do algoritmo.

A Figura 7 apresenta os símbolos utilizados nos diagramas de blocos adotados neste trabalho.

A Figura 8 apresenta um exemplo de algoritmo construído utilizando diagramas de blocos.

Os algoritmos serão representados internamente como estruturas compostas por elementos (blocos) e relações (conexões). Essa estrutura permitirá que o sistema interprete o diagrama como uma entidade computacional manipulável.

4.2.2 Modelagem do Fluxo em Grafo

Internamente, os algoritmos construídos no ambiente visual serão representados por meio de uma estrutura baseada em grafos direcionados.

Pretende-se modelar o fluxo do algoritmo como um grafo direcionado $G = (V, E)$, no qual V representa o conjunto de blocos (vértices) e E representa o conjunto de conexões (arestas) estabelecidas entre os elementos do diagrama. Cada vértice $v \in V$ possui um tipo associado, como início, processo, decisão ou saída, além de propriedades específicas utilizadas

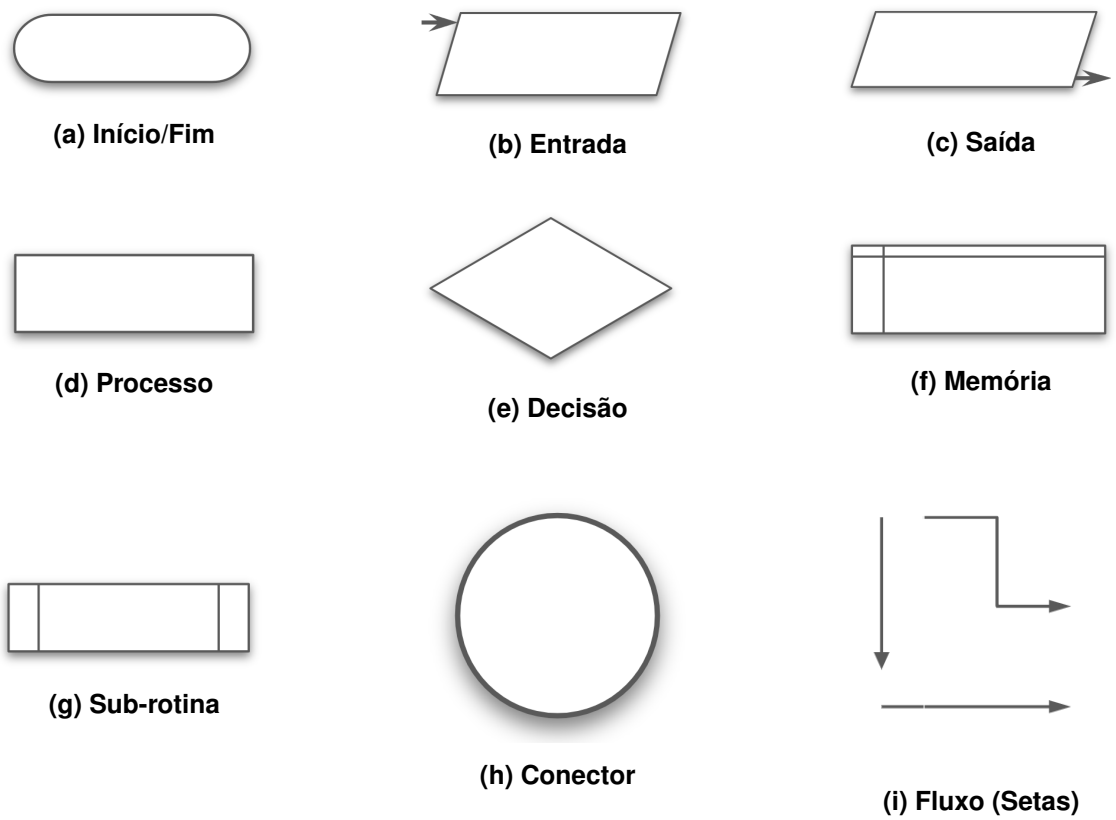


Figura 7 – Símbolos utilizados nos diagramas de blocos
Fonte: Material da disciplina Pensamento Computacional e Linguagem de Programação Visual.

durante a validação estrutural e interpretação do fluxo. As arestas representam relações de precedência entre os blocos, definindo a direção do fluxo de execução do algoritmo.

A modelagem proposta permitirá interpretar estruturalmente os diagramas construídos pelo usuário, possibilitando a realização de validações relacionadas à continuidade do fluxo, conectividade dos blocos e formação de estruturas condicionais e de repetição.

A modelagem proposta permitirá a realização de validações estruturais sobre o diagrama construído, considerando propriedades como:

- existência de um único bloco inicial sem conexões de entrada;
- verificação da conectividade entre os blocos do diagrama;
- verificação da consistência das estruturas condicionais e de repetição.

Além da validação estrutural, a representação em grafo também fornecerá base para futuras integrações com mecanismos de interpretação e execução do algoritmo, permitindo percorrer o fluxo de acordo com as conexões definidas entre os blocos.

A Tabela 3 apresenta a relação entre os elementos visuais utilizados nos diagramas de blocos e sua interpretação estrutural baseada em grafos direcionados.

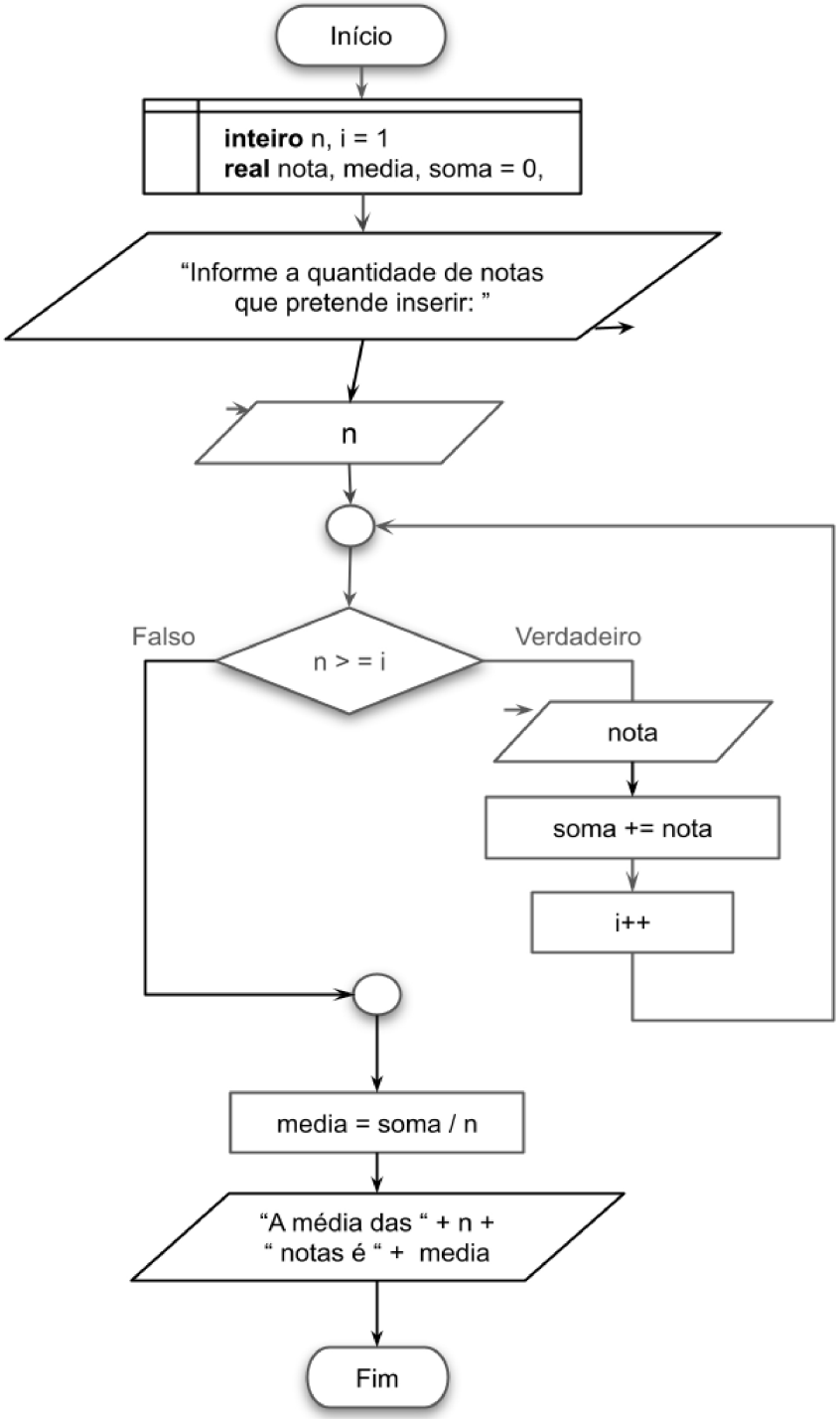


Figura 8 – Exemplo de algoritmo utilizando diagramas de blocos
Fonte: Material da disciplina Pensamento Computacional e Linguagem de Programação Visual.

Tabela 3 – Modelagem Conceitual: Diagrama de Blocos como Estrutura de Dados

Bloco Visual	Papel na Estrutura Lógica	Função Estrutural no Fluxo do Algoritmo
Início	Nó Inicial.	Define o ponto de partida do algoritmo. Na validação estrutural, é o único bloco que não pode receber conexões de entrada.
Processo / Memória / Entrada / Saída	Nós de Execução Linear.	Representam instruções sequenciais. Recebem uma conexão de entrada e emitem uma única conexão de saída, mantendo a linearidade do fluxo.
Subprocesso / Sub-rotina	Nó de Modularização.	Representa a chamada de uma rotina auxiliar reutilizável dentro do algoritmo, permitindo dividir o fluxo em componentes menores e organizados.
Decisão	Nó de Bifurcação (Condicional).	Representa estruturas condicionais do algoritmo, permitindo dividir o fluxo em diferentes caminhos possíveis.
Conector (Junção)	Ponto de Convergência.	Utilizado para unificar caminhos previamente divididos por uma estrutura condicional, restabelecendo o fluxo principal do algoritmo.
Conector (Repetição)	Elo de Retorno (Loop).	Cria um ciclo estrutural ao apontar o fluxo para um bloco anterior. Representa o fechamento de uma estrutura de repetição.
Setas	Conexões de Fluxo.	Representam a ordem de execução do algoritmo, indicando o caminho que o fluxo deve percorrer entre os blocos.

A partir dessa modelagem estrutural, diferentes construções algorítmicas podem ser interpretadas de acordo com o comportamento do fluxo no grafo direcionado.

Estruturas condicionais podem ser representadas como nós de bifurcação responsáveis pela divisão do fluxo em múltiplos caminhos. De forma semelhante, conectores podem atuar como pontos de convergência utilizados para unificar caminhos previamente separados.

Estruturas de repetição podem ser modeladas como ciclos no grafo, caracterizando retornos no fluxo de execução. Já sub-rotinas podem ser compreendidas como estruturas modulares reutilizáveis associadas a partes específicas do algoritmo.

Essa abordagem fornecerá suporte aos mecanismos de validação estrutural do sistema, permitindo identificar inconsistências relacionadas à continuidade do fluxo, conexões inválidas e estruturas incompletas.

4.3 Regras Estruturais dos Blocos

Esta seção apresenta as regras estruturais definidas para organização e validação dos diagramas de blocos utilizados no sistema. Essas regras têm como objetivo garantir a consistência do fluxo algorítmico representado visualmente, restringindo conexões incompatíveis e identificando estruturas inválidas durante a construção do algoritmo.

4.3.1 Regras Gerais de Validação

Pretende-se incorporar verificações de validação estrutural responsáveis por analisar a organização dos blocos e das conexões estabelecidas no fluxo do algoritmo.

Cada tipo de bloco possuirá restrições específicas relacionadas à quantidade de conexões permitidas, direção do fluxo e comportamento esperado dentro da estrutura algorítmica. Essas regras permitirão verificar a consistência estrutural do diagrama construído pelo usuário antes de sua interpretação ou execução.

Entre as principais regras gerais de validação estrutural adotadas no sistema, destacam-se:

1. existência obrigatória de pelo menos um bloco de início e um bloco de fim válidos;
2. continuidade do fluxo entre os blocos por meio de conexões válidas;
3. restrição de conexões incompatíveis entre determinados tipos de blocos;
4. controle de bifurcações em estruturas condicionais;
5. formação adequada de ciclos em estruturas de repetição;
6. identificação de blocos desconectados ou inacessíveis no fluxo principal.

Essas regras serão utilizadas pelo sistema para verificar a integridade estrutural do algoritmo representado visualmente, contribuindo para a organização do fluxo construído e a redução de inconsistências durante a construção dos diagramas.

4.3.2 Regras Específicas dos Blocos

Cada bloco utilizado no sistema possui características estruturais próprias, relacionadas à sua função dentro do fluxo algorítmico e às conexões permitidas durante a construção do diagrama.

