UNIVERSIDADE VEIGA DE ALMEIDA – UVA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SYM: UMA APLICAÇÃO PARA AUXÍLIO NO ENSINO DA DISCIPLINA DE ESTRUTURA DE DADOS

ALEXANDRE MOTA SOUZA

RIO DE JANEIRO

2019

UNIVERSIDADE VEIGA DE ALMEIDA – UVA

ALEXANDRE MOTA SOUZA

Monografia apresentada ao curso de Ciência da Computação da Universidade Veiga de Almeida, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): André Lucio de Oliveira

SYM: UMA APLICAÇÃO PARA AUXÍLIO NO ENSINO DA DISCIPLINA DE ESTRUTURA DE DADOS

RIO DE JANEIRO

2019

UNIVERSIDADE VEIGA DE ALMEIDA - UVA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE MOTA SOUZA

SYM: UMA APLICAÇÃO PARA AUXÍLIO NO ENSINO DA DISCIPLINA DE ESTRUTURA DE DADOS

Monografia apresentada como requisito parcial à conclusão do curso em Bacharel em Ciência da Computação.

APROVADA EM:

CONCEITO: _____

BANCA EXAMINADORA:

_____

PROF. MSc ANDRÉ LUCIO DE OLIVEIRA

ORIENTADOR

_____

PROF. MSc CAMILLA LOBO PAULINO

_____

PROF. MSc MARCELO NASCIMENTO COSTA

Coordenação de Ciência da Computação

Rio de Janeiro

AGRADECIMENTOS

Em primeiro lugar, a Deus, por ter me auxiliado e me dado forças durante todo o curso, me permitindo chegar até aqui.

À minha mãe, especialmente, que cuidou de mim com extremo carinho no decorrer de toda a minha vida e me colocou no melhor caminho possível, me dando condições para conquistar o emprego que sonhávamos e, consequentemente, para cursar esta faculdade, mas que, infelizmente, devido à violência que assola sobretudo o estado do Rio de Janeiro, foi repentinamente retirada de mim e não pôde estar presente para dividir comigo esta conquista.

Ao meu pai, pela compreensão e pelo apoio no sustento na maior parte de minha vida, trabalhando durante muitos anos, ajudando sempre que possível até os dias de hoje, e ao meu irmão, por estar sempre ao meu lado em todos os momentos.

Ao meu orientador, André Lucio, pelo enorme auxílio que prestou até aqui, até mesmo quando ainda não havia se tornado oficialmente meu orientador, me atendendo prontamente em todas as vezes que foram necessárias e me guiando ativamente durante todo o desenvolvimento deste trabalho.

Ao meu amigo Heitor, por ter estado ao meu lado em boa parte do tempo durante os cinco últimos períodos, onde sempre mantivemos uma ótima amizade, nos ajudando mutuamente, e por ter demonstrado ser um verdadeiro amigo, inclusive nos momentos difíceis.

A todos os outros amigos que fiz no decorrer do curso, e a todos os professores que fizeram parte da minha formação.

RESUMO

Tradicionalmente, o ensino nas escolas e universