CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
( ) PRÉ-PROJETO ( ) PROJETO	ANO/SEMESTRE:

SMALG: SISTEMA PARA MODELAGEM E APRENDIZADO DE ALGORITMOS

Adriner Maranho de Andrade
Dalton Solano dos Reis

1 INTRODUÇÃO

Atualmente, a computação faz parte da realidade e do cotidiano de muitas pessoas, e vem ganhando força**.** Uma prova dessa realidade, é o surgimento de conceitos como *computação ubíqua*, que define a computação como de caráter onipresente na vida das pessoas (GREENFIELD, 2006, p. 11-12), se fundindo ao nosso modo de viver. Além disso, as diversas camadas de diversas áreas abstraídas no processo de desenvolvimento também nos proporcionam a computação de alto nível (COLBURN; SHUTE, 2007), fator importante para a facilitação da produção tecnológica. Essa integração muitas vezes imperceptível da computação com nossas vidas e a facilidade do seu desenvolvimento em determinados cenários, faz com que para a grande maioria das pessoas a complexidade que envolve cada uma das áreas da computação passem despercebidas.

A partir de uma perspectiva educacional, a ciência da computação pode então se demonstrar desafiadora, resultando em um difícil aprendizado, principalmente para aqueles que estão iniciando na área (QIAN e LEHMAN, 2017). Esta área possui altos índices de desistência entre os estudantes, sendo que maioria dos abandonos ocorre nos primeiros dois anos de estudos. Dentre as causas levantadas estavam a baixa qualidade de ensino, além da grade rigorosa e demandas densas (GIANNAKOS, Michail N. *et al*., 2017). Um outro levantamento mostrou uma correlação entre o esforço do aluno e a desistência, destacando que alunos que aplicavam um bom índice de esforço, e consequentemente persistência, eram os que permaneciam com os estudos (GIANNAKOS, Michail N. *et al*., 2016).

Ao lidarmos com a baixa qualidade de ensino, é importante frisar que ela pode resultar em sérias consequências para o estudante durante a formação de sua base de conhecimento. Segundo Qian e Lehman (2017, p.5, tradução nossa) problemas na compreensão conceitual de programação dos alunos podem levar a profundos e significantes equívocos relacionados ao modelo mental do estudante de execução de código e de sistemas de computadores.

Tendo em vista essa realidade, propõe-se um ambiente para auxiliar no aprendizado de algoritmos através da codificação, possibilitando um acompanhamento da execução de código junto com uma representação visual, além da customização de cenários a fim de dar liberdade para moldá-los conforme necessidade do educador.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma plataforma que disponibilize recursos que facilitem o entendimento da lógica codificada pelo estudante, dando a flexibilidade para gerenciar cenários, tendo um foco principal nos cenários que envolvem estrutura de dados.

Os objetivos específicos são:

a) possibilitar o cadastro de cenários de programação, sendo estes baseados em contratos e assertivas para validação de execução, com uma documentação para descrição do cenário;
b) disponibilizar um ambiente de interação do estudante com o cenário criado, permitindo a implementação desses contratos e execução do código;
c) apresentar uma representação visual da execução do código;
d) permitir o acompanhamento da execução do código;
e) implementar por parte do autor, os algoritmos de Array List, Linked List, Hash Map e Bubble Sort em cima da plataforma construída como exemplos e para validação da mesma.

2 TRABALHOS CORRELATOS

A seguir serão apresentados trabalhos correlatos que apresentam semelhanças com as principais características ao trabalho proposto. A seção 2.1 abordará a ferramenta de Halim et al. (2012) que consiste em uma plataforma para auxiliar no aprendizado de algoritmos. A seção 2.2 descreve a ferramenta StarLogo TNG de Klopfer et al. (2009) que permite a modelagem de sistemas descentralizados por meio de programação baseada em agentes. A última seção 2.3 comenta a ferramenta Jeliot 3, Moreno et al. (2004), que apresenta uma representação visual da execução de código programado na linguagem JAVA.

