Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

PENSAMENTO COMPUTACIONAL: APLICAÇÃO WEB COMO FERRAMENTA DE AUXÍLIO NA INTRODUÇÃO DA LÓGICA DE PROGRAMAÇÃO

Paulo Rodrigo Silva Santos

Dr. Diogo Fernando Trevisan(Orientador)

Dourados - MS
2023

PENSAMENTO COMPUTACIONAL: APLICAÇÃO WEB COMO FERRAMENTA DE AUXÍLIO NA INTRODUÇÃO DA LÓGICA DE PROGRAMAÇÃO

Paulo Rodrigo Silva Santos

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso, devidamente corrigida e defendida por Paulo Rodrigo Silva Santos e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 07 de Novembro de 2023.

Prof. Dr. Diogo Fernando Trevisan(Orientador)

S237p Santos, Paulo Rodrigo Silva

Pensamento computacional: aplicação web como ferramenta de auxílio na introdução da lógica de programação / Paulo Rodrigo Silva Santos. – Dourados, MS: UEMS, 2023.
49 p.

Trabalho de Conclusão de Curso (Graduação) - Ciência da Computação - Universidade Estadual de Mato Grosso do Sul (UEMS), 2023.

Orientador: Prof.º Dr.º Diogo Fernando Trevisan

1. Lógica de programação 2. Programação (Computadores). 3. Pensamento computacional 4. Aplicação web 4. Fluxograma I. Trevisan, Diogo Fernando. II. Título.

CDD 23 ed. 005.1

Ficha Catalográfica elaborada pela bibliotecária da Universidade Estadual de Mato Grosso do Sul (UEMS) Aline Perdomo Soutelo, inscrita sob o CRB Provisório n° 714 - 1ª Região.

Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

PENSAMENTO COMPUTACIONAL: APLICAÇÃO WEB COMO FERRAMENTA DE AUXÍLIO NA INTRODUÇÃO DA LÓGICA DE PROGRAMAÇÃO

Paulo Rodrigo Silva Santos

Novembro de 2023

Banca Examinadora:

Prof. Dr. Diogo Fernando Trevisan (Orientador)
Área de Computação – UEMS

Profa. Dra. Glaucia Gabriel Sass
Área de Computação – UEMS

Profa. Dra. Mercedes Rocío Gonzales Márquez
Área de Computação – UEMS

Dedico este trabalho a minha família, que sempre acreditou e me apoiou durante meu percurso da minha graduação. E obrigado aos meus amigos e colegas que sempre permaneceram presentes e me apoiando.

AGRADECIMENTOS

Agradeço à minha família que sempre esteve me apoiando durante todo o processo da minha graduação, em especial minha mãe Maria Aparecida da Silva e meu pai Wilson Raimundo dos Santos mesmo que estando em lugares diferentes sempre estavam em contato comigo dando apoio e as forças quando mais eu precisava. À minha mulher Ana Clara Nascimento Galvão e ao meu filho Leonardo Silva Galvão, pois sempre estiveram comigo sempre me ajudando a superar todas as adversidades.

Agradeço também ao meu orientador e professor, Diogo Fernando Trevisan por me dar a oportunidade de estar sobre sua orientação durante todo o processo deste trabalho, que sempre esteve disposto a me ajudar e me dando a direção correta para eu seguir. Gostaria de agradecer em especial a professora Glaucia Gabriel Sass e ao professor André Chastel Lima por me orientarem e terem me dado a oportunidade de participar do projeto de extensão e ter a experiência repassar os conhecimentos do mundo tecnológico para o público alvo da terceira idade. E também agradecer a todos os professores do curso por me fornecerem apoio e conhecimento que foram fundamentais para minha formação acadêmica.

A todos meus amigos que me acompanharam nesta jornada da minha graduação e que foram as amizades que realmente foram um apoio a contribuir para chegar nesta fase atual da graduação.

Agradeço a Deus por sempre estar me dando forças para nunca desistir do caminho ao qual eu escolhi seguir, pois somente com a força que ele me concedeu tudo foi possível, porque só ele sabe como é árduo o caminho até conquistarmos nossos objetivos.

Muito obrigado a todos!

RESUMO

A lógica de programação é uma habilidade fundamental no mundo atual, especialmente para quem deseja ingressar na área de Computação. Contudo, compreender a lógica por meios de representações textuais nem sempre é tão fácil como se pensa, o que ocasiona em um custo de tempo e esforço. Com base nisso, este trabalho será responsável pela criação de uma aplicação web para criação de fluxogramas, a qual tem uma abordagem interativa e visual para o aprendizado da lógica de programação. Assim, por meio da representação visual, torna-se mais fácil de entender, pois é como usar desenhos para explicar algo, principalmente para aqueles que estão iniciando na área de programação. Afinal, um fluxograma é como uma receita simples para algo no qual você simplesmente vai descrever o passo a passo para cada ação. A aplicação web para criação de fluxogramas foi desenvolvida com êxito, desta forma servindo como ferramenta para auxílio à introdução a lógica de programação.

Palavras-chave: Pensamento Computacional, Fluxogramas, Lógica de Programação, Aplicação Web.

SUMÁRIO

LISTA DE SIGLAS

HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
WWW	*World Wide Web*

LISTA DE FIGURAS

1. INTRODUÇÃO

Estamos vivendo em um mundo da tecnologia, a qual está presente em torno de nós sempre através de celulares, caixa eletrônicos, computadores, chips, entre outros. Aprender como se solucionar problemas como um computador, é uma forma de Pensamento Computacional.

Assim como no pensamento computacional, que tem como um de seus elementos essenciais a decomposição de problemas complexos em partes para facilitar a compreensão, no desenvolvimento para jogos temos uma ferramenta muito poderosa que é sistema *Blueprint Visual Scripting* no Unreal Engine. Este sistema de *script visual* é baseado no conceito de criar elementos de jogo usando uma interface de nós diretamente no Unreal Editor, ou seja, programar por meio de blocos. Uma das maiores vantagens do sistema *Blueprint* é a sua facilidade de uso e acessibilidade para não programadores. Isso permite que artistas e designers participem ativamente do processo de criação de jogos, dando vida às suas ideias sem depender exclusivamente de programadores (UNREAL ENGINE, 2023).

Ao explorar o aprendizado de algoritmos, experimentos empíricos evidenciam resultados e vantagens práticas. Esses estudos oferecem *insights* concretos, destacando a eficácia e os benefícios associados à compreensão e aplicação de algoritmos. Essa abordagem baseada em evidências não apenas enriquece o entendimento teórico, mas também ressalta a relevância prática de incorporar algoritmos no desenvolvimento de soluções para diversos desafios.

Crews e Ziegler (1998) relataram que, após realizarem um experimento, foi mostrado que estudantes iniciantes que buscam aprender algoritmos cometem menos erros, possuem maior confiança e resolvem algoritmos de maneira mais rápida quando se faz uso de fluxogramas de algoritmo.

Os resultados acima ressaltam não apenas a eficácia prática dos fluxogramas, mas também a relevância das abordagens visuais no ensino de conceitos algorítmicos. Essa constatação se alinha intimamente com o pensamento computacional, especialmente no que diz respeito à abstração e decomposição de

problemas complexos. Ao integrar abordagens visuais, como fluxogramas, no ensino de algoritmos, estamos promovendo elementos essenciais do pensamento computacional, enfatizando a importância da visualização e simplificação para a compreensão efetiva dos processos algorítmicos. Essa sinergia entre o uso de ferramentas visuais e os princípios do pensamento computacional contribui não apenas para a aprendizagem de algoritmos, mas também para o desenvolvimento de habilidades fundamentais no contexto computacional.

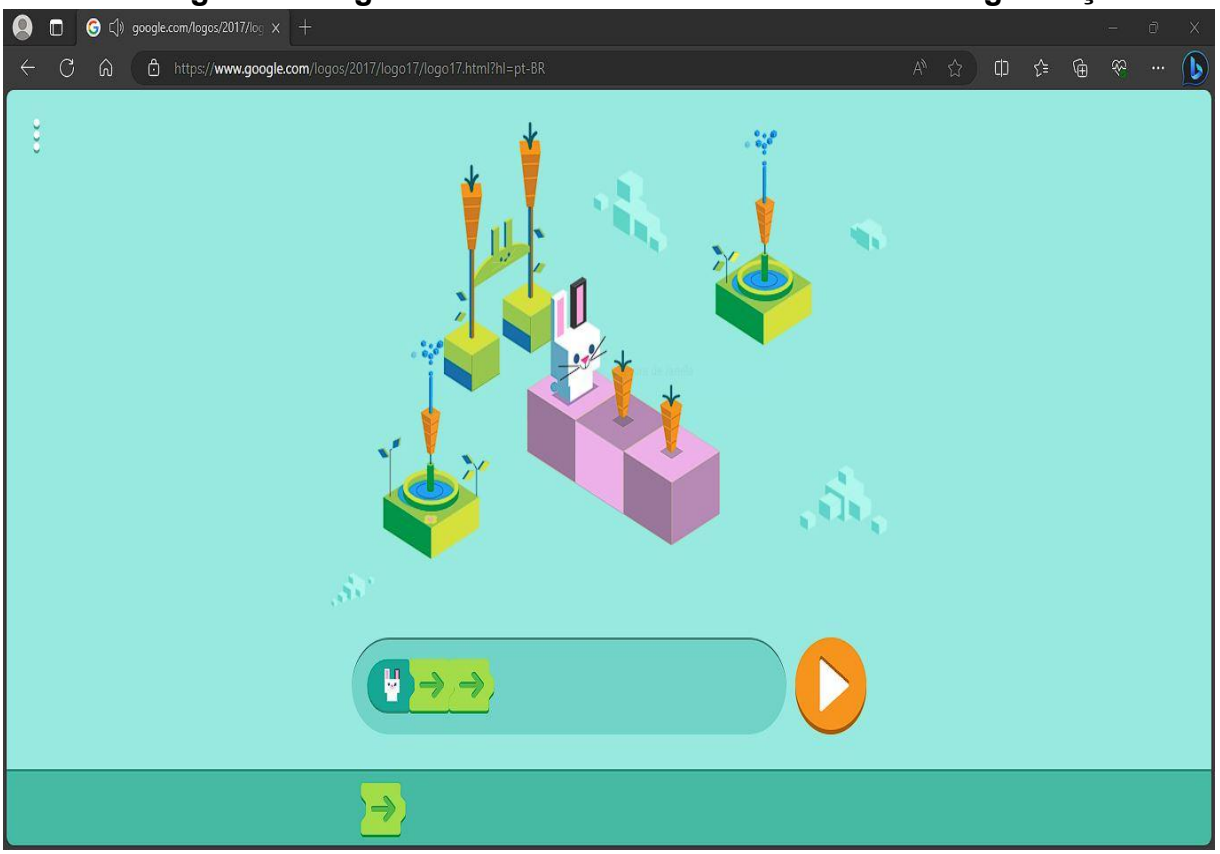
O pensamento computacional está cada vez mais presente na vida das pessoas. Segundo (Wing, 2006) o pensamento computacional visa ajudar a todos e deve ser inserido na vida das crianças juntamente com as habilidades de escrita, a leitura e a aritmética, e não somente profissionais ou estudantes da computação a resolverem problemas pensando como um profissional dessa área. E ainda relata que o pensamento computacional é uma forma de como as pessoas resolvem problemas, e não fazer com que pensem como computadores, e ainda ressalta que os humanos são inteligentes e criativos ao contrário de computadores que são tediosos e enfadonhos.

