CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
( ) PRÉ-PROJETO (X) PROJETO	ANO/SEMESTRE: 2020-1

SMALG: SISTEMA PARA MODELAGEM E APRENDIZADO DE ALGORITMOS

Adriner Maranho de Andrade

Prof. Dalton Solano dos Reis – Orientador

1 INTRODUÇÃO

A partir de uma perspectiva educacional, a ciência da computação pode se demonstrar desafiadora, resultando em um difícil aprendizado, principalmente para aqueles que estão iniciando na área (QIAN; LEHMAN, 2017). Ela possui altos índices de desistência entre os estudantes, sendo que maioria dos abandonos ocorre nos primeiros dois anos de estudos (GIANNAKOS et al., 2017). Ainda segundo Giannakos et al. (2017), dentre as causas levantadas estavam a baixa qualidade de ensino, além da grade rigorosa e demandas densas. 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 de execução de código e de sistemas de computadores".

A utilização de ferramentas visuais que ilustram os conceitos de programação e o funcionamento do código têm sido úteis para facilitar neste processo educacional, auxiliando no entendimento dos estudantes (QIAN; LEHMAN, 2017, p. 12). Isso porque na ciência da computação, de acordo com Fouh et al. (2012, p. 96, tradução nossa) "muitos dos seus conceitos centrais referem-se a abstrações. Não se pode ver ou tocar um algoritmo ou uma estrutura de dados [...]".

O uso de tipos abstratos, como interfaces, juntamente com o uso de testes automatizados é uma estratégia a se oferecer para a formulação de uma ferramenta auxiliar ao ensino. Interfaces são consideradas tipos de dados abstratos com uma especificação formal de comportamento, permitindo uma estrutura desacoplada de sua implementação (STEIMANN; MAYER, 2005). Esta estrutura pode então ser utilizada no desenvolvimento de testes, sendo estes responsáveis pela definição de um plano de execução e validação de um código que ainda não possui implementação real, conceito conhecido como *test-first* (NAIK; TRIPATHY, 2008; FUCCI et al., 2017).

Diante do exposto, propõe-se a criação de uma ferramenta para auxiliar no aprendizado de algoritmos e estrutura de dados através da programação e representação visual, sendo essa relação intermediada pela criação de uma biblioteca própria da ferramenta. A especificação de uma interface junto de seus testes automatizados, por parte do educador, proporciona a formulação de um problema, sendo que a implementação de uma interface por parte do estudante, através da biblioteca disponibilizada, caracteriza a solução deste problema.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma ferramenta para auxiliar no aprendizado de algoritmos e estrutura de dados.

Os objetivos específicos são:

a) disponibilizar uma tela para a especificação de uma interface e seus respectivos testes;
b) disponibilizar uma tela que exiba a representação visual da execução de uma implementação, por meio de seus testes automatizados definidos;
c) disponibilizar uma biblioteca própria da ferramenta para ser utilizada na implementação das interfaces e mapear para uma representação visual;
d) possibilitar através de um mecanismo a especificação de novas interfaces e gerenciamento das existentes, flexibilizando o ensino tanto para o educador, quanto para o estudante.

2 TRABALHOS CORRELATOS

A seguir serão apresentados trabalhos correlatos que apresentam semelhanças com as principais características do 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 proporciona uma representação visual através da programação em blocos. A seção 2.3 descreve a ferramenta Jeliot 3 de Moreno et al. (2004), que apresenta uma representação visual da execução de um código programado na linguagem JAVA.