A Tabela 4 apresenta as principais regras estruturais associadas aos blocos visuais definidos no sistema.

Tabela 4 – Regras estruturais associadas aos blocos visuais

Bloco	**Função**	**Regras Estruturais**
Início/Fim	Delimitar o início e o encerramento do algoritmo.	O bloco de início não permite conexões de entrada e o bloco de fim não permite conexões de saída. O algoritmo deve possuir pelo menos um bloco inicial e um bloco final válidos.
Memória	Definir variáveis utilizadas durante a execução.	As variáveis devem ser declaradas antes de sua utilização em outros blocos. Quando utilizado, o bloco deve estar associado ao fluxo principal do algoritmo.
Entrada	Receber dados fornecidos pelo usuário.	Os identificadores utilizados devem corresponder a variáveis previamente definidas no bloco de memória.
Saída	Exibir informações produzidas durante a execução.	Permite apresentação de mensagens, valores de variáveis e resultados de operações realizadas no fluxo.
Processo	Executar operações de processamento e manipulação de dados.	Permite atribuições, cálculos e atualização de variáveis durante a execução do algoritmo.
Decisão	Representar estruturas condicionais.	Deve possuir duas conexões de saída válidas, associadas aos resultados verdadeiro e falso da condição lógica avaliada.
Conector	Organizar continuidade e junção do fluxo.	Pode ser utilizado para unificação de caminhos condicionais ou estabelecimento de conexões associadas às estruturas de repetição.
Sub-rotina	Representar módulos reutilizáveis do algoritmo.	Permite decomposição estrutural do fluxo em componentes independentes e reutilizáveis.

As regras estruturais apresentadas serão utilizadas pelos mecanismos de validação do sistema para verificar a consistência do fluxo construído pelo usuário, permitindo identificar conexões inválidas, estruturas incompletas e inconsistências na organização do algoritmo.

4.3.3 Validação Estrutural

A validação estrutural do sistema será realizada por meio da análise do grafo direcionado formado pelos blocos e conexões definidos pelo usuário durante a construção do algoritmo.

Nesse modelo, cada bloco é tratado como um nó do grafo, enquanto as conexões representam as arestas responsáveis por definir o fluxo de execução do algoritmo. Essa representação permitirá que o sistema analise estruturalmente o diagrama construído, verificando sua consistência organizacional antes da etapa de interpretação ou execução.

Durante o processo de validação, o sistema identificará situações que podem comprometer a integridade do fluxo algorítmico, como ausência de elementos obrigatórios, interrupções no fluxo principal, conexões incompatíveis entre blocos e estruturas condicionais incompletas.

Além disso, pretende-se incorporar verificações relacionadas à formação de ciclos em estruturas de repetição, buscando identificar possíveis inconsistências estruturais no fluxo.

O sistema também será capaz de identificar blocos desconectados ou inacessíveis dentro do diagrama, permitindo detectar estruturas que não participam efetivamente do fluxo principal do algoritmo.

Essa abordagem permite que o ambiente realize verificações estruturais antes da execução do algoritmo, contribuindo para organização do fluxo construído e redução de inconsistências durante o processo de modelagem visual.

4.4 Modelagem da Interface

Esta seção apresenta a organização da interface visual do sistema, descrevendo a estrutura adotada para separar os elementos responsáveis pela edição do fluxo, disponibilização dos blocos e exibição das propriedades dos componentes. A interface foi projetada com foco na interação visual entre o usuário e os elementos do algoritmo, permitindo manipulação dinâmica dos blocos e conexões por meio de operações de criação, movimentação e ligação entre componentes gráficos. Também são apresentados os protótipos desenvolvidos para planejamento visual do ambiente.

4.4.1 Organização Visual da Interface

A interface do sistema foi organizada em componentes responsáveis pela edição visual do fluxo, manipulação dos blocos e exibição das propriedades associadas aos elementos do algoritmo.

O ambiente principal é composto por uma área central de edição dos diagramas, um painel lateral contendo os blocos disponíveis e um painel auxiliar destinado à configuração dos elementos selecionados.

Essa organização busca facilitar a interação do usuário com o ambiente visual, separando os recursos de construção, configuração e navegação do fluxo algorítmico.

Com base nessa organização, foram definidos os principais componentes responsáveis pela interação do usuário com o ambiente visual. A Figura 9 apresenta a organização geral da interface proposta.