Logo Blikstein (2008), afirma que o pensamento computacional se trata de como aumentar a habilidade de pensar, compreender, processar informações, raciocinar e resolver problemas de forma mais eficaz e eficiente e utilizar o computador para potencializar essas habilidades do pensamento computacional.

Na literatura, pode-se encontrar diferentes trabalhos com o intuito de facilitar que pessoas sem conhecimento de programação possam resolver problemas usando, às vezes de maneira não clara, criação de algoritmos.

Um exemplo de uso de pensamento computacional é o Doodle apresentado pelo Google, no qual o usuário deve inserir setas de direção para guiar um coelho para coletar cenouras. Estas setas indicam para que ele ande para frente ou gire para uma direção. Em níveis mais avançados aparecem algumas estruturas de repetição. A Figura 1 mostra o Doodle e a paleta de comandos disponíveis no primeiro nível.

Figura 1 : Página *web* da fase inicial do Doodle de Programação.



Fonte: https://www.google.com/logos/2017/logo17/logo17.html?hl=pt-BR Acesso em: 15 out. 2023.

Este trabalho teve como objetivo desenhar e implementar uma ferramenta que sirva como apoio para estudantes de programação. A ideia é que a ferramenta seja visual, ou seja, o estudante criará programas simples a partir de elementos visuais ao invés de códigos. Espera-se também que a ferramenta também seja útil para pessoas que não sejam da área da computação, para que possam fazer pequenos programas e estimulem seu pensamento computacional.

Para este trabalho será criada uma aplicação *web,* por ser acessível de diferentes dispositivos e não necessitar instalação por parte do usuário. O objetivo desta aplicação é atender o interesse daqueles que não possuem conhecimento na área de lógica de programação, permitindo criar programas simples por meio de fluxogramas e deles extrair uma representação textual. No entanto, surge a questão: qual seria o principal desafio a ser superado para garantir a eficácia funcional dessa aplicação? Além disso, como podemos criar a aplicação de maneira a tornar intuitivas todas as suas funcionalidades, permitindo que os usuários compreendam

facilmente cada aspecto e identifiquem claramente o propósito de cada funcionalidade oferecida?

1.1. Objetivos

Como objetivo geral do estudo, construir uma aplicação *web* na qual pode-se criar um fluxograma que permite a qualquer usuário, compreender o processo de passos de um algoritmo desde a sua entrada de dados e passando pelas estruturas de decisões até chegar a saída de dados e no fim a sua representação textual. Especificamente, para a construção da aplicação tem de se conhecer sobre as linguagens que serão utilizadas a fim de ter amplo conhecimento sobre tais, como HyperText Markup Language(HTML), Cascading Style Sheets(CSS), JavaScript. Verificar a exequibilidade da aplicação *web* para que suas funções estejam todas plenas e de forma simples para qualquer usuário compreenda elas.

Com a utilização da aplicação *web* o usuário irá poder criar um fluxograma de algoritmo, poderá proporcionar um aprendizado mais eficiente, uma vez que encontrará um entendimento melhor do seu funcionamento em função do fluxograma criado. Portanto, com a aplicação que permite facilitar o entendimento sobre o funcionamento de algoritmos, a criação da mesma é relevante e de bom aproveitamento principalmente para estudantes iniciantes de algoritmos.

2. REFERENCIAL TEÓRICO

2.1. Pensamento Computacional

Na sociedade digital, a capacidade do pensamento computacional tornou-se cada vez mais importante. Ele não está restrito apenas a programar, em vez disso, ele pode ser usado para enfrentar questões complexas e encontrar soluções eficazes. A capacidade de dividir problemas em partes menores, reconhecer padrões, desenvolver algoritmos e pensar logicamente são todos aspectos do pensamento computacional (WING, 2006).

Estudos mostram que o Pensamento Computacional deve ser incentivado nos estudantes desde o ensino primário (Zanetti et. al, 2016).

A prática do pensamento computacional nos leva a utilizar as técnicas da área da computação, e vem inovando na educação mundial dos estudantes, e oferecendo um conjunto de habilidades para solucionar problemas junto com outras habilidades. Alguns resultados obtidos por escolas Espanholas e Brasileiras, apontam dados estatísticos ao qual os alunos que praticaram o pensamento computacional mostram uma grande melhoria em seus desempenhos (BRACKMANN, 2017). O mesmo ainda ressalta que aprender a programar não implica necessariamente que os estudantes se tornarão programadores ou profissionais na área de computação. Pelo contrário, destaca a importância do aprendizado como uma oportunidade para criar novas possibilidades de conhecimento e desenvolvimento.

BRACKMANN (2017) ainda conceitua que não devemos tratar as tecnologias só como ferramentas para fins de aprendizado, e sim como auxiliar para estruturar problemas e como resolver o mesmos utilizando o pensamento computacional. Abordando o Pensamento Computacional como uma habilidade básica trás muitos benefícios variados como um aumento nas oportunidades de empregos, uma compreensão mais aprofundada do mundo e como aplicar a habilidades não

somente na área de computação, se ter a capacidade e compreender as tecnologia atuais, ter uma eficiência significativa maior, entre outros.

Ao praticar o pensamento computacional, os benefícios transcendem a mera esfera da computação, estendendo-se à melhoria das habilidades de aprendizado, compreensão, análise e resolução de problemas. Diversos experimentos ilustram resultados positivos entre estudantes que incorporam o pensamento computacional em suas práticas educacionais. Essa abordagem não apenas fortalece as competências específicas da área de computação, mas também enriquece as capacidades cognitivas e analíticas de modo mais amplo.

França e Tedesco (2015) relatam que após realizarem um experimento com estudantes e que a forma como foi realizado o experimento trouxe benefício principalmente para a formação dos estudantes, que se evidenciou um desempenho notável nas habilidades de depuração e criação de códigos, que são consideradas fundamentais para a área de computação.

Ao correlacionar esses resultados com as definições de lógica de programação, a aplicação do pensamento computacional não só contribui para o desempenho técnico, mas também promove a formação lógica dos estudantes, destacando a interconexão entre essas duas esferas cognitivas.

De acordo com (BERG; FIGUEIRÓ,2006, p. 22), lógica se trata de como pensar corretamente, ou seja, colocar ordem no pensamento. Para (XAVIER, 2018, p. 20), ele afirma que a lógica serve para ordenar e corrigir os pensamentos ou ações para soluções de problemas.

Lógica de programação é você buscar as melhores sequências de ações para solucionar dado problema, essa sequência de ações é chamada de algoritmo, ou seja, uma sequência de passos para chegar a uma solução conforme o problema que se têm (XAVIER, 2018, p. 21).

Segundo (FRANÇA et al., 2014), o ensino de programação é uma forma para melhorar o raciocínio lógico. No entanto, para ajudar os iniciantes a superarem as

dificuldades de aprendizado da programação, foram usados ambientes visuais, o que tornou o aprendizado mais simples. Com essas ferramentas, os alunos podem se concentrar apenas na lógica do projeto, sem precisar se preocupar com as regras específicas de uma linguagem de programação.

Uma forma de representar um programa ou algoritmo de maneira visual é através de um Fluxograma. Os fluxogramas são representações visuais que empregam formas geométricas para visualizar uma sequência de etapas necessárias para a realização de uma determinada atividade, como afirma (BÚRIGO, 2014).

Tendo como base as afirmações acima, o ensino da lógica da programação é mostrar como conseguir solucionar um dado problema a partir de uma sequência de ações. Além disso, podemos representar esses conjuntos de ações através de fluxogramas, no qual cada ação é constituída de um bloco ao qual pertence a estrutura que compõe o fluxograma.

2.2. Fluxogramas

Os fluxogramas são usados para mostrar o passo a passo de uma atividade usando formas geométricas e setas. Cada forma tem um nome e representa uma ação específica, e as setas indicam a ordem em que fluxo destas ações são executadas (BÚRIGO, 2014).