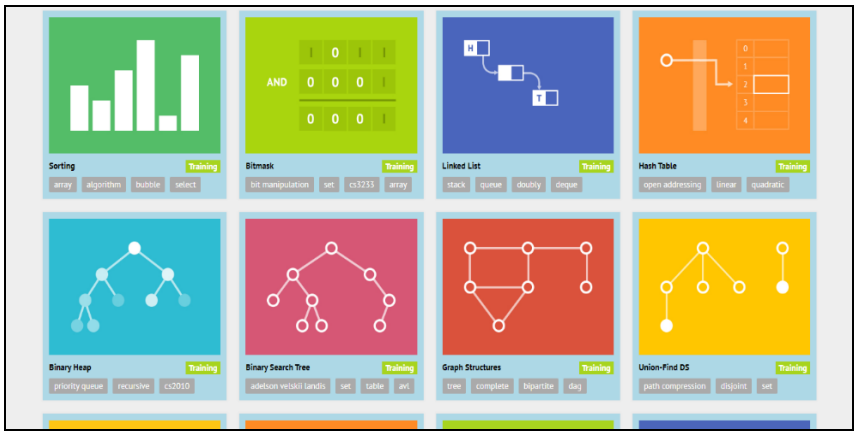
2.1 VISUALGO

A ferramenta de Halim et al. (2012) é um ambiente unificado e interativo para visualização de algoritmos. De acordo com Halim et al. (2012), a ideia de um ambiente unificado consiste em oferecer uma padronização no método de ensino, diminuindo a necessidade do estudante se adaptar às particularidades de cada ferramenta utilizada para este propósito. Disponível em um *website*, possui fácil acessibilidade apesar de não ter sido desenhado para funcionar bem com dispositivos com telas pequenas, como *smartphones*, 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). Apresenta um conjunto de problemas pré-definidos, representados na ferramenta por objetos de estudos em cima de cada conceito. Não é

permitida a formulação de novos problemas ou gerenciamento de existentes para flexibilizar o ensino por parte de um educador. É necessário a atualização do autor para se inserir novos itens de estudo ou alterar o funcionamento de existentes.

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.

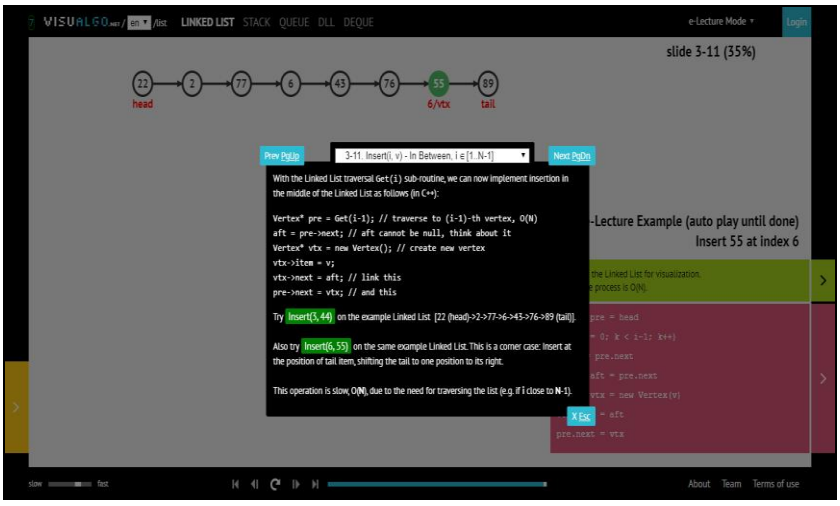
Figura 1 – Tela inicial com a listagem dos módulos



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, 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).

É possível também utilizar o modo exploratório, em que o cenário fica livre para alterações a partir de ações pré-definidas e específicas para cada algoritmo, sem a possibilidade de qualquer tipo de codificação. 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, p. 54, 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

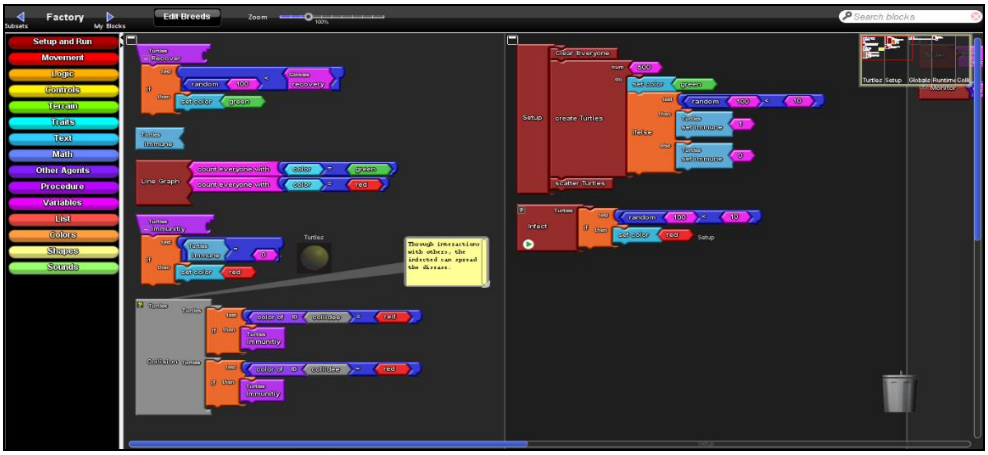
A ferramenta para aprendizado de programação criada por Klopfer et al. (2009), foi desenvolvida para plataformas *desktop* e disponibiliza um ambiente interativo ao estudante, possibilitando a construção de sistemas baseados em agentes através de blocos lógicos e visuais, que simplificam a codificação. O método de ensino consiste na criação de um "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 de aprendizado (KLOPFER et al., 2009).