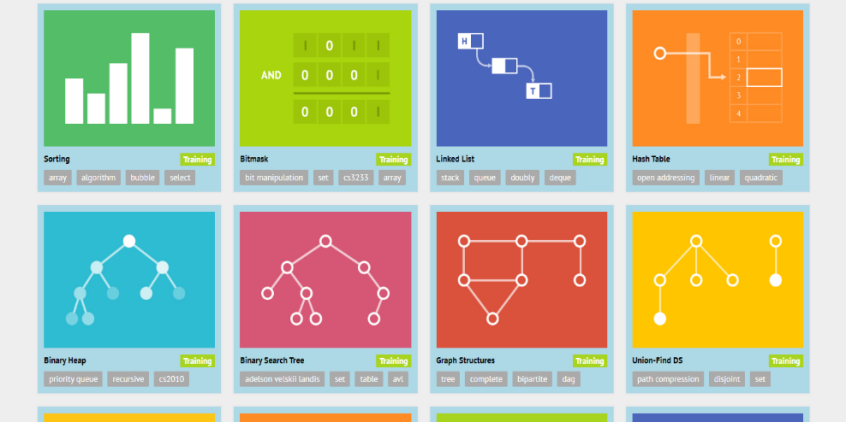
2.1 VISUALGO

A ferramenta de Halim et al. (2012), trata-se de um ambiente unificado e interativo para visualização de algoritmos. Disponível em um website, possui fácil acessibilidade apesar de não ter sido desenhado para funcionar bem com dispositivos com telas pequenas, como smarthphones, conforme enfatiza o autor. Tem como objetivo principal o aprendizado autodidático do aluno de vários algoritmos clássicos e não clássicos da área, em um ritmo que ele se sinta confortável (HALIM et al., 2012).

Ao abrir a ferramenta é apresentada uma tela inicial com os itens disponíveis para estudo conforme ilustra a Figura 1. Cada item é denominado módulo e agrupa uma série de

algoritmos para estudo. O módulo denominado Linked List, por exemplo, aborda algoritmos de fila, pilha, listas duplamente encadeadas, entre outros.

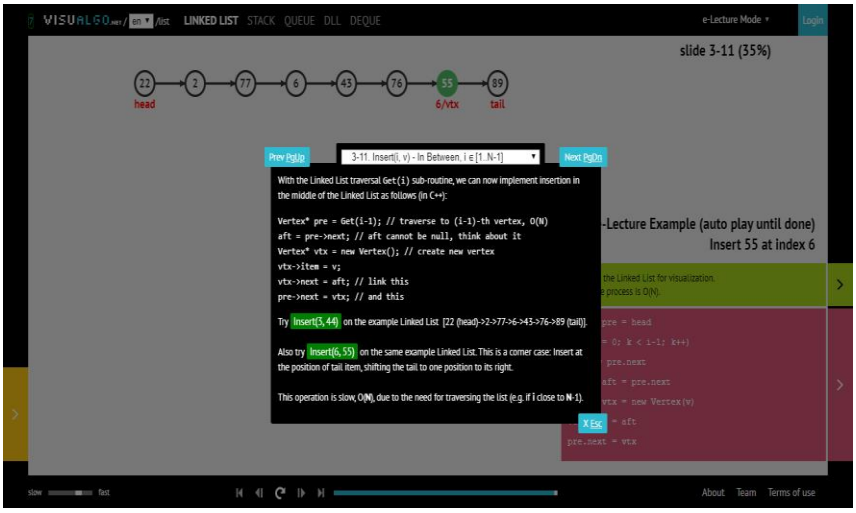
Figura 1 – Tela inicial com a listagem dos módulos



Fonte: Halim et al. (2012).

Ao selecionar um módulo para estudo, é iniciado por padrão o modo assistido, que dá ao usuário explanações sobre o algoritmo em questão, como o seu conceito e a lógica de seu funcionamento. A execução neste modo é de caráter progressivo intercalando entre explicações sobre o algoritmo e partes conceituais e execuções de pseudocódigos passo a passo, tendo no canto superior direito uma indicação do progresso efetuado naquele conteúdo, conforme ilustrado na Figura 2.

Figura 2 – Modo assistido de execução



Fonte: Halim et al. (2012).