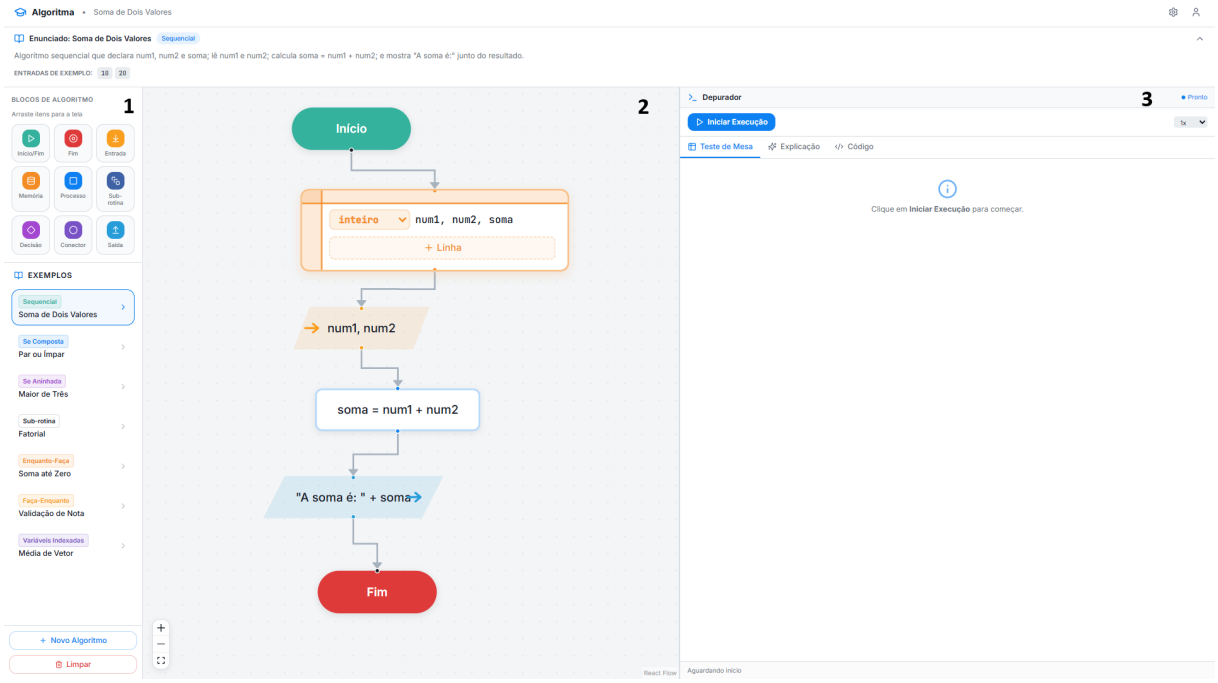


Figura 9 – Organização geral da interface: (1) painel lateral com blocos disponíveis, (2) área central de edição do fluxo, (3) painel de propriedades à direita
Fonte: Elaborado pelo autor (2026).

4.4.2 Área de Edição

A área de edição corresponde ao espaço principal de interação do usuário com o sistema, sendo responsável pela construção visual dos algoritmos. Nesse ambiente, os blocos podem ser inseridos, posicionados e conectados livremente, permitindo organizar o fluxo de execução de acordo com a lógica definida pelo usuário.

Pretende-se permitir a criação de conexões por meio de interações gráficas diretas, utilizando operações do tipo *drag-and-drop* e criação de arestas entre pontos de conexão compatíveis.

Além da representação visual do algoritmo, a área de edição também fornecerá suporte às operações de seleção, movimentação e reorganização dos componentes presentes no fluxo.

4.4.3 Painel de Blocos

O painel de blocos é responsável por disponibilizar os elementos utilizados na construção dos algoritmos.

Nesse componente, os blocos são organizados de acordo com suas funções estruturais, permitindo que o usuário selecione elementos relacionados a entrada, saída, processamento, decisão e controle do fluxo.

A organização visual dos blocos busca facilitar a identificação dos componentes disponíveis e reduzir a complexidade inicial durante a construção do algoritmo.

A Figura 10 apresenta o painel lateral utilizado para disponibilização dos blocos no ambiente visual.