Fluxograma é uma forma de representação visual de algoritmos, em que temos diversas formas que representam diferentes tipos de ações distintas. É uma forma de representação intermediária se comparado à descrição narrativa no qual se tem o passo a passo de forma natural e o pseudocódigo em que é descrito o código com todo detalhamento, portanto um fluxogramas não é tão abstrato como a descrição narrativa e também não se aprofunda em detalhes de implementação do programa como o pseudocódigo (MATHIAS, 2017).

Representar algoritmos por meio de fluxogramas tem muitas vantagens, como facilitar e aumentar a compreensão do comportamento do algoritmo, até mesmo

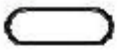
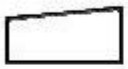
para quem não tem conhecimento sobre, e as pessoas têm maior facilidade em se adaptar a desenvolver algoritmos quando se tem uma representação gráfica do que a ser feito (MEDINA; FERTIG, 2006).

Ainda, apesar de ter muitas vantagens, o fluxograma possui suas desvantagens. Por exemplo, a representação visual de um algoritmo através de fluxograma tem um alto nível de abstração, ou seja, não se tem tantos detalhes de como implementar o código. Também, deve-se conhecer as ações que cada forma geométrica é responsável de executar.

Os símbolos gráficos usados na criação de fluxogramas têm como objetivo principal tornar evidente a linha de raciocínio lógico utilizada por um programador. Isso facilita a compreensão do que se espera que o programa realize, mesmo por parte daqueles que não têm experiência em programação é mencionado por (MANZANO, 2009). Logo se você sabe qual é as ações que cada símbolos, mesmo uma pessoa que não tem conhecimento da área em algoritmo consegue esboçar o comportamento de um programa em sua forma gráfica através do fluxograma.

Com base no mesmo autor, não existe uma definição padronizada para a simbologia do fluxogramas, o que acontece de se ter várias formas de definições distintas. A Figura 2 mostra uma das definições de símbolos que compõem um fluxograma, que é uma adaptação de (CRUZ,2014).

Figura 2: Definição da simbologia de um fluxograma.

Símbolo	Nome	Finalidade
	Terminal	Utilizado para representar o início e o fim do fluxo lógico de um programa. Empregado também na definição de sub-rotinas de procedimento ou função.
	Entrada manual	Utilizado para representar a entrada manual de dados, via de regra pelo teclado do computador.
	Processamento	Utilizado para representar a execução de uma operação ou grupo de instruções que estabelecem o resultado de uma operacão lógica ou matemática.
	Exibição	Utilizado para representar a execução da operação de saída visual de dados em um monitor de vídeo conectado ao computador.
	Documento	Utilizado para representar a execução da operação de saída de dados em um documento emitido por uma impressora na forma de relatório.
	Decisão	Utilizado para representar o uso de desvios condicionais para outros pontos do programa de acordo com situações variáveis.
	Conector	Utilizado para representar a entrada ou saída em outra parte do diagrama de blocos. Pode ser usado na definição de quebras de linha e na continuação da execução de decisões.
	Processo pré-definido	Utilizado para representar um grupo de operações estabelecidas como uma sub-rotina de processamento anexa ao diagrama de blocos (referência a um subprograma externo).
	Linha (com seta)	Utilizado para representar o vínculo existente entre os vários símbolos de um diagrama de blocos. Indica o sentido de fluxo de execução.

Fonte: CRUZ, Antonio José Gonçalves da. Informática para Engenharia Ambiental - Unidade 2: Algoritmos e programação. 2013. Disponível em: http://livresaber.sead.ufscar.br/handle/123456789/1489.

2.3. Linguagens de Programação WEB

Para essa aplicação *web* ser desenvolvida, será de grande importância e se faz necessário ter pleno conhecimentos em algumas ferramentas, tais como as linguagens *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS), *JavaScript*(JS), *React JavaScript* e *React Flow*, as quais serão abordadas agora sobre o que cada uma delas representa.

HTML é uma linguagem de marcação usada para estruturar uma página *web* e o conteúdo a ser adicionado nela, e consiste em uma série de elementos que utilizam tags e cada uma delas tem sua função e que auxilia na construção de uma página *web* segundo (MOZILLA, 2023a).

CSS é uma linguagem de folhas de estilos, ou seja, ele usará o seletor do elemento HTML e vai estilizar dando uma propriedade e um valor, assim proporcionando uma apresentação visual agradável para sua página *web* segundo o (MOZILLA, 2023b). Com ele é possível definir uma cor de fundo, cor da fonte, tamanho da fonte, posicionar os elementos do HTML e uma infinidade de outras possibilidades.

JavaScript é uma linguagem de programação que permite a você criar conteúdo que se atualiza dinamicamente, controlar multimídias, imagens animadas, e tudo o mais que há de interessante, afirma (MOZILLA, 2023c). Ela é usada no *frontend* junto com o HTML e o CSS, o que permite que os usuários possam interagir com a página por meio de botões, ocultar ou mostrar informações, atualizações de elementos sem precisar atualizar a página, transições de imagens e diversas coisas mais. Ainda, é possível realizar o desenvolvimento de *backend*, servidores, *mobile* e games.

O React Flow é uma biblioteca utilizada para trabalhar conceitos em nós. Como exemplos diagramas, fluxogramas, árvores e entre outras aplicações. Com esta biblioteca é possível customizar nós e arestas. Além disso, são oferecidos diversos componentes para deixar a sua aplicação bem mais completa, sendo

alguns deles como minimapas, controles de visualização, fundo da aplicação (REACT FLOW, 2022).

O React JavaScript é uma biblioteca usada para desenvolvimento de interfaces de usuários. Ela é baseada em uma programação orientada a componentes, ou seja, pode-se ter componentes como botões, imagens e textos. A biblioteca também facilita a criação de componentes, compostos por conjuntos de variados componentes. Esta biblioteca tanto serve para criação de aplicações *web* como aplicativos móveis (REACT JAVASCRIPT, 2023).

3. DESENVOLVIMENTO

Neste Capítulo será detalhado o desenvolvimento da aplicação e como se dá o seu funcionamento, quais blocos fazem parte da aplicação e algumas telas da aplicação.

Para essa aplicação a biblioteca React Flow foi de suma importância, visto que ela é uma ferramenta muito poderosa. A mesma dá a possibilidade de se criar blocos, arestas de conexão, mini controles e outros recursos úteis para a criação da aplicação. Além de trazer seus padrões pré-definidos, ela permite que aqueles que fazem uso dela tenham a liberdade de customizar seus componentes e recursos para que se ajuste ao que ela está sendo aplicada. Para a aplicação *web* que foi criada entre componentes e recursos que foram ajustados ou não, se encontra, os nodes, arestas, handles, entre outros.

O primeiro passo para o desenvolvimento da aplicação começou ao escolher onde seria o local que se localizariam os blocos e as formas geométricas para cada ação do fluxograma. Como a ideia era ter a opção de arrastar e soltar os blocos, então foi tomada a decisão de criar uma sidebar para os blocos à esquerda.

A Figura 3 mostra a forma que representa o bloco início. Esse bloco é necessário para que a aplicação seja executada corretamente, pois sem ele quando for realizar a execução da aplicação resultará em um erro falando que está faltando tal bloco. Também é permitida a criação de apenas um único bloco dessa ação e deve estar contido no fluxo do fluxograma criado.

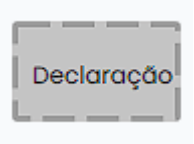
Figura 3: Símbolo início do fluxograma.

Fonte: O Autor.

A Figura 4 mostra o bloco declaração, este bloco representa como as variáveis do fluxograma serão criadas e a tipagem para cada variável. O usuário não

conseguirá criar uma variável se o mesmo não nomear ou definir o tipo dela. Entre os tipos possíveis estão o real, inteiro e caractere. Somente após definir um nome e o tipo para variável ela será criada com sucesso.

Figura 4: Símbolo declaração de variável do fluxograma.



Fonte: O Autor.

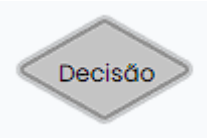
A Figura 5 mostra o bloco de entrada. Este bloco representa a atribuição de valores. Para esse bloco, caso o usuário tente inserir um caractere numa variável do tipo inteiro ou real ele não conseguirá. Foi realizada uma restrição para que só se consiga inserir o valor correto para seus respectivos tipos.

Figura 5: Símbolo da entrada de dados do fluxograma.

Fonte: O Autor.

A Figura 6 mostra o bloco de decisão. Este bloco é responsável por avaliar se uma expressão é verdadeira ou falsa. Ele conta com uma entrada de fluxo e com duas saídas, sendo uma das saídas para *não,* caso a expressão seja falsa, e outra saída para *sim,* caso a expressão seja verdadeira.

Figura 6: Símbolo decisão do fluxograma.



Fonte: O Autor.

A Figura 7 mostra o bloco de processo. Este bloco representa o local no qual é feito o processamento de dados, aos quais contamos com algumas operações sendo elas atribuição, soma, subtração, multiplicação, divisão.

Figura 7: Símbolo do processamento de dados do fluxograma.

Fonte: O Autor.

A Figura 8 mostra o bloco de saída. Este bloco é responsável pela saída de dados. Nele pode-se ter como saída de dados tanto inserindo informações por texto ou variável, ficando a escolha do usuário.

Figura 8: Símbolo da saída de dados do fluxograma.

Fonte: O Autor.

A Figura 9 mostra o bloco de fim de algoritmo. Este bloco representa nada menos do que o fim do algoritmo, e, assim como o bloco início, este bloco também só será criado uma única vez. Para que possa executar a aplicação ele deve está contido no fluxograma criado e fazer parte do fluxo.