O usuário se preferir, pode utilizar o modo exploratório. Nesse modo o cenário fica livre para alterações a partir de ações pré-definidas e específicas para cada algoritmo. Além disso, não são mais exibidas as explicações entre cada passo do algoritmo. Porém, a cada execução que o usuário realiza ainda ocorre a execução do pseudocódigo junto com sua representação visual. Halim et al. (2012, tradução nossa) afirma que “a escolha em ter um

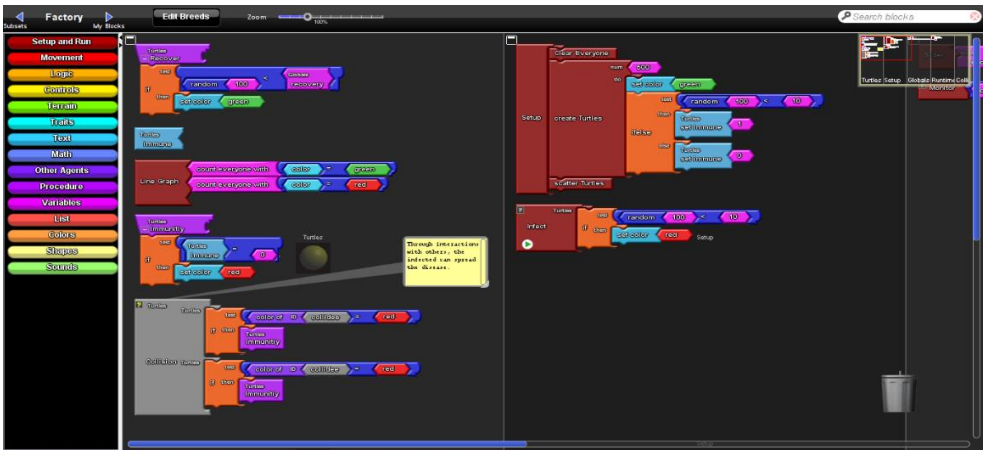
caráter interativo ao invés de visualizações estáticas é para os estudantes obterem mais profundidade a respeito do algoritmo sendo visualizado". Ainda afirma que, os estudantes terão um melhor entendimento do algoritmo se os dados puderem ser inseridos e manipulados a partir deles mesmos (HALIM et al., 2012).

2.2 STARLOGO TNG

A ferramenta para aprendizado de programação criada por Klopfer et al. (2009) possibilita a criação de cenários virtuais pelo usuário onde o mesmo pode interagir construindo sistemas baseados em agentes através de blocos lógicos e visuais, que simplificam a codificação. O método de utilização da ferramenta cria um certo "ciclo de simulação" que combina a engenharia de projeto com o método científico, sendo que a etapa de engenharia consiste no planejamento e construção do cenário, tendo como característica científica a observação e coleta de dados do resultado da execução, gerando novas questões que resultam em alterações no cenário reiniciando o ciclo (KLOPFER et al., 2009).

Ao abrir a ferramenta, são exibidas duas janelas. Uma janela possui o espaço onde são construídos os blocos de código definindo a parte lógica da execução que irá agir sobre os agentes e o cenário. Nela existe uma divisão em raias que permitem a organização o código conforme a responsabilidade de cada segmento programado. No canto esquerdo possui um agrupamento em categorias dos blocos disponíveis para serem utilizados. Entre eles estão: os blocos de configuração, que definem a inicialização do cenário; os blocos lógicos, responsáveis pelas tomadas de decisões; blocos de procedure, que criam rotinas a serem executadas; e os blocos de operações matemáticas. É possível também a criação de blocos customizados, que poderão conter variáveis próprias que se adequarão melhor ao cenário desejado pelo usuário. A Figura 3 ilustra a descrição realizada acima.