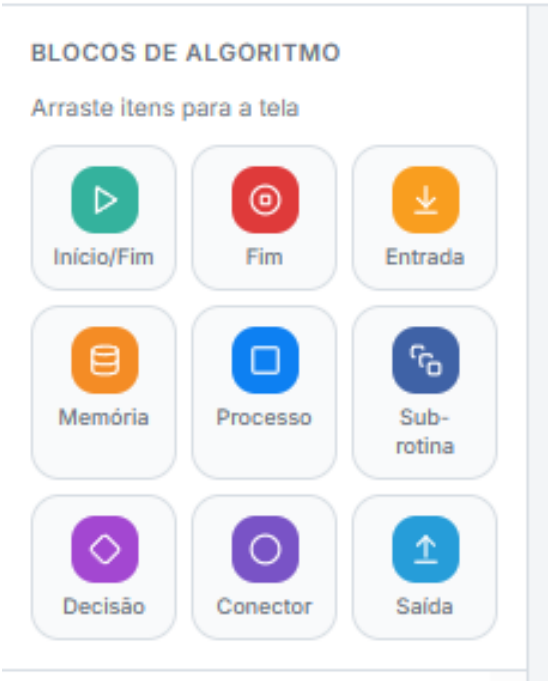
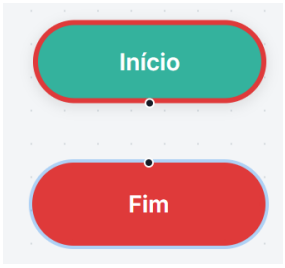
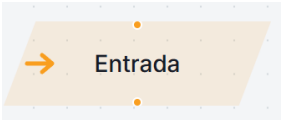


Figura 10 – Painel de blocos do ambiente visual
Fonte: Elaborado pelo autor (2026).

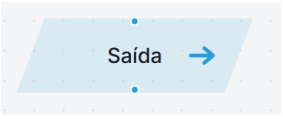
A Figura 11 apresenta os blocos implementados no protótipo do sistema.



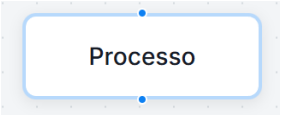
(a) Início/Fim



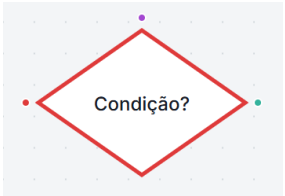
(b) Entrada



(c) Saída

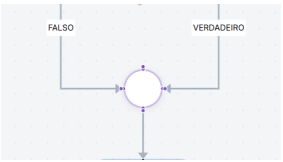


(d) Processo

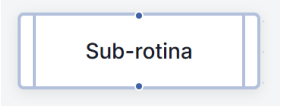


(e) Decisão

(f) Memória



(g) Conector



(h) Sub-rotina

Figura 11 – Blocos implementados no protótipo do sistema
Fonte: Elaborado pelo autor (2026).

4.4.4 Painel de Propriedades

O painel de propriedades é responsável pela exibição e edição das informações associadas aos blocos selecionados no ambiente visual. Nesse componente, o usuário poderá configurar atributos específicos dos elementos do algoritmo, como nomes de variáveis, expressões lógicas, operações matemáticas e mensagens de saída.

Essa abordagem permite separar a representação visual dos blocos das informações utilizadas durante a modelagem estrutural e interpretação do fluxo.

4.4.5 Prototipação da Interface

A prototipação da interface foi utilizada como etapa de apoio ao planejamento visual e estrutural do ambiente proposto. Durante essa etapa, foram exploradas diferentes possibilidades de organização dos componentes da interface, incluindo disposição dos painéis, organização dos blocos e mecanismos de interação visual.

Os protótipos também contribuíram para análise preliminar da usabilidade do sistema, permitindo avaliar aspectos relacionados à navegação, organização visual e fluxo de interação do usuário. A construção inicial dos protótipos foi realizada com apoio da plataforma Lovable IA (Lovable, 2026), utilizada como ferramenta auxiliar na elaboração da prova de conceito da interface.

4.5 Integração com Execução e Teste de Mesa

O sistema proposto foi modelado considerando integração com mecanismos de execução e teste de mesa, permitindo que os algoritmos construídos visualmente possam ser posteriormente interpretados por um módulo de execução.

Neste trabalho, o foco está no desenvolvimento da interface visual, da modelagem estrutural dos algoritmos e dos mecanismos de validação das conexões entre os blocos. Dessa forma, a aplicação fornecerá a estrutura necessária para integração com um mecanismo externo responsável pela execução do fluxo e realização do teste de mesa.