Figura 9: Símbolo fim do fluxograma.

Fonte: O Autor.

Como objetivo da aplicação *web* criada é ser uma ferramenta para auxiliar na introdução à lógica de programação, a escolha desses blocos visou em dar a melhor experiência para o usuário, na qual ele possa começar criando fluxogramas simples e dando a possibilidade de criação de outros mais completos, considerando o que está sendo realizado a cada passo e evitando confusões na utilização da aplicação, garantindo uma execução da aplicação que resulte na obtenção correta de sua representação textual.

Com a definição da simbologia usada na criação da aplicação, foi criada a sidebar, apresentada na Figura 10.

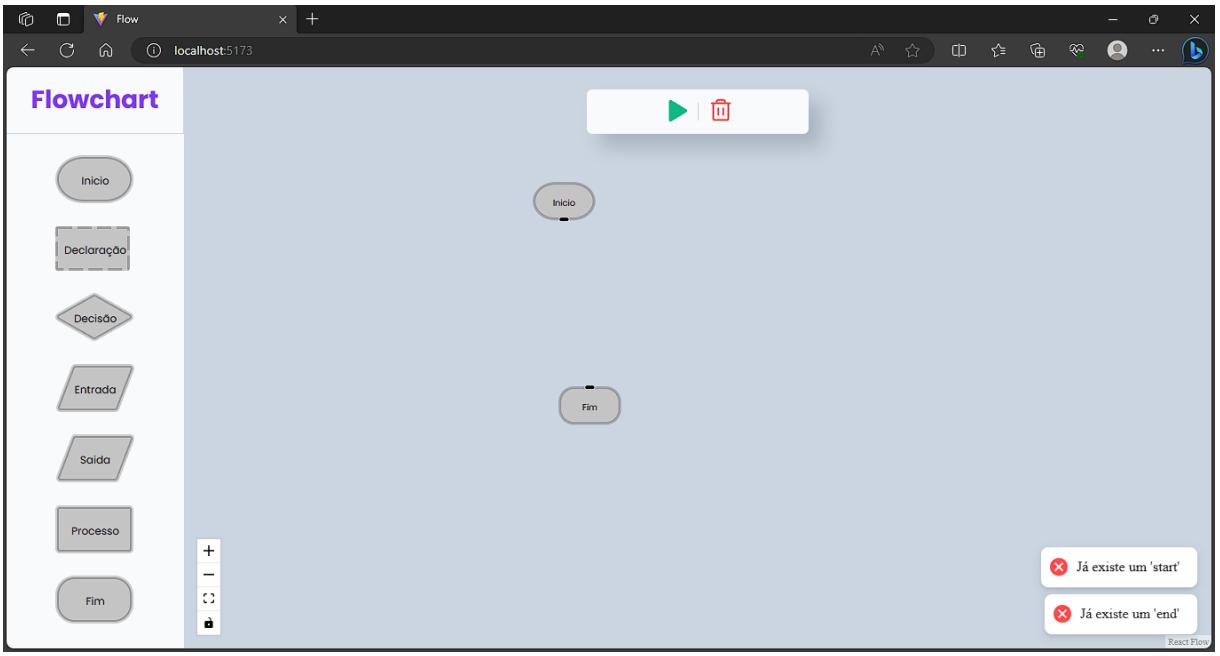
Figura 10: Sidebar da aplicação *web*.

Fonte: O Autor.

Tendo a definição de simbologia, a sidebar foi criada e como próximo passo foi realizada a implementação da funcionalidade *drag-and-drop,* na qual o usuário tem a opção de clicar sobre o bloco que deseja criar e arrastar até onde deseja que seu bloco fique localizado e soltá-lo.

Na Figura 11, é apresentado o que ocorre quando o usuário tenta criar um bloco de início ou fim em situações em que já se possui o tipo do bloco que está sendo tentado criar novamente. Conforme evidenciado na imagem, pode-se notar que já existiam previamente os blocos de início e fim. Ao tentar criar um novo bloco de início ou fim, uma mensagem de erro é exibida, indicando a existência prévia do bloco. Essa restrição foi implementada com o intuito de prevenir que os usuários criem mais de um bloco de início ou fim.

Figura 11: Erro ao se tentar criar dois blocos do mesmo tipo no caso do início ou para o fim.



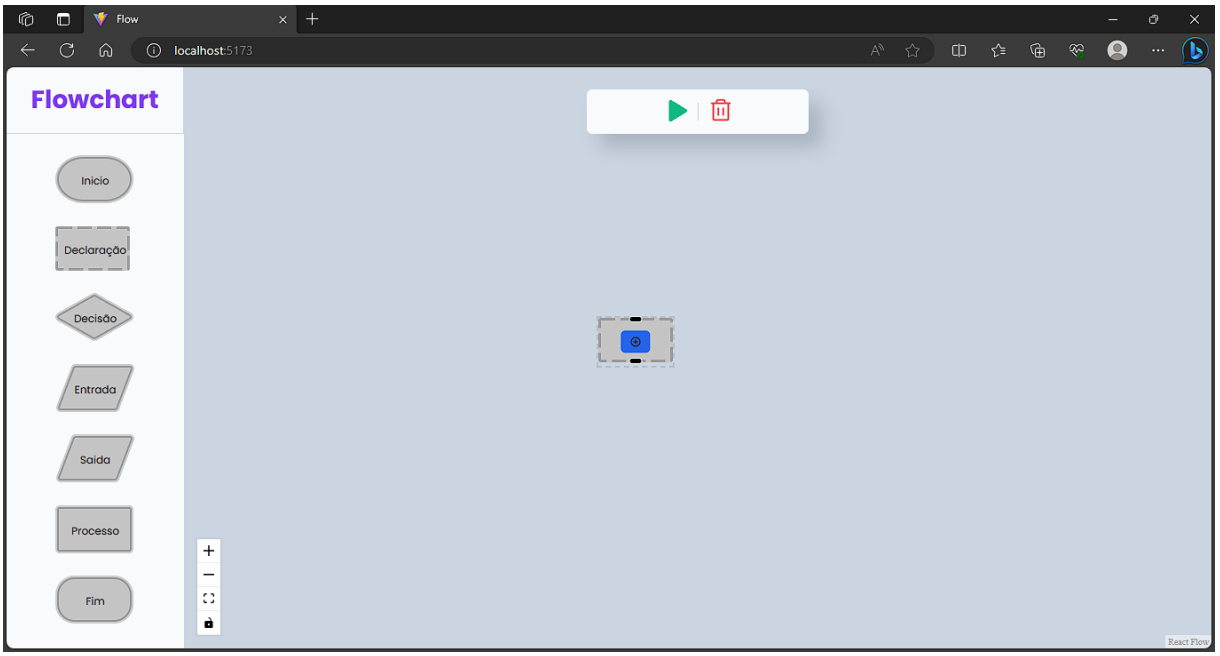
Fonte: O Autor.

Após esta etapa, o desafio era decidir como seria a lógica para coleta de dados para os outros blocos. A solução encontrada foi a implementação de um *modal,* para através dele esta tarefa ser realizada. O *modal* é uma janela flutuante que se sobrepõe à janela atual. Assim, o *modal* será exibido acima do fluxograma, aguardando os dados serem inseridos ou os usuários decidirem fechar ele. Para abrir o modal do bloco, primeiramente se deve selecionar o bloco, e, vai ser exibido um botão de adicionar. Ao clicar neste botão abre-se o *modal*.

As Figuras 12 e 13 mostram o modal do bloco de declaração, o qual quando aberto *modal* mostra que será necessário preencher os dados e clicar no botão

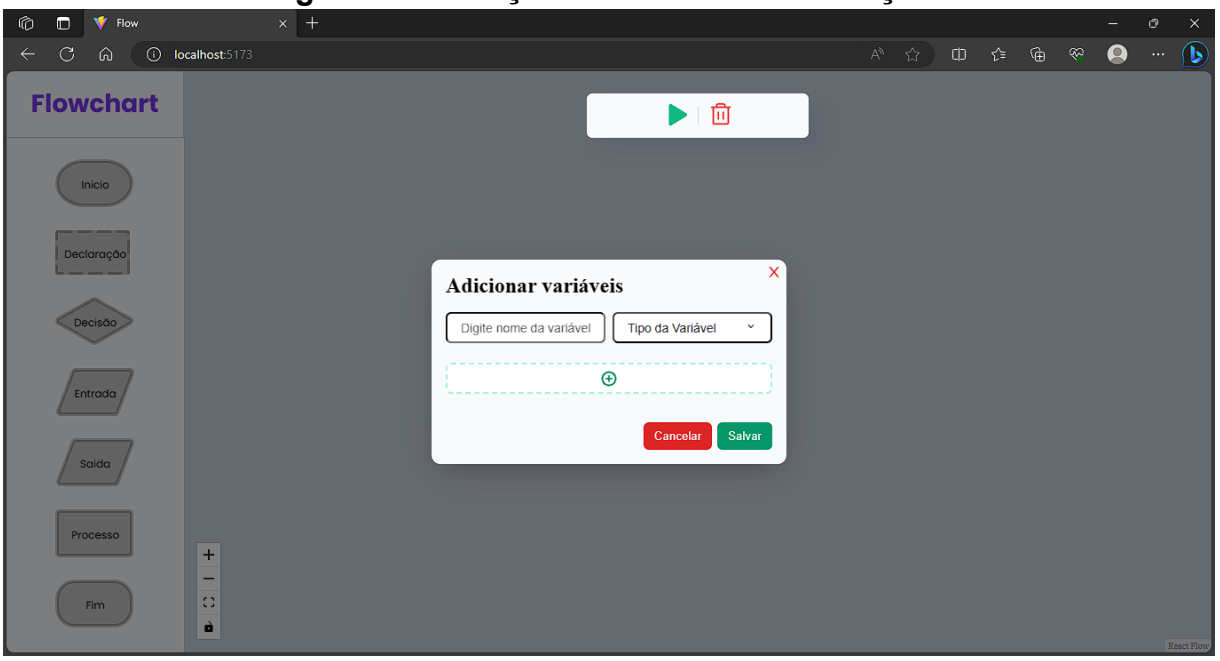
Salvar para a coleta de dados ser realizada com sucesso, ou, pode-se apenas cancelar a coleta dos dados clicando no botão *Fechar,* assim finalizando este *modal*.