Ao abrir a ferramenta, são exibidas duas janelas. Uma janela possui o espaço em que ocorre a codificação, realizada através de blocos de códigos. 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. Todo o código programado, será então representado visualmente de modo que o estudante possa compreender e coletar informações a respeito do que foi implementado. A possibilidade do estudante expressar e explorar os seus próprios entendimentos, é algo não proporcionado por visualizações estáticas (KLOPFER et al., 2009).

O caráter livre e interativo da ferramenta, possibilita a formulação de problemas que se adequem a momentos situacionais no ensino. Ao detectar uma determinada necessidade professores podem, por exemplo, sugerir desafios aos seus alunos e auxiliá-los neste processo.

Apesar disso, a ferramenta não dispõe de um ambiente unificado para o aprendizado, responsável por facilitar o gerenciamento, listando problemas conhecidos e provendo uma padronização do método de ensino. A Figura 3 ilustra a tela de codificação da ferramenta.

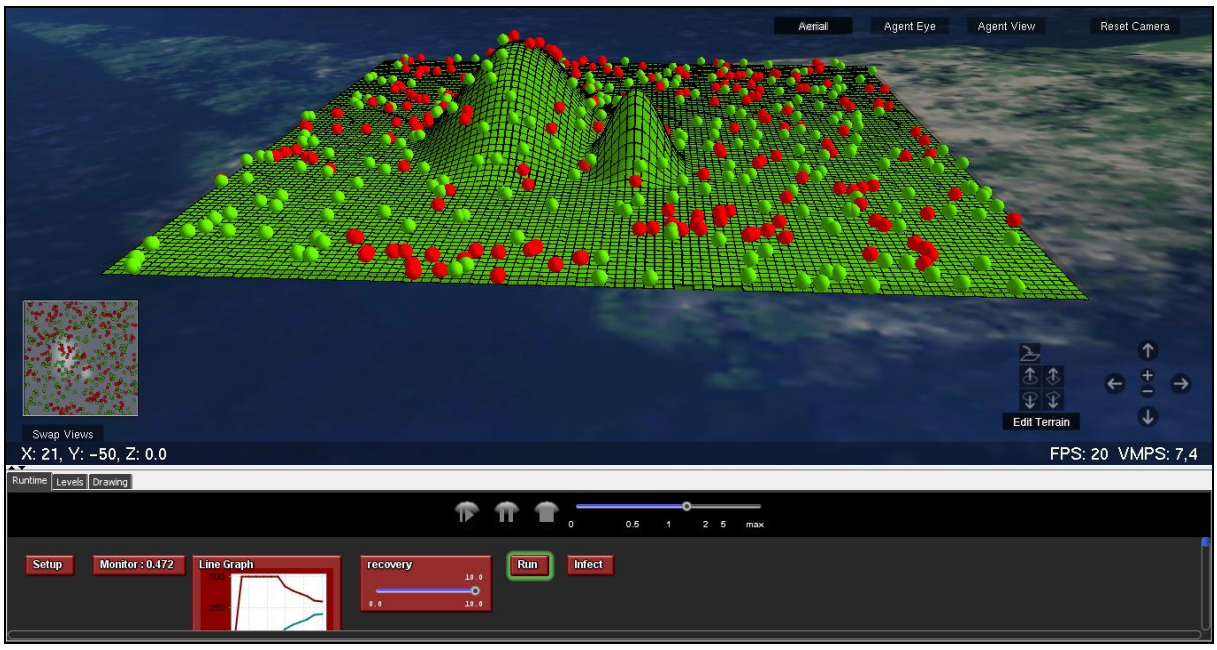
Figura 3 – Janela de codificação de blocos



Fonte: Klopfer et al. (2009).

A outra janela possui o cenário de execução, ou SpaceLand, como denomina a ferramenta. Os controles sobre a execução, definidos na etapa de programação, ficam na parte inferior da janela e são eles os responsáveis por acionar as ações programadas. Na parte central da janela ocorre toda a representação visual do que está acontecendo, realizando o acompanhamento da execução do código, conforme demonstrado na Figura 4.

Figura 4 – Janela de visualização da execução



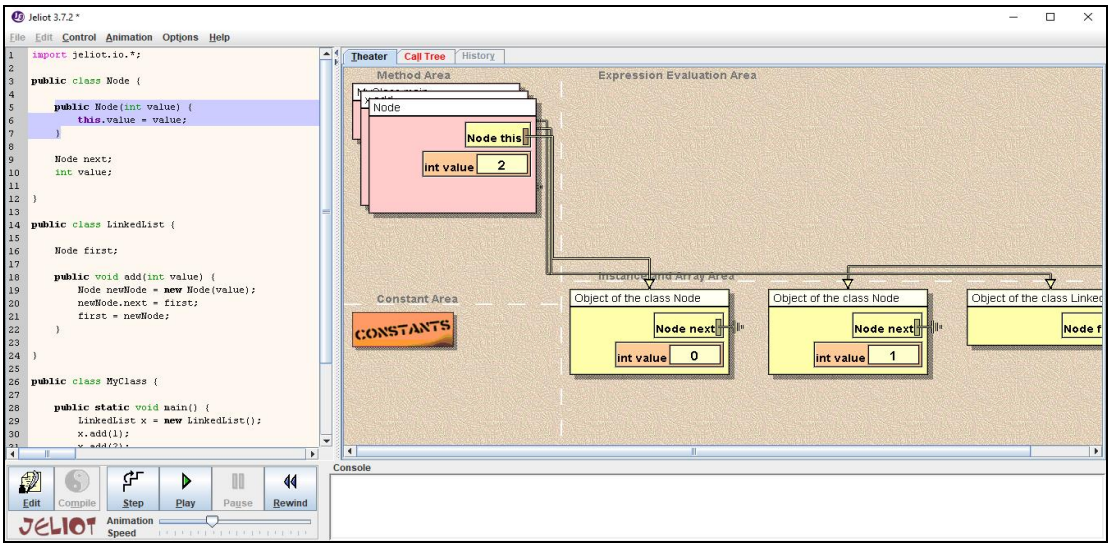
Fonte: Klopfer et al. (2009).

2.3 JELIOT 3

O trabalho Jeliot 3, de Moreno et al. (2004), é uma ferramenta feita em JAVA para a plataforma *desktop*, 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 interpretado, apresenta uma representação visual passo a passo da execução do código. Apesar de ser focado para novatos, suporta também a visualização de uma larga quantidade de programas escritos em JAVA (MORENO et al., 2004).

Dentro da ferramenta, é disponibilizado um painel para implementação do código, sendo que em uma região ao lado está o painel de visualização para acompanhamento da execução do código implementado. A visualização do código, por sua vez, ilustra a chamada de métodos, a execução de expressões, as constantes definidas, e por último, as instâncias criadas durante a execução. A Figura 5 demonstra a execução de um código construído na ferramenta.

Figura 5 – Execução de um código



Fonte: Moreno et al. (2004).

Cada etapa de execução do código é reproduzida visualmente, como a criação de uma nova instância, a atribuição de uma nova variável e as referências que são criadas entre os objetos (MORENO et al., 2004). O caráter dinâmico da ferramenta, caracterizado principalmente pela codificação livre, permite aos professores uma definição de problemas aos alunos. Dada uma determinada situação, é possível formular um enunciado e repassar uma atividade de implementação utilizando a ferramenta como auxiliar ao ensino. Apesar dessa possibilidade, a ferramenta foca apenas no momento de implementação e representação visual, não disponibilizando um ambiente unificado onde seja possível realizar a

formalização, o compartilhamento e o gerenciamento de problemas já definidos, para consequentemente auxiliar na gestão do ensino.

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
Possibilita o acompanhamento da execução do código	Sim	Sim	Sim
Apresenta representação visual da execução	Sim	Sim	Sim
Permite a formulação de problemas	Não	Sim	Sim
Apresenta ambiente unificado para o aprendizado	Sim	Não	Não
Possibilita a programação por parte do estudante	Não	Sim	Sim
Tipo de código utilizado para instrução	Pseudocódigo	Blocos	Código JAVA

Fonte: Elaborado pelo autor.