A implementação do interpretador responsável pela execução passo a passo do algoritmo não faz parte do escopo deste trabalho, sendo desenvolvida em projeto complementar por Fajardo (2026). Assim, o sistema proposto atua como módulo responsável pela construção, organização estrutural e exportação dos algoritmos desenvolvidos pelo usuário.

Essa separação permite desacoplar o ambiente de construção visual do mecanismo de interpretação, favorecendo modularidade, reutilização da estrutura de dados e futuras expansões da plataforma.

4.5.1 Integração entre os Módulos

A integração entre o módulo de construção visual e o módulo de execução está prevista para ocorrer por meio da exportação da estrutura do algoritmo em formato estruturado.

Nesse modelo, os diagramas criados pelo usuário serão convertidos em uma representação baseada em grafo direcionado, contendo os blocos do algoritmo e suas respectivas conexões.

Essa abordagem permite desacoplar a interface gráfica do mecanismo de interpretação, mantendo separação entre os componentes responsáveis pela modelagem visual e pela execução do fluxo lógico.

4.5.2 Estrutura de Dados Compartilhada

A estrutura compartilhada entre os módulos será composta por dois elementos principais:

- nós (*nodes*), responsáveis por representar os blocos do algoritmo;
- conexões (*edges*), responsáveis por representar o fluxo entre os blocos.

Cada bloco deverá possuir informações relacionadas ao seu identificador, tipo, posição e propriedades específicas utilizadas durante a execução do algoritmo.

As conexões deverão armazenar informações sobre origem, destino e direção do fluxo, permitindo que o módulo de execução percorra o algoritmo de forma estruturada.

A Listagem 4.1 apresenta um exemplo simplificado da estrutura utilizada para representação do algoritmo.

```
1  {
2    "nodes": [
3      {
4        "id": "node_1",
5        "type": "start",
6        "position": { "x": 100, "y": 50 },
7        "data": {}
8      },
9      {
10       "id": "node_2",
11       "type": "memory",
12       "position": { "x": 100, "y": 150 },
```

```
13        "data": {
14          "variables": [
15            { "name": "n", "type": "inteiro" },
16            { "name": "soma", "type": "real" }
17          ]
18        }
19      }
20    ],
21    "edges": [
22      {
23        "source": "node_1",
24        "target": "node_2",
25        "type": "default"
26      }
27    ]
28  }
```

Lista 4.1 – Estrutura de saída do módulo de construção

Esta estrutura permite que o módulo de execução interprete o fluxo sem dependência direta da interface visual, mantendo a separação de responsabilidades entre os módulos.

5 CONSIDERAÇÕES FINAIS

Este projeto apresentou a modelagem e especificação de uma ferramenta web voltada à construção visual e validação estrutural de algoritmos por meio de diagramas de blocos.

A modelagem da proposta foi desenvolvida considerando o contexto do ensino introdutório de programação, buscando fornecer um ambiente visual capaz de auxiliar estudantes na compreensão da lógica algorítmica e da organização estrutural dos fluxos de execução.

A análise de dados históricos da UTFPR-Guarapuava indicou que abordagens visuais aplicadas ao ensino de algoritmos podem estar associadas à melhoria no desempenho acadêmico de estudantes em disciplinas introdutórias de programação, reforçando a relevância pedagógica da proposta desenvolvida.

A fundamentação teórica permitiu estabelecer relações entre pensamento computacional, representação visual de algoritmos, fluxogramas e teoria de grafos, fornecendo base conceitual para a modelagem estrutural adotada no sistema.

A partir dessa fundamentação, o projeto definiu uma modelagem baseada em grafos direcionados para representação interna dos algoritmos construídos visualmente pelos usuários. Essa modelagem permitirá especificar regras estruturais associadas aos blocos visuais e aos mecanismos de conexão do fluxo, possibilitando definir mecanismos destinados à validação de propriedades relacionadas à continuidade do algoritmo, conectividade dos blocos, controle de bifurcações e formação de ciclos.

Além da modelagem estrutural, o projeto também especificou os principais componentes da interface do sistema, incluindo área de edição, painel de blocos, painel de propriedades e arquitetura visual baseada em grafos interativos.

Desse modo, esse projeto estabelece uma base conceitual e estrutural para futura implementação de um ambiente web educacional voltado ao ensino de lógica de programação, contemplando representação visual de algoritmos, validação estrutural e definição de mecanismos previstos para suporte à execução passo a passo.