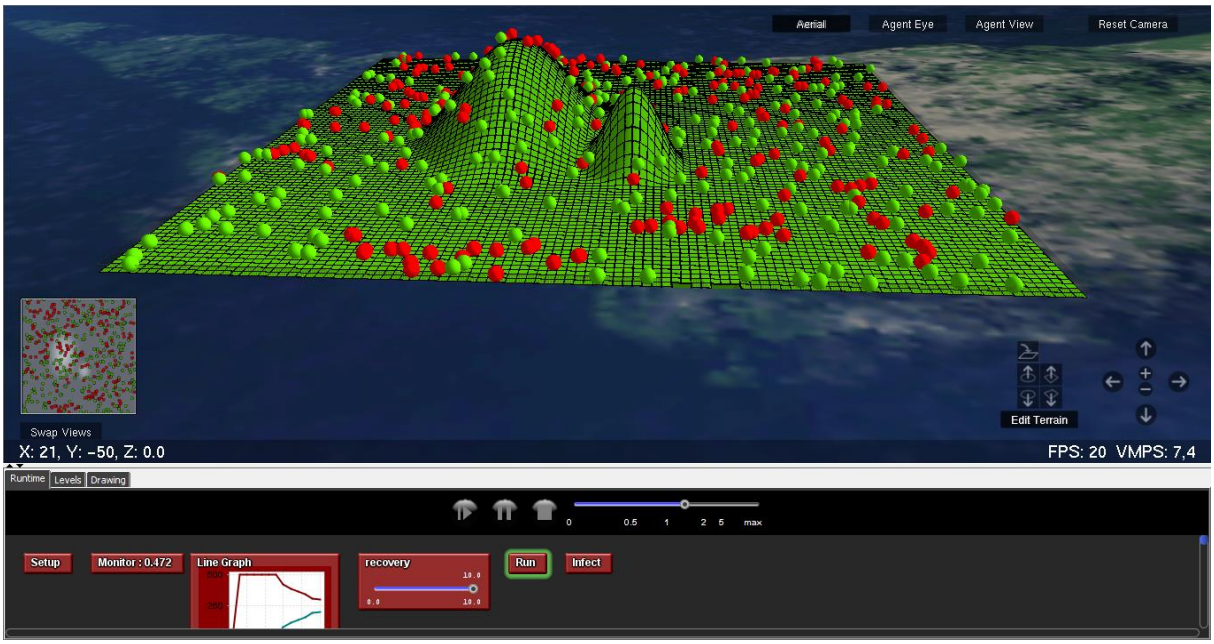
Figura 3 – Janela de configuração de blocos



Fonte: Klopfer et al. (2009).

A outra janela possui o cenário de execução, ou SpaceLand, como denomina a ferramenta. É possível então alterar o mapa, colocando texturas, imagens, entre outros objetos. Também pode-se alterar a configuração do relevo com montanhas ou depressões e alterar o posicionamento dos agentes. Os controles de execução, definidos na etapa de configuração, ficam na parte inferior da janela e são eles os responsáveis por acionar as ações que ocorrerão no cenário. A execução não é executada passa a passo em cima dos blocos construídos, mas na parte central da janela está toda a representação visual do que está acontecendo, conforme demonstrado na Figura 4. Um recurso interessante é a seleção de agente, onde através dos botões disponibilizados no canto superior direito é possível mudar a posição da câmera para acompanhar o agente, ter a visão do agente ou ter uma visão aérea do espaço.

Figura 4 – Janela do cenário de execução



Fonte: Klopfer et al. (2009).

Conforme afirma Klopfer et al. (2009, p.72, tradução nossa),

> “enquanto simulações pré-construídas podem proporcionar aos estudantes uma visualização acessível, um aprendizado imersivo e a oportunidade de analisar os dados a partir de experimentos virtuais, elas não podem prover a liberdade para expressar e explorar seus próprios entendimentos de um fenômeno proporcionado através do uso da simulação [...]”.