Como 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 para *desktop* através de instaladores para serem executados no computador de cada usuário. Todos os trabalhos apresentam o

acompanhamento da execução do código, além realizar uma representação visual do que está sendo executado. O trabalho de Halim et al. (2012) trabalha com uma abordagem de problemas pré-definidos. Sua metodologia consiste em trabalhar com cenários controlados, não provendo a possibilidade do professor estender e, por exemplo, adicionar outros tipos de algoritmos conforme sua necessidade. Os trabalhos de Klopfer et al. (2009) e Moreno et al. (2004), já apresentam um caráter dinâmico focando na flexibilização da construção de problemas e provendo a capacidade de se adequar conforme a necessidade do ensino. Por outro lado, o trabalho de Halim et al. (2012) é o único que apresenta um ambiente unificado, onde existe uma espécie de padronização do ensino, trabalhando com uma única linguagem de aprendizado e uma definição formal de problemas.

Em relação ao uso da programação como meio de ensino na ferramenta, o trabalho de Halim et al. (2012) trabalha com pseudocódigos pré-definidos que permitem somente o acompanhamento do aluno com explicações textuais. Os trabalhos de Klopfer et al. (2009) e Moreno et al. (2004) disponibilizam um ambiente de programação, sendo que 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 limitada, porém intuitiva. A programação procedural e orientada a objetos oferece liberdade em troca de um nível maior de complexidade.

Dessa forma, o estudo proposto é uma oportunidade de contribuir para a área de ensino da computação, utilizando uma nova estratégia baseada na combinação das características então descritas. A utilização de uma plataforma *Web* proporciona ao usuário da ferramenta um acesso mais simplificado e facilitado, não precisando de nenhum tipo de configuração ou instalação, sendo somente necessário o acesso a um *website* em um navegador. O acompanhamento da execução junto com uma representação visual facilita ao estudante o entendimento e coleta de informações a respeito do funcionamento do código, auxiliando no aprendizado. A possibilidade de formular novos problemas e modificar existentes, proporciona a flexibilização do ensino. A disponibilização disso em um ambiente unificado, proporciona a padronização e gestão do ensino, direcionando e diminuindo o esforço do estudante em se adaptar a contextos diferentes. No estudo proposto, um problema se caracteriza pela especificação de uma interface (definição formal de comportamento) e definição de seus testes para validação estrutural e funcional, por parte do educador, sendo que a implementação realizada por parte do estudante, caracteriza a solução. Para execução da solução, serão utilizados os testes então definidos. Essa ideia parte do princípio do professor identificar situações excepcionais e complexas que poderiam passar despercebidas pelo

estudante. A codificação a ser realizada pelo estudante terá intermédio de uma biblioteca a ser desenvolvida para o trabalho proposto, sendo esta responsável por mapear a execução em uma representação visual. Essa ideia possibilita uma interação do estudante com a ferramenta de maneira que o que é codificado é representado visualmente. Alterações em seu código implicam em alterações visuais.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A ferramenta desenvolvida deve:

a) permitir o cadastro de usuários (RF);
b) possibilitar a especificação de interfaces (RF);
c) possibilitar o desenvolvimento de testes automatizados para cada interface especificada (RF);
d) permitir o gerenciamento de interfaces especificadas (RF);
e) disponibilizar uma biblioteca a ser utilizada na implementação das interfaces (RF).
f) permitir a implementação e execução das interfaces, por meio da biblioteca disponibilizada (RF);
g) apresentar uma representação visual da execução de uma implementação de uma interface (RF);
h) disponibilizar modelos prontos com a implementação dos algoritmos de *Array List*, *Linked List*, *Hash Map* e *Bubble Sort* (RF);
i) ser construído na arquitetura Web (RNF);
j) utilizar o editor de código Monaco como componente de programação dentro da ferramenta *web* (RNF);
k) utilizar a ferramenta Cytoscape para a representação visual (RNF);
l) utilizar a base PostgreSql para armazenamento dos dados (RNF);
m) 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 o uso da visualização neste processo, além de apresentar elementos que fundamentem os conceitos de abstração por meio de interfaces e a utilização de testes;

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 *frontend* em javascript com o *framework* Angular. Para a parte de edição de código dentro do navegador será utilizado a ferramenta Monaco Editor e para a representação visual da execução do código a ferramenta Cytoscape. A criação de uma biblioteca para utilização na implementação de uma interface irá mapear a execução do código, e em cima disso gerar sua representação visual da execução;
d) testes: validar o funcionamento da ferramenta como um todo, disponibilizando-a para um grupo de professores da área de ciência da computação, com o objetivo de coletar informações a respeito do uso e do potencial educacional do trabalho desenvolvido.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2020									
	ago.		set.		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 a realização deste trabalho. A seção 4.1 comenta a respeito de dificuldades encontradas no ensino da ciência da computação. A seção 4.2 aborda o uso da visualização como forma de auxiliar no processo educacional. A seção 4.3 aborda a utilização de conceitos abstratos na ciência da computação. A seção 4.4 aborda os conceitos por trás da utilização de testes na proposta.

4.1 DIFICULDADES NO ENSINO DA CIÊNCIA DA COMPUTAÇÃO

Todo estudante de ciência da computação deve aprender os conceitos básicos de ciência da computação e programação, mas em muitos casos isso pode ser desafiante, principalmente para alunos que estão nos anos iniciais e nunca tiveram um contato com a área (DEBABI; BENSEBAA, 2016). Um fator que indica isso é um elevado índice de desistência nos dois primeiros anos, sendo que dentre as causas levantadas estavam a baixa qualidade de

ensino, além da grade rigorosa e demandas densas (GIANNAKOS 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 et al., 2016).

Muitos fatores podem contribuir para dificultar o aprendizado, e consequentemente o ensino, dentre eles está a formulação de modelos mentais imprecisos por parte do estudante (QIAN; LEHMAN, 2017). Ainda de acordo com Qian e Lehman (2017, p. 5), o estudante pode enfrentar dificuldades até mesmo no entendimento de estruturas fundamentais como condicionais e estruturas de repetições, não compreendendo perfeitamente os critérios para execução ou o funcionamento. Ambas estruturas são básicas e utilizadas para a construção de algoritmos no modelo procedural. Outra questão importante, é que os estudantes apresentam dificuldades em como desenvolver a solução para um problema e em como criar situações hipotéticas e excepcionais, para então encontrar situações de erro (DEBABI; BENSEBAA, 2016, p. 129).

Esse problema se estende também para conceitos não materializáveis onde não existem analogias diretas na realidade. É possível dizer que o primeiro passo para se aprender programação é partindo de um pensamento humanizado, relacionado com a vida real, para então se adentrar aos conceitos da programação (DEBABI; BENSEBAA, 2016).

Diante dessa realidade, o ensino na área de algoritmos e estruturas de dados já tem recebido atenção. Um estudo de Danielsiek et al. (2012), aponta que alunos do primeiro ano da Universidade Técnica de Dortmund demonstraram equívocos na disciplina durante a realização de exames. Dentre os resultados, estava o algoritmo de ordenação *heapsort* com acerto médio de 43%. Realizaram então um levantamento contendo temas como as estruturas de dados *heap* e *binary tree* (árvore binária). Ao formular testes e apresentarem aos alunos, identificaram novamente outros equívocos, caracterizados, por exemplo, por confusões entre as definições *binary tree* com *search tree* (árvore de busca) e *heap* (DANIELSIEK, 2012).

4.2 O USO DA VISUALIZAÇÃO NO ENSINO

Muitos dos conteúdos centrais da área de ciência de computação estão relacionados com abstrações, contribuindo para certas dificuldades no aprendizado. Não é possível ver ou tocar um algoritmo ou uma estrutura de dados (FOUH et al., 2012). Algumas estratégias então, têm sido adotadas para facilitar neste processo, entre elas o recurso da visualização.

> Boas visualizações de algoritmos trazem algoritmos a vida graficamente representando os seus vários estados e animando a transição entre eles. Eles ilustram estruturas de dados em um caminho natural e abstrato ao invés de focar em endereços de memória e chamadas de funções (FOUH et al. 2012, p. 96, tradução nossa).