Embora este projeto não implemente diretamente o mecanismo de execução e teste de mesa, foi especificada a estrutura de integração responsável por fornecer ao módulo de execução os dados necessários para interpretação do algoritmo construído visualmente. Dessa forma, o sistema proposto atua como módulo de construção e organização estrutural do fluxo algorítmico, permitindo integração futura com ferramentas responsáveis pela interpretação e execução do algoritmo.

Como próximos passos, destacam-se: **a)** implementação completa do ambiente web especificado; e, **b)** desenvolvimento dos mecanismos automáticos de validação estrutural;

Por fim, considera-se que a proposta desenvolvida apresenta potencial para contribuir tanto no apoio ao ensino de algoritmos quanto na investigação de abordagens visuais aplicadas ao processo de aprendizagem de programação introdutória.

REFERÊNCIAS

BOWER, M. *et al.* Improving the computational thinking pedagogical capabilities of school teachers. **Australian Journal of Teacher Education**, v. 42, p. 53–72, 2017. Disponível em: https://api.semanticscholar.org/CorpusID:151346430.

COOK, D. **Flowgorithm - Visual Programming Language**. 2023. https://flowgorithm.org. Acesso em: 25 jan. 2026.

CREWS, T.; ZIEGLER, U. **The Flowchart Interpreter for Introductory Programming Courses**. Bowling Green, 1998.

DIESTEL, R. **Graph Theory**. 5th. ed. Berlin, Heidelberg: Springer, 2017. v. 200. (Graduate Texts in Mathematics, v. 200).

FAJARDO, L. G. Desenvolvimento de um simulador de teste de mesa para apoio ao ensino de algoritmos representados em diagramas de blocos. Projeto de Trabalho de Conclusão de Curso em desenvolvimento. Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná – Câmpus Guarapuava. 2026.

Google LLC. **Blockly: A Visual Programming Editor**. 2024. https://developers.google.com/blockly. Acesso em: 19 fev. 2026.

Lovable. **Lovable - AI Full Stack App Builder**. 2026. https://lovable.dev. Acesso em: 08 maio 2026.

Lucid Software Inc. **Lucidchart Visual Workspace**. 2024. https://www.lucidchart.com. Acesso em: 30 jan. 2026.

MANZANO, J. A. N. G. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores**. 29. ed. [*S.l.*]: Érica, 2019.

MARCZAL, D. Transposição de algoritmos em pascal para representação externa gráfica no contexto educacional. Trabalho de Conclusão de Curso (Graduação) – Universidade Estadual do Centro-Oeste, UNICENTRO. 2007.

MEDEIROS, R. P.; RAMALHO, G. L.; FALCÃO, T. P. A systematic literature review on teaching and learning introductory programming in higher education. **IEEE Transactions on Education**, v. 62, n. 2, p. 77–90, 2019.

MIT Media Lab. **Scratch - Imagine, Program, Share**. 2024. https://scratch.mit.edu. Acesso em: 07 mar. 2026.

PERES, F. F. **Como interpretar (e construir) um gráfico boxplot?** São Paulo, 2022. Acesso em: 14 abr. 2026. Disponível em: https://fernandafperes.com.br/blog/interpretacao-boxplot/.

SHNEIDERMAN, B. Direct manipulation: A step beyond programming languages. **Computer**, v. 16, n. 8, p. 57–69, 1983.

SILVA, A. R. d.; SOUZA, M. V. d. O construcionismo de seymour papert e suas contribuições para a educação. **Cadernos de Educação**, v. 17, n. 34, p. 1–15, 2018. Disponível em: https://revistas.fucamp.edu.br/index.php/cadernos/article/view/2820/1766.

SOUZA, M. V. R. d.; FRANÇA, A. C. C. Um estudo sobre as dificuldades no processo de aprendizagem de programação no curso de análise e desenvolvimento de sistemas na fafica

– faculdade de filosofia, ciências e letras de caruaru-pe. **Revista da Escola Regional de Informática**, v. 2, n. 2, p. 19–27, 2013.

SUN, Y.; LIN, J.; WU, Y. Block-based versus text-based programming: A comparison of learners' programming behaviors, computational thinking skills, and attitudes toward programming. **Computers & Education**, Elsevier, v. 200, p. 104789, 2024. ISSN 0360-1315.

XAVIER, G. F. C. **Lógica de programação**. [*S.l.*]: Senac, 2018.