Figura 12: Tela mostrando o botão adicionar ao selecionar o bloco.



Fonte: O Autor.

Figura 13: Exibição do *modal* de declaração.

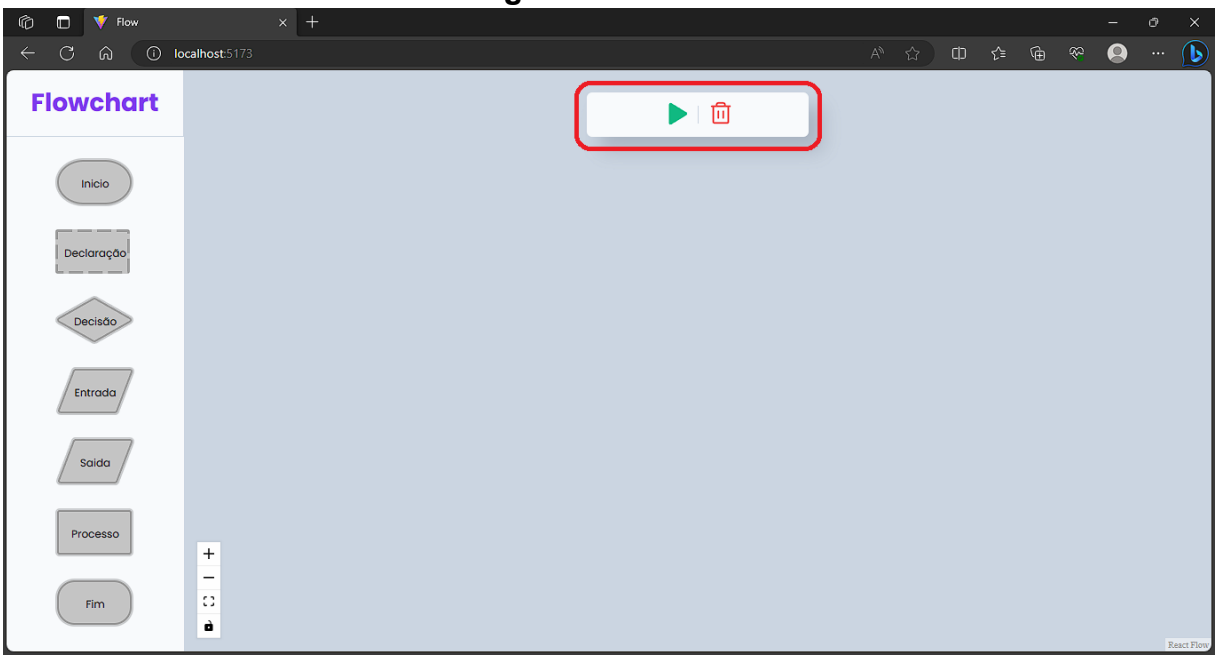


Fonte: O Autor.

Como ilustrado nas Figura 12 e 13, todos os outros blocos restantes, exceto início e fim, têm seu *modal*, cada um com suas peculiaridades de acordo com os dados que se deseja coletar no bloco.

Além disso, foi implementado um componente de controle, o qual é composto por duas funcionalidades. Uma das funcionalidades é a execução da aplicação, localizada na parte superior da aplicação e identificada por um ícone de play de cor verde. A outra funcionalidade é a para exclusão de todos elementos que foram criados na aplicação. Na Figura 14 pode-se observar o componente que contém as duas funcionalidades.

Figura 14: Tela com o componente de controle destacado em vermelho por um retângulo arredondado.

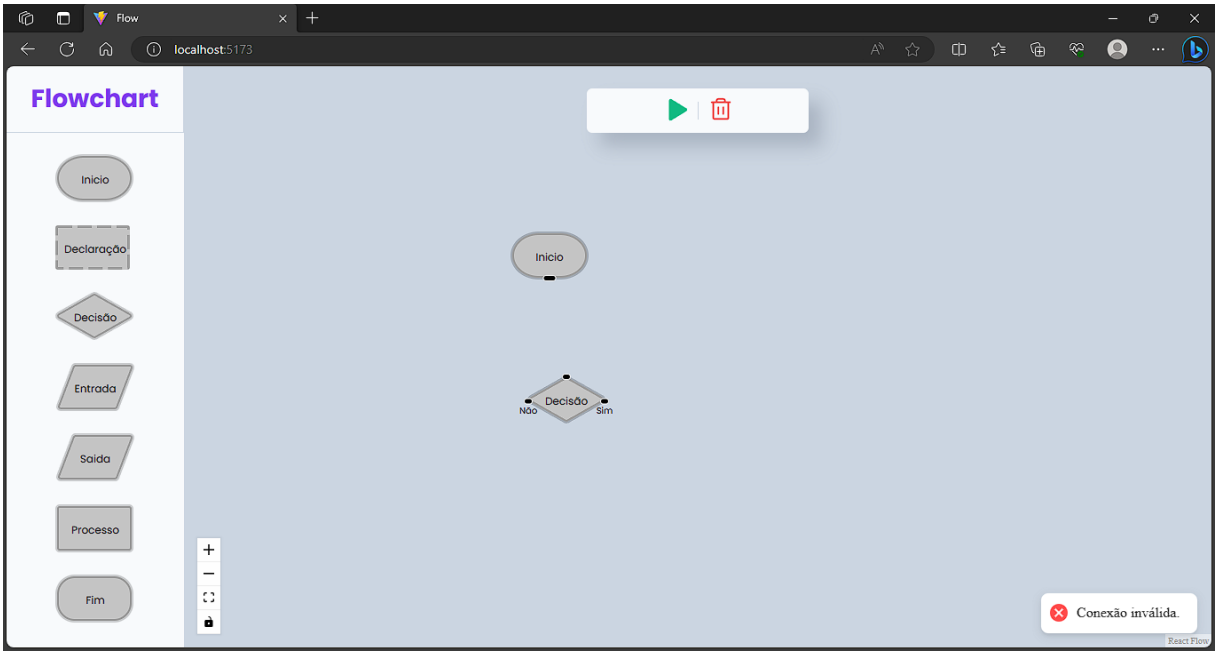


Fonte: O Autor.

Além das funcionalidades básicas descritas, é importante destacar o método pelo qual esses blocos interagem e se relacionam. Os blocos são conectados por meio de arestas que ligam um bloco ao outro. Essas arestas servem como conexões que permitem a passagem de informações e controle entre os blocos. Através dessas conexões, os usuários podem construir fluxogramas de forma intuitiva e eficiente.

Como forma de evitar possíveis erros na hora de criar essas conexões, se fez necessário ter restrições de conexões, ou seja, qual bloco poderá se conectar com outro bloco. Quando o usuário tentar criar uma conexão que não seja possível é emitida uma mensagem de erro avisando que a conexão que se deseja realizar é uma conexão inválida. A Figura 15 mostra a tentativa de realizar uma conexão inválida.

Figura 15: Mensagem de erro de conexão inválida.



Fonte: O Autor.

Após a implementação de todas as funcionalidades, componentes e restrições, a aplicação se encontra totalmente completa e pronta para ser executada.

4. A APLICAÇÃO E TESTES

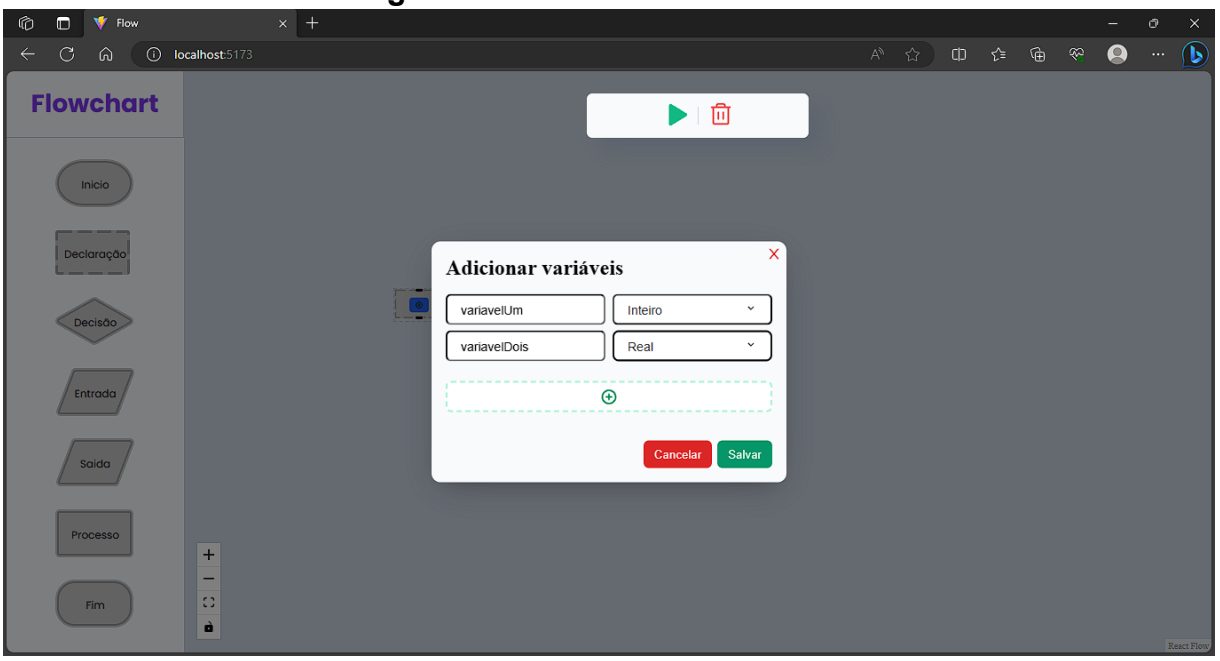
Neste Capítulo será realizada a execução da aplicação. Para este exemplo será criado um fluxograma simples e a partir dele será obtida a representação textual do mesmo. Resumidamente, a forma e a descrição dos passos, de como o fluxograma foi criado são mostrados em detalhes abaixo.

A descrição para o problema que o fluxograma criado é a seguinte:

- Inicio:
- Declare as variáveis inteiras variavelUm e variavelDois.
- Ler os valores de variavelUm e variavelDois.
- Verifica "Se variavelUm é menor que variavelDois?".
    - Se a condição for verdadeira, atribui a variavelDois o valor resultante de variavelDois - 1.
    - Se a condição for falsa, atribui a variavelUm o valor resultante de variavelUm + 1.
- Mostra os valores atualizados de variavelUm e variavelDois.
- Fim:

A Figura 16 apresenta o bloco de declaração, em que está sendo feita a declaração de duas variáveis para fazer parte do fluxograma.

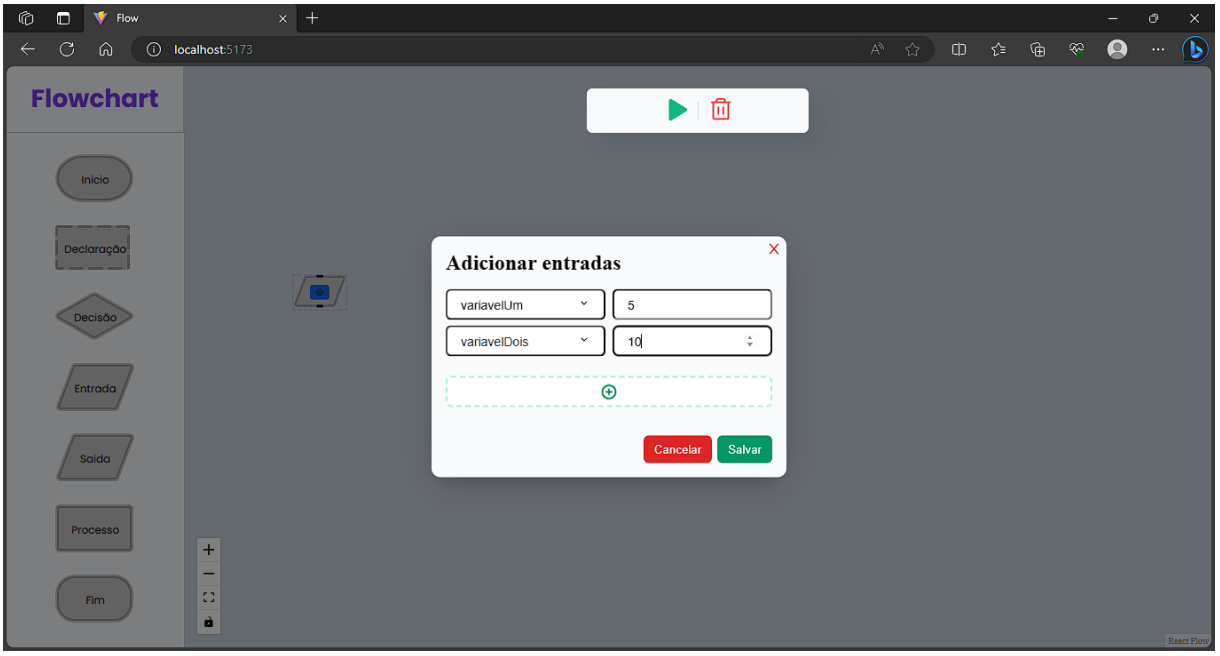
Figura 16: Adicionando variável.



Fonte: O Autor.

Na Figura 17 tem-se o bloco de entrada de dados, no qual está sendo atribuído valor para as duas variáveis.

Figura 17: Atribuindo valores às variáveis.

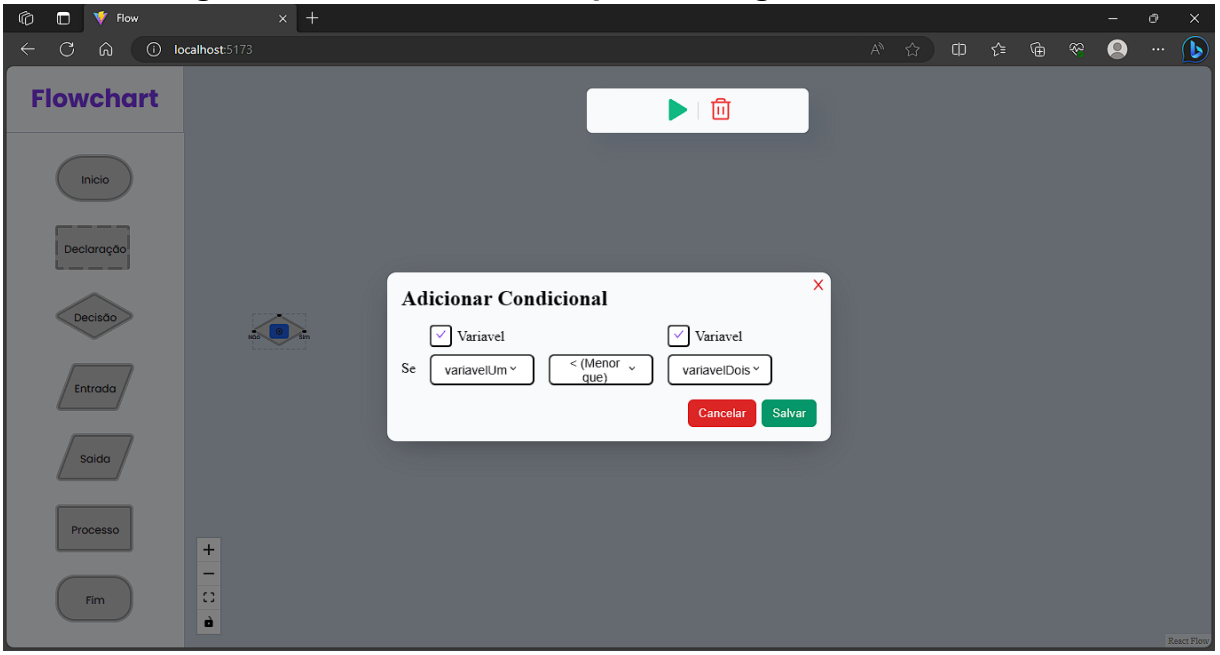


Fonte: O Autor.

Tendo a declaração e atribuição de valores para a primeira e segunda variável, o próximo passo é fazer uma avaliação da expressão lógica. Neste

exemplo foi escolhido verificar se a primeira variável denominada *variávelUm* é menor que a segunda variável denominada *variávelDois*. Assim como é demonstrado na Figura 18.

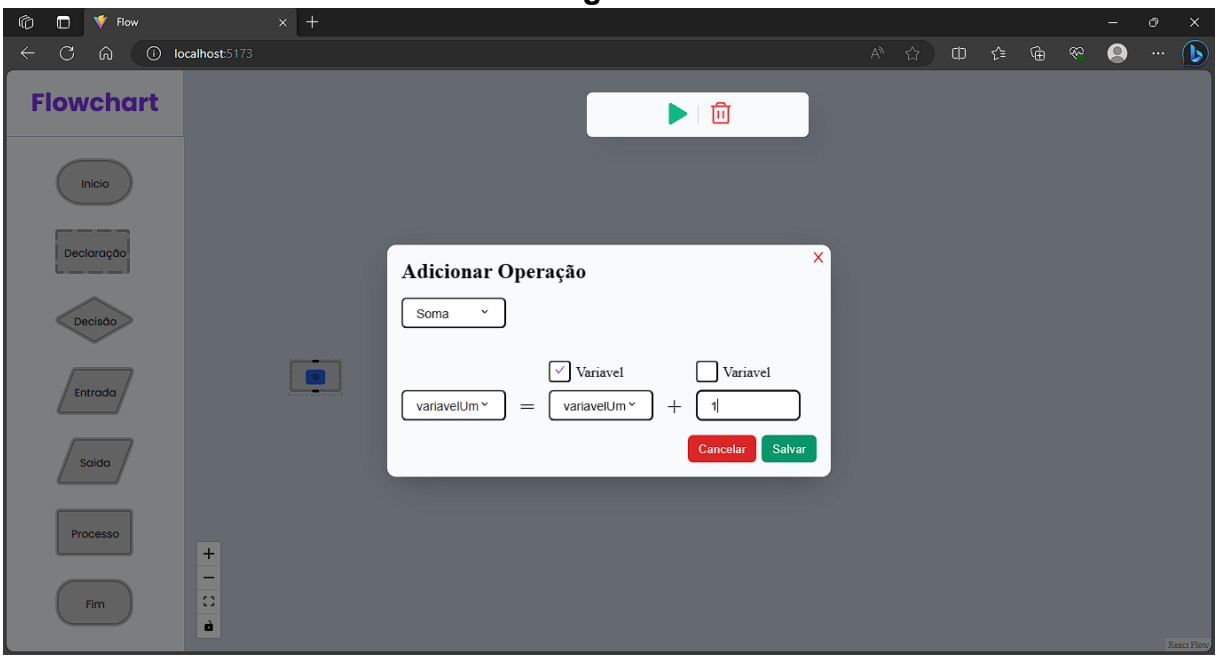
Figura 18: Adicionando a expressão lógica a ser avaliada.