De acordo com Sorva et al. (2013, p. 209, tradução nossa) “uma visualização de como um programa de computador é executado pode mostrar o que está normalmente escondido, implícito e automático em algo focal, explícito e controlável”. Ilustrações podem ser realizadas por professores em quadros ou *softwares* de desenho durante as aulas, mas isso normalmente toma tempo limitando a quantidade de cenários que podem ser cobertos com a visualização. Sendo assim, têm se buscado através de ferramentas de visualização a automatização ou semi-automatização desse processo (SORVA et al., 2013, p. 209-210).

Estudos experimentais a respeito da eficiência desse método sugerem que alunos que se engajam com a ferramenta, possuindo um tipo de atividade mais ativa, apresentam melhores resultados do que os estudantes que utilizam a visualização somente passivamente (YUAN et al., 2010). Complementando, Yuan et al. (2010) afirma que não interessa então o quão bem a ferramenta foi desenhada, ela não terá uma efetividade tão grande se não proporcionar a possibilidade de um engajamento ativo do estudante. O modo que o estudante interage com a visualização é mais importante do que as técnicas de visualização utilizadas (SORVA et al., 2013). O conceito de interação não se resume a botões e cliques, mas em ações que o estudante pode realizar alterando o comportamento da visualização.

4.3 ABSTRAÇÃO NA PROGRAMAÇÃO E INTERFACES

Conforme Ganascia (2015, p. 28, tradução nossa) afirma, “a abstração desempenhou um papel crucial na computação”. No princípio, seu propósito foi criar uma generalização descritiva dos dados de um programa de modo que pudessem ser trabalhados sem se preocupar em qual máquina seriam executados, ainda que cada máquina tivesse as suas próprias particularidades (GANASCIA, 2015). Uma demonstração da importância da abstração no desenvolvimento da área de computação é o fato que, de acordo com Ganascia (2015, p. 28, tradução nossa), “computadores são compostos por diversos dispositivos diferentes conectados de uma maneira tão intrínseca e complexa que seria impossível isolá-los na mente humana para representar seu comportamento”. A abstração surge então como forma

de encapsulamento, ocultando informações específicas e disponibilizando informações genéricas (COLBURN; SHUTE, 2007).

Partindo deste princípio, no meio da programação existem as definições de *tipos de dados abstratos*. Tipos de dados, como números e vetores, são definições de estruturas e de operadores de como um dado será representado e manipulado. A inclusão da abstração em um tipo caracteriza a possibilidade de uma definição estrutural e comportamental sem a necessidade de referenciar uma implementação real (COLBURN; SHUTE, 2007; GANASCIA, 2015).

Dentro do grupo de tipos de dados abstratos encontram-se as *interfaces*, que são especificações formais de comportamento, indicando quais operações podem ser realizadas para um determinado tipo. Apesar de sua estrutura estar desacoplada de sua implementação, toda implementação realizada em cima de uma interface deve seguir suas especificações comportamentais (STEIMANN; MAYER, 2005). De acordo com Steimann e Mayer (2005, p.79) os operadores são normalmente restringidos a métodos, mas conceitualmente é possível a definição de atributos.

4.4 CONCEITOS RELACIONADOS A UTILIZAÇÃO DE TESTES

A utilização de testes vai além de somente garantir a execução de um programa sem erros. Naik e Tripathy (2008, p. 7-8) destacam dois conceitos relacionados a definição de um teste: a *validação* e a *verificação*. A validação consiste na confirmação de que um determinado código atende ao seu propósito. A verificação, consiste em determinar se o código satisfaz as especificações definidas antes do início de seu desenvolvimento, como estruturas de código (NAIK; TRIPATHY, 2008).

Existem diferentes abordagens para se desenvolver testes, dentre elas, está o desenvolvimento do teste antes do código a ser testado, conceito conhecido como *test-first* (FUCCI et al., 2017, p. 597). Com este conceito, de acordo com Sommerville (2011, p. 47) “implicitamente se define uma interface e uma especificação de comportamento para uma funcionalidade a ser desenvolvida”. Essa afirmação reforça a possibilidade de implementação de testes através de tipos de dados abstratos. Em casos onde já existe a definição de uma interface explícita, consequentemente, ocorre apenas a especificação e validação de um comportamento.

REFERÊNCIAS

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

DANIELSIEK, Holger *et al*. Detecting and Understanding Students' Misconceptions Related to Algorithms and Data Structures. **Proceedings...**, New York, New York, USA, p. 21-26. fev. 2012.