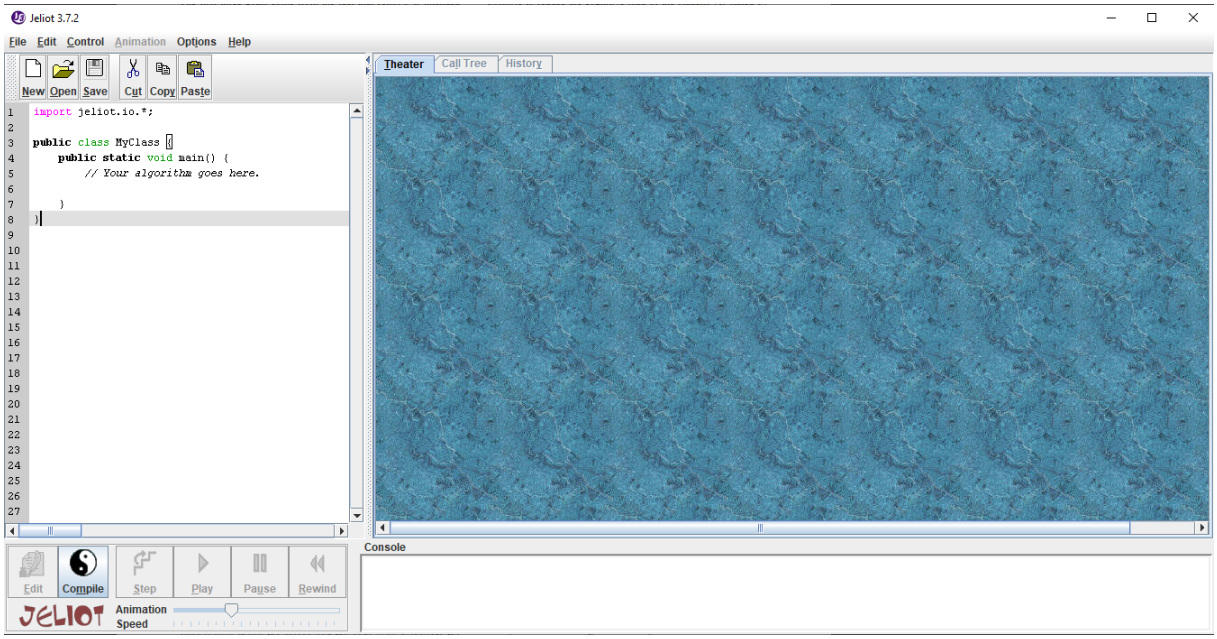
2.3 JELIOT 3

Jeliot 3 se trata de uma ferramenta feita em JAVA com objetivo de ajudar novos estudantes a aprender programação procedural e programação orientada a objetos. A dinâmica

se dá a partir da programação em um código JAVA, que após ser compilado, possui uma representação visual passo a passo da execução do código. Apesar de ser focado para novatos, deveria suportar também a visualização de uma larga quantidade de programas escritos em JAVA (MORENO et al., 2004).

Ao abrir a aplicação temos quatro painéis. Ao lado esquerdo o código a ser feito pelo usuário e compilado para execução passo a passo. No canto inferior esquerdo estão os botões de controle e ao lado, no canto inferior direito as saídas do programa. Em uma região maior no lado direito está o painel de visualização da execução do código. A visualização do código, por sua vez, é subdivida em quatro regiões: uma região para a chamada de métodos, outra região para a execução de expressões, uma terceira região para as constantes, e uma última região para as instâncias criadas na execução. A Figura 5 ilustra o cenário descrito acima.

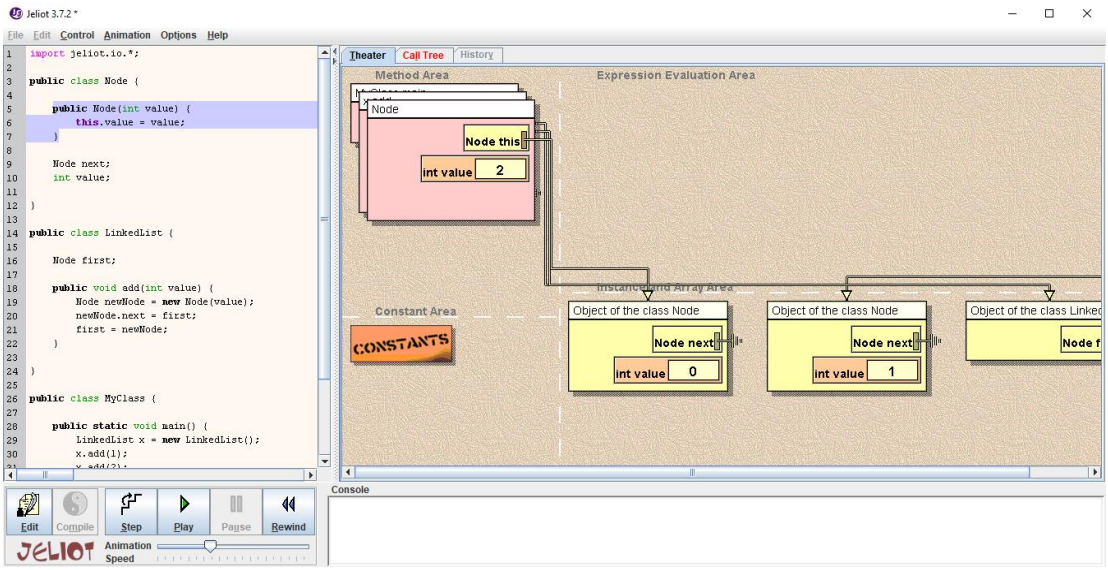
Figura 5 – Tela inicial da ferramenta



Fonte: Moreno et al. (2004).