Fonte: O Autor.

Caso a expressão lógica seja falsa, a *variavélUm* tem o processamento de dados que incrementa seu valor em um, como mostrado na Figura 19.

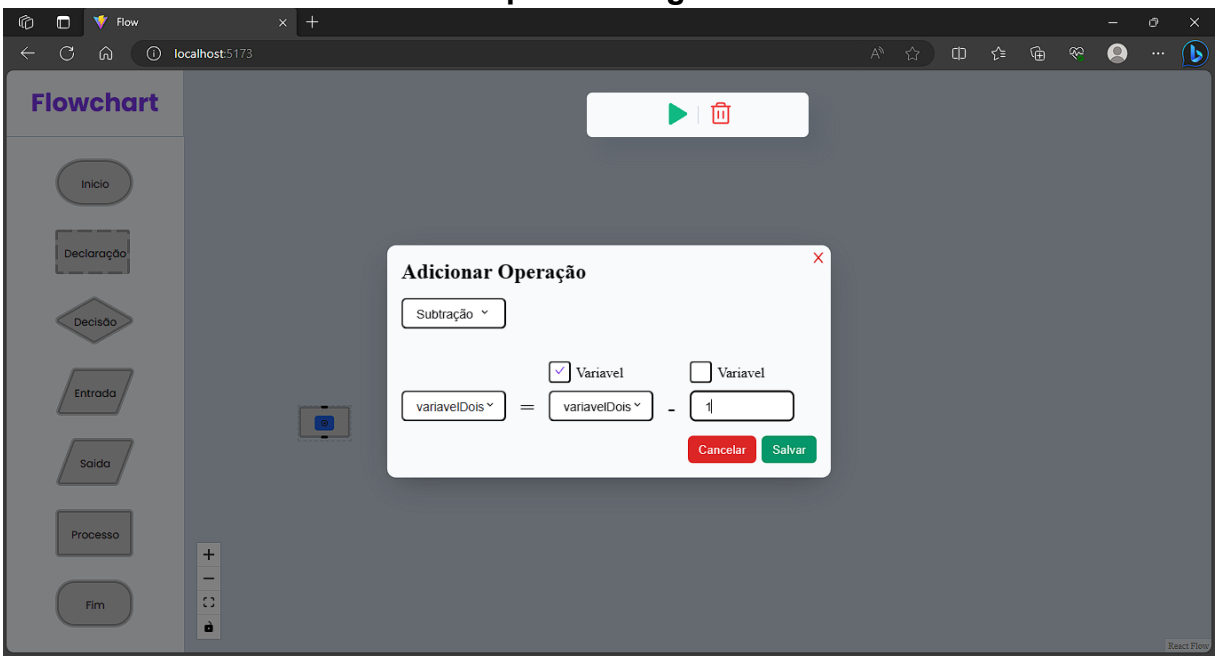
Figura 19: Adicionando o processamento de dados caso seja falsa a expressão lógica.



Fonte: O Autor.

Caso a expressão lógica seja verdadeira, a *variavélDois* tem o processamento de dados que decrementa seu valor em um, como mostra a Figura 20.

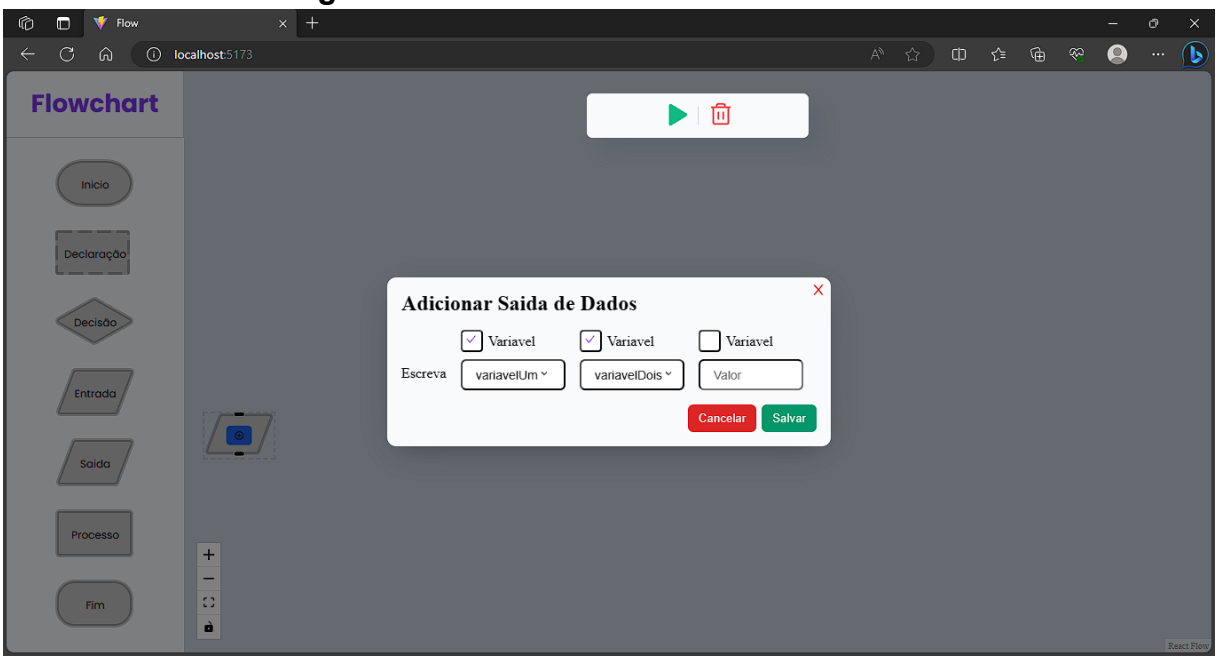
Figura 20: Adicionando o processamento de dados caso seja verdadeira a expressão lógica.



Fonte: O Autor.

O próximo passo para a criação do fluxograma será o bloco de saída que vai conter as duas variáveis para que elas sejam exibidas, como mostra a Figura 21.

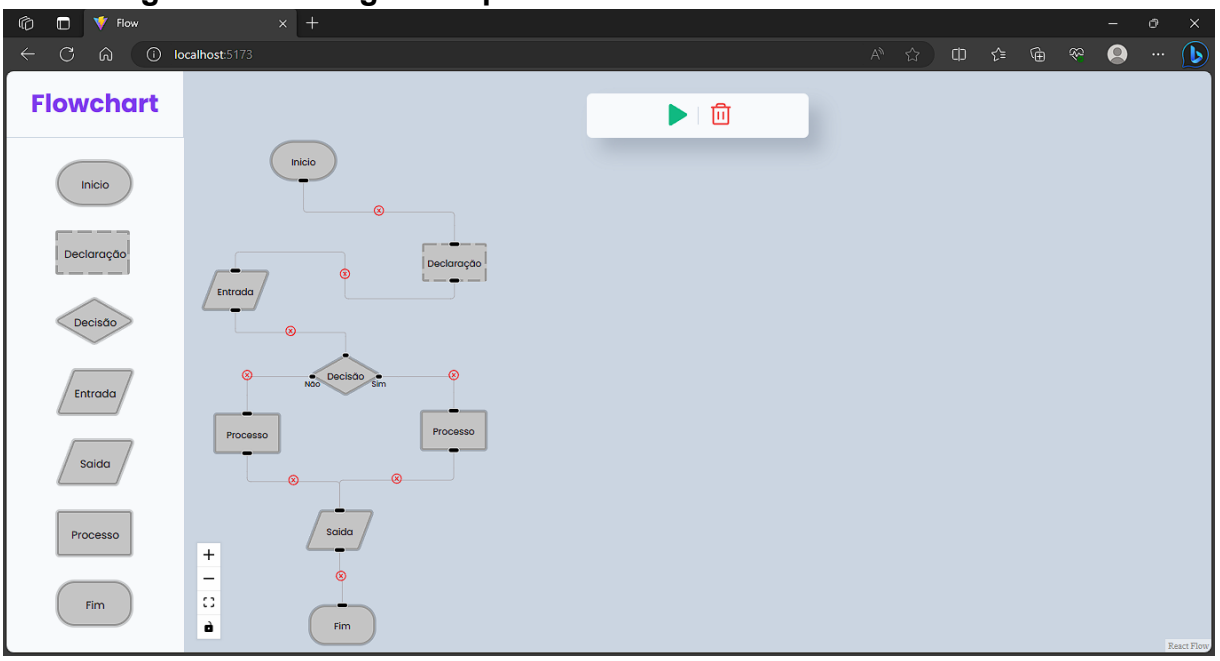
Figura 21: Adicionando a saída de dados.



Fonte: O Autor.

Após feita toda coleta de dados são realizadas as conexões entre os blocos para ser criado o fluxo de dados que o fluxograma deve seguir. Os dados não precisam ser fornecidos primeiramente, podendo-se criar as conexões antes, assim como os blocos. A Figura 22 apresenta o fluxograma completo e pronto para executar a aplicação, devendo-sese deve clicar no ícone de *play* de coloração verde na parte superior.