DEBABI, W.; BENSEBAA, T. Using Serious Game to Enhance Learning and Teaching Algorithmic. **Journal of e-Learning and Knowledge Society**, [*S. l.*], v. 12, n. 2, p. 127–140, mai. 2016.

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.

FUCCI, Davide et al. A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last? **Ieee Transactions On Software Engineering.** [*S. l.*], p. 597-614. jul. 2017.

GANASCIA, Jean-Gabriel. Abstraction of levels of abstraction. **Journal of Experimental \& Theoretical Artificial Intelligence**, [*S. l.*], v. 27, n. 1, p. 23-35, ago. 2015.

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, set. 2017.

GIANNAKOS, Michail N. *et al*. Investigating Factors Influencing Students' Intention to Dropout Computer Science Studies. **Proceedings...**, [*S. l.*], New York, New York, USA, p. 198–203, 1 jul. 2016.

HALIM, Steven *et al*. Learning Algorithms with Unified and Interactive Web-Based Visualization. **Olympiads in Informatics**, [*S. l.*], 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...**, New York, New York, USA, p. 373–376, mai. 2004. Disponível em: <https://dl.acm.org/doi/10.1145/989863.989928>. Acesso em: 12 abr. 2020.

NAIK, Kshirasagar; TRIPATHY, Priyadarshi. **Software Testing and Quality Assurance**: Theory and Practice. Hoboken, New Jersey: Wiley & Sons, Inc., 2008. 616 p. ISBN 978-0-471-78911-6.

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

SOMMERVILLE, Ian. **Engenharia de Software**. Tradução de Kalinka Oliveira e Ivan Bosnic. 9º. ed. São Paulo: Pearson Education do Brasil, 2011. ISBN 978-85-7936-108-1.

SORVA, Juha *et al*. Students' ways of experiencing visual program simulation. **Computer Science Education**, [*S. l.*], v. 23, p. 207-238, set. 2013.

STEIMANN, Friedrich; MAYER, Philip. Patterns of Interface-Based Programming. **Journal Of Object Technology.** [*S. l.*], p. 75-94. jul. 2005.

YUAN, Xiaohong *et al*. Visualization Tools for Teaching Computer Security. **Acm Trans. Comput. Educ..** New York, New York, USA, p. 1-28. jan. 2010.

GLOSSÁRIO

BIBLIOTECA: Conjunto de subprogramas ou rotinas previamente desenvolvidas disponibilizadas para o desenvolvimento de um código.

INTERFACE: Tipo abstrato que não possui dados, mas define métodos e comportamentos. Toda implementação de uma interface deve ser feita de maneira que atenda sua definição. Este termo não se refere à interface gráfica.

TESTES AUTOMATIZADOS: Código implementado com o objetivo de executar e validar a implementação de outro código desenvolvido com o fim de aplicação real.

TEST-FIRST: Metodologia onde testes são implementados antes do desenvolvimento do código a ser testado.

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): _____

StartLogo TNG - Não acredito que esteja deslocado do tema. O conceito da ferramenta vai além de ser apenas algo gráfico. As alterações no relevo e no cenário foram apenas o meio que os autores encontraram para flexibilizar o ensino, e possibilitar dentro da própria ferramenta a formulação de problemas. Temos como características semelhantes ao trabalho proposto: um ambiente de programação, a representação visual do que foi programado, a liberdade de se formular um problema (através da instrução do professor), ... É alterada apenas o modo como a informação é passada ao aluno, mas os princípios colocados no quadro são comparáveis às características propostas pelo trabalho.

Jeliot 3 - Apesar de ser mais antigo (2004), é um bom trabalho e inclusive com um nível de complexidade maior que trabalhos recentes. Foi desenvolvido por várias pessoas e durante anos possuindo duas versões anteriores, sendo ainda referenciado em trabalhos recentes. Foi encontrado inclusive em um trabalho de 2019, que não foi utilizado na proposta por não estar disponibilizado para interação do autor, sendo escolhido então o Jeliot 3.

Outro ponto é que o critério de escolha dos correlatos se baseou no princípio do trabalho estar disponibilizado e possibilitar a interação do autor, para que ele mesmo usasse e compreendesse o que estava sendo proposto, sem se basear somente em um texto escrito, mas também no resultado. Alguns artigos foram encontrados, mas somente descreviam, sem permitir a interação, dificultando a compreensão em determinados assuntos.

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.