Moreno et al. (2004, tradução nossa) destacam que “todo o material visualizado é coerente e completo em um sentido em que nenhum dos elementos visualizados aparecem do nada, mas cada um dos elementos possui seu próprio lugar para aparecerem”. Isso pode ser percebido na execução, onde cada etapa do código é reproduzida visualmente, como a criação de uma nova instância, a atribuição de uma nova variável e o relacionamento e as referências que são criadas entre os objetos. A Figura 6 demonstra a execução de um código construído na ferramenta.

Figura 6 – Execução de um código



Fonte: Moreno et al. (2004).

3 PROPOSTA DA FERRAMENTA

Neste capítulo na seção 3.1 será abordada a justificativa para o desenvolvimento deste trabalho, seguido da seção 3.2, contendo os Requisitos Funcionais (RF) e Não Funcionais (RNF), finalizando com a seção 3.3 que descreverá a metodologia de desenvolvimento tal como seu cronograma.

3.1 JUSTIFICATIVA

No Quadro 1 é apresentado um comparativo entre as principais características dos trabalhos correlatos apresentados.

Quadro 1 – Comparativo entre os trabalhos correlatos

Trabalhos / Características	Halim et al. (2012)	Klopfer et al. (2009)	Moreno et al. (2004)
Plataforma	Web	Desktop	Desktop
Execução passo a passo	Sim	Sim	Sim
Representação visual	Sim	Sim	Sim
Estilos dos cenários	Pré-definidos	Dinâmicos	Dinâmicos
Tipo de interação	Acompanhamento de código pré-definido	Codificação	Codificação
Tipo de código	Pseudocódigo	Blocos	Código JAVA

Fonte: Elaborado pelo autor.

Conforme se pode observar no Quadro 1, percebe-se que o trabalho de Halim et al. (2012) é o único disponibilizado em uma plataforma Web. Os trabalhos de Klopfer et al. (2009) e Moreno et al. (2004) são disponibilizados através de instaladores para serem executados no computador de cada usuário. Todos os trabalhos utilizados apresentam execução passo a passo do código além realizar uma representação visual do que está sendo executado. Porém quando nos deparamos com o estilo dos cenários que o usuário pode interagir, o trabalho de Halim et al. (2012) possui cenários pré-definidos, com algoritmos já cadastrados. Os trabalhos de Klopfer et al. (2009) e Moreno et al. (2004), no entanto possuem uma dinamicidade que permite ao usuário criar os seus próprios cenários. Isso se trata de uma estratégia que cada autor escolheu, pois trata-se de uma troca entre uma espécie de tutoria pela liberdade do usuário. Em relação a interação com a ferramenta, o trabalho de Halim et al. (2012) possui pseudocódigos pré-definidos que permitem somente o acompanhamento do aluno com explicações textuais. Algumas pequenas alterações na estrutura são permitidas pelo aluno, mas são ínfimas se comparadas aos trabalhos de Klopfer et al. (2009) e Moreno et al. (2004) que oferecem uma maior liberdade através da codificação. Mas apesar de ambos os trabalhos permitirem o usuário programar, é interessante ressaltar duas diferenças cruciais, Klopfer et al. (2009) trabalha com programação orientada a blocos enquanto Moreno et al. (2004) trabalha com programação procedural e orientada a objetos em JAVA. A programação orientada a blocos é mais limitada, porém mais intuitiva. A programação procedural e orientada a objetos oferece mais liberdade em troca de um maior nível de complexidade.

Diante do exposto, se percebe que possuímos ferramentas com o propósito de auxiliar no ensino de programação, cada uma com suas estratégias e características importantes, mas que se encontram espalhadas.

Dessa forma, o trabalho proposto é uma oportunidade de apresentar uma nova estratégia baseada na combinação de determinadas características já encontradas nos trabalhos apresentados. A utilização de uma plataforma Web oferece à ferramenta um acesso mais simplificado e facilitado, não precisando nenhum tipo de configuração ou instalação, sendo somente necessária o acesso a um website em um navegador. O estilo pré-definido de cenário é interessante pois oferece a quem o desenvolve um melhor planejamento e controle, mas um cenário dinâmico oferece também ao usuário uma maneira de interagir e coletar informações através de suas próprias ações. O trabalho proposto busca então equilibrar essas duas características permitindo o gerenciamento e especificações de cenários sem tirar a dinamicidade do usuário ao interagir com ele através da codificação.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A aplicação desenvolvida deve:

a) permitir o cadastro de usuários (RF);
b) manter cenários de programação criados pelos usuários (RF);
c) disponibilizar no gerenciamento do cenário a definição do contrato que o estudante deverá seguir (RF);
d) disponibilizar no gerenciamento do cenário a definição de assertivas que validarão a execução do código (RF);
e) disponibilizar no gerenciamento do cenário a definição da documentação que irá instruir o estudante sobre o problema (RF);
f) possibilitar no gerenciamento do cenário a definição da implementação correta (RF);
g) permitir a implementação e execução dos cenários de programação por parte do estudante (RF);
h) apresentar uma representação visual da execução do código implementado para o estudante (RF);
i) permitir ao estudante o acompanhamento da execução do código (RF);
j) permitir ao estudante consultar a implementação correta para fins de correção (RF);
k) possuir modelos de cenários prontos com a implementação dos algoritmos de Array List, Linked List, Hash Map e Bubble Sort (RF);
l) ser construído em cima da arquitetura Web (RNF);
m) utilizar a base PostgreSql para armazenamento dos dados (RNF);
n) ter a interface web construída com o framework Angular (RNF).

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: Realizar o levantamento bibliográfico da utilização de ferramentas como auxiliares no processo educacional de computação, tal como os seus benefícios;
b) elicitação de requisitos: detalhar os requisitos e reavaliá-los, se preciso for, com base nas informações levantadas na etapa de revisão bibliográfica;
c) implementação: realizar o desenvolvimento do sistema com base nos requisitos levantados. A plataforma de desenvolvimento será Web, sendo o backend desenvolvido em Java, e o frontend em javascript com o framework Angular. Para

a parte de edição de código dentro do browser será utilizado a ferramenta Monaco Editor e para a representação visual da execução do código a ferramenta Cytoscape;

d) testes: validar o funcionamento da ferramenta como um todo, como a criação de cenários de programação, o acompanhamento de execução e a representação visual do mesmo. Nesta etapa é incluída também a validação dos cenários modelos de Array List, Linked List, Hash Map e Bubble Sort criados.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2020									
	jul.		ago.		out.		nov.		dez.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico										
elicitação de requisitos										
implementação										
testes										

Fonte: elaborado pelo autor.

4 REVISÃO BIBLIOGRÁFICA

Este capítulo descreve brevemente os assuntos que irão fundamentar o estudo a ser realizado: o uso de ferramentas para auxiliar no ensino de Ciência da Computação e o papel da representação visual nesta atividade educativa.

O ensino na Ciência da Computação possui equívocos e dificuldades. A complexidade das tarefas e os encargos cognitivos são apenas alguns dos fatores que influenciam nesse processo. Considerando isso, a criação de ambientes de programação e ferramentas de ensino vem como uma estratégia para auxiliar nesse processo (QIAN e LEHMAN, 2017). Como uma opção de funcionalidade, o uso de representações visuais vem sendo utilizado no ensino de Ciência da Computação desde a década de 80. Sendo assim, algumas experiências já foram adquiridas e ajudam a nortear novas decisões (NAPS et al., 2002).

Estes assuntos serão abordados posteriormente de maneira mais detalhada por Fouh et al. (2012), Grissom et al. (2003), Naps et al. (2002) e Qian e Lehman (2017).

REFERÊNCIAS

COLBURN, Timothy; SHUTE, Gary. Abstraction in Computer Science. **Minds and Machines**, [*S. l.*], n. 17, p. 169–184, 5 jun. 2007.

FOUH, Eric *et al*. The Role of Visualization in Computer Science Education. **Computers in the Schools**, [*S. l.*], v. 29, n. 1-2, p. 95-117, 18 abr. 2012. Disponível em: <http://people.cs.vt.edu/~shaffer/CS6604/Papers/AVpedagogypost.pdf>. Acesso em: 12 abr. 2020.

GIANNAKOS, Michail N. *et al*. Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. **Education and Information Technologies**, [*S. l.*], v. 22, p. 2365–2382, 1 set. 2017.

GIANNAKOS, Michail N. *et al*. Investigating Factors Influencing Students' Intention to Dropout Computer Science Studies. **Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education**, New York, NY, USA, p. 198–203, 1 jul. 2016.

GREENFIELD, Adam. **Everyware**: The Dawning Age of Ubiquitous Computing. 1. ed. [*S. l.*]: New Riders, 2006. 288 p. ISBN 978-0-321-38401-0.

GRISSOM, Scott *et al*. Algorithm Visualization in CS Education: Comparing Levels of Student Engagement. **Proceedings of the 2003 ACM Symposium on Software Visualization**, New York, NY, USA, p. 87–94, 1 jun. 2003. Disponível em: < https://dl.acm.org/doi/pdf/10.1145/774833.774846>. Acesso em: 12 abr. 2020.

HALIM, Steven; KOH, Zi C.; LOH, Victor B. H.; HALIM, Felix. Learning Algorithms with Unified and Interactive Web-Based Visualization. **Olympiads in Informatics**, Vol. 6, pp. 53-68. 2012. Disponível em: < http://www.ioinformatics.org/oi/pdf/INFOL099.pdf >. Acesso em: 5 abr. 2020.

KLOPFER, Eric *et al*. The Simulation Cycle: Combining Games, Simulations, Engineering and Science Using StarLogo TNG. **E-Learning and Digital Media**, [*S. l.*], v. 6, n. 1, p. 71-96, 1 jan. 2009. Disponível em: <https://journals.sagepub.com/doi/pdf/10.2304/elea.2009.6.1.71>. Acesso em: 6 abr. 2020.

MORENO, Andrés *et al*. Visualizing Programs with Jeliot 3. **Proceedings of the Working Conference on Advanced Visual Interfaces**, New York, NY, USA, p. 373–376, 1 maio 2004. Disponível em: <https://dl.acm.org/doi/10.1145/989863.989928>. Acesso em: 12 abr. 2020.

NAPS, Thomas L. *et al*. Exploring the Role of Visualization and Engagement in Computer Science Education. **Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education**, New York, NY, USA, v. 35, n. 2, p. 131–152, 1 jun. 2002. Disponível em: <https://dl.acm.org/doi/pdf/10.1145/960568.782998>. Acesso em: 12 abr. 2020.

QIAN, Yizhou; LEHMAN, James. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. **ACM Trans. Comput. Educ.**, New York, NY, USA, v. 18, n. 1, p. 5, 1 out. 2017.

ASSINATURAS

(Atenção: todas as folhas devem estar rubricadas)

Assinatura do(a) Aluno(a): _____

Assinatura do(a) Orientador(a): _____

Assinatura do(a) Coorientador(a) (se houver): _____

Observações do orientador em relação a itens não atendidos do pré-projeto (se houver):

FORMULÁRIO DE AVALIAÇÃO – PROFESSOR TCC I

Acadêmico(a): _____

Avaliador(a): _____

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?			
	O problema está claramente formulado?			
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?			
	Os objetivos específicos são coerentes com o objetivo principal?			
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?			
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?			
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?			
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?			
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?			
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?			
	7. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?			
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?			
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?			
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?			
	9. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido?			
	10. ILUSTRAÇÕES (figuras, quadros, tabelas) As ilustrações são legíveis e obedecem às normas da ABNT?			
	11. REFERÊNCIAS E CITAÇÕES As referências obedecem às normas da ABNT?			
	As citações obedecem às normas da ABNT?			
	Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes?			

PARECER – PROFESSOR DE TCC I OU COORDENADOR DE TCC
(PREENCHER APENAS NO PROJETO):

O projeto de TCC será reprovado se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou
- pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADO ( ) REPROVADO

Assinatura: _____ Data: _____

[1] Quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.

FORMULÁRIO DE AVALIAÇÃO – PROFESSOR AVALIADOR

Acadêmico(a): _____

Avaliador(a): _____

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?			
	O problema está claramente formulado?			
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?			
	Os objetivos específicos são coerentes com o objetivo principal?			
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?			
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?			
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?			
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?			
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?			
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?			
	7. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?			
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?			
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?			
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?			

PARECER – PROFESSOR AVALIADOR:
(PREENCHER APENAS NO PROJETO)

O projeto de TCC ser deverá ser revisado, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **5 (cinco)** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADO ( ) REPROVADO

Assinatura: _____ Data: _____

[1] Quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.