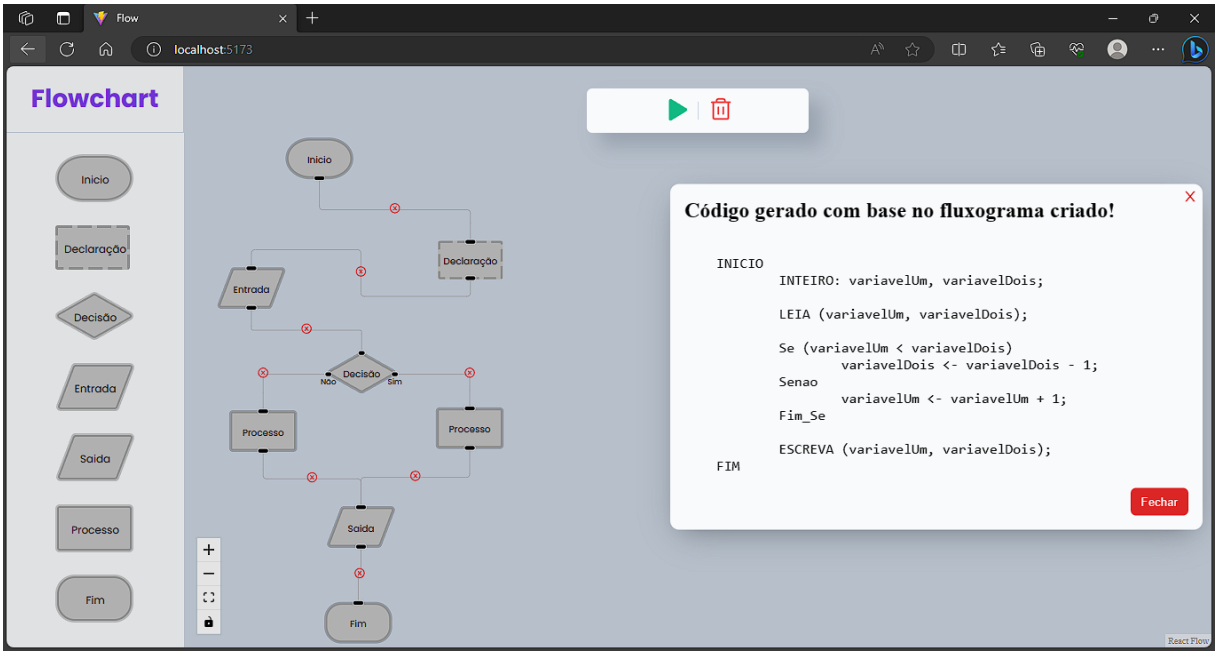
Figura 22: Fluxograma após realizar a conexão entres os blocos.



Fonte: O Autor.

Tendo o fluxograma pronto, será executada a aplicação e o resultado é ilustrado na Figura 23.

Figura 23: Tela mostrando o fluxograma criado e o código gerado dele.



Fonte: O Autor.

Como pod-se ver, a aplicação está funcionando corretamente e trazendo o resultado corretamente.

5. CONCLUSÃO

Com a conclusão deste trabalho, fica evidente que a aplicação web desenvolvida oferece duas funcionalidades principais que permitem aos usuários criar fluxogramas de algoritmos e gerar pseudocódigos correspondentes. Essas funcionalidades são importantes para o processo de simplificação do aprendizado de lógica de programação, pois fornecem duas representações complementares dos algoritmos: visual e textual.

Pode-se afirmar que o objetivo proposto para este trabalho foi alcançado com sucesso. A aplicação funciona de forma eficiente e apresenta os resultados esperados. Essa aplicação poderia ser usada para melhorar o aprendizado de programação e fornecer aos usuários as habilidades essenciais necessárias para compreender alguns dos conceitos básicos da lógica de programação. Logo, é uma ótima ferramenta para a introdução à lógica de programação.

Com base nos resultados deste trabalho, foram identificadas possíveis melhorias que podem ser implementadas em trabalhos futuros. A primeira delas é a introdução de uma funcionalidade que permita a visualização em tempo real dos dados dos blocos por meio de uma janela de visualização. A segunda melhoria proposta é aprimorar a experiência do usuário durante a navegação pelo fluxo do algoritmo. Isso envolve destacar o bloco atual com uma coloração distinta em relação aos demais, e, ao mesmo tempo, gerar o código de forma simultânea no modal que está implementado. Essas melhorias têm o potencial de enriquecer ainda mais a aplicação, proporcionando aos usuários uma experiência mais intuitiva e eficiente.

REFERÊNCIAS BIBLIOGRÁFICAS

BERG, Alexandre; FIGUEIRÓ, Joice Pavek. Lógica de Programação. 3ª ed. Canoas: ULBRA, 2006.

BLIKSTEIN, Paulo. (2008) “O Pensamento Computacional e a Reinvenção do Computador na Educação”. Disponível em: http://www.blikstein.com/paulo/documents/online/ol_pensamento_computacional.html. Acesso em: 20 de set. 2023.

BRACKMANN, Christian Puhlmann. Desenvolvimento do pensamento computacional através de atividades desplugadas na Educação Básica, 2017. Disponível em: http://hdl.handle.net/10183/172208. Acesso em: 6 out. 2023.

BÚRIGO, Bruno Roberto. Flow: ferramenta web para criação de fluxogramas executáveis. 2014. Trabalho de Conclusão do Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2014. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/28349. Acesso em: 05 out. 2023.

CREWS, Thad; ZIEGLER, Uta. The Flowchart Interpreter For Introductory Programming Courses. 1998. Disponível em: http://dx.doi.org/10.1109/FIE.1998.736854. Acesso em: 29 mai. 2023.

CRUZ, Antonio José Gonçalves da. Informática para Engenharia Ambiental - Unidade 2: Algoritmos e programação. 2013. Disponível em: http://livresaber.sead.ufscar.br/handle/123456789/1489. Acesso em 06 out. 2023.

FRANÇA, Rozelma; FERREIRA, Victor; DE ALMEIDA, Luma; DO AMARAL, Haroldo. A disseminação do pensamento computacional na educação básica: lições aprendidas com experiências de licenciandos em computação, 2014. Disponível em: http://dx.doi.org/10.13140/2.1.4355.3926. Acesso em: 02 out. 2023

FRANÇA, Rozelma; TEDESCO, Patricia. Desafios e oportunidades ao ensino do pensamento computacional na educação básica no Brasil, 2015. Disponível em: http://dx.doi.org/10.5753/cbie.wcbie.2015.1464. Acesso em: 08 out. 2023.

MANZANO, José Augusto Navarro Garcia . Construção de Diagramas de Blocos. PROGRAMAR - REVISTA PORTUGUESA DE PROGRAMAÇÃO, 2009.

MATHIAS, Ivo Mario. Algoritmos e programação I. Ponta Grossa: UEPG/ NUTEAD, 2017. 175p.; il. ISBN: 978.85.8024.298.0. Disponível em: http://educapes.capes.gov.br/handle/capes/176223. Acesso em: 06 out. 2023

MEDINA, Marco; FERTIG, Cristina. Algoritmos e Programação - Teoria e Prática. 2ª Ed. Novatec, 2006.

MOZILLA. HTML básico. Mozilla Foundation, 2023a. Disponível: https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/HTML_basics. Acesso em: 01 out. 2023.

MOZILLA. CSS básico. Mozilla Foundation, 2023b. Disponível: https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/CSS_basics. Acesso em: 01 out. 2023.

MOZILLA. O que é JavaScript?. Mozilla Foundation, 2023c. Disponível: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 01 out. 2023.

REACT FLOW. Introduction to React Flow, 2022. Disponível em: https://reactflow.dev/docs/concepts/introduction. Acesso: 05 out. 2023.

REACT JAVASCRIPT. Describing the UI, 2023. Disponível em: https://react.dev/learn/describing-the-ui. Acesso em: 05 out. 2023.

REACT JAVASCRIPT. Thinking in React, 2023. Disponível em: https://pt-br.react.dev/learn/thinking-in-react. Acesso em: 05 out. 2023.

UNREAL ENGINE. Blueprint Overview, 2023. Disponível em: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints. Acesso em: 03 out. 2023

WING, Jeannette. Computational thinking. Communications of the ACM. 49. 33-35. 2006. Disponível: https://dl.acm.org/doi/10.1145/1118178.1118215. Acesso em: 10 jul. 2023.

XAVIER, Gley Fabiano Cardoso. Lógica de programação. 13ª Ed. Senac, 2018.

Zanetti, H., Borges, M., & Ricarte, I. (2016). Pensamento Computacional no Ensino de Programação: Uma Revisão Sistemática da Literatura Brasileira. Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE), 27(1), 21. Disponível: https://doi.org/10.5753/cbie.sbie.2016.21. Acesso em: 25 set. 2023

APÊNDICE A - CONFIGURANDO E EXECUTANDO A APLICAÇÃO

Para desenvolver e rodar a aplicação em um ambiente local, é fundamental instalar uma ferramenta específica no sistema operacional, que é o *Node.js* e o seu gerenciador de pacotes conhecido como npm.

Para a instalação da ferramenta e seu gerenciador de pacotes se faz necessário acessar os seus site oficial no qual é descrito como realizar a instalação dependendo do sistema operacional qual se vai rodar a aplicação, abaixo será disponibilizado os links de sites.

- https://nodejs.org

- https://www.npmjs.com

Uma vez que as ferramentas estão devidamente instaladas e configuradas, você pode prosseguir com a execução da aplicação, seguindo as etapas a seguir:

1. Abrir um terminal dentro da pasta raiz da aplicação.
2. Instalar as dependências ReactJS necessárias da aplicação
   npm install
3. Executar a aplicação *web*
   npm run dev

Após seguir esse passos a aplicação estará rodando localmente já, agora basta acessar ela no navegador, e para isso devemos acessar o seguinte endereço:

http://localhost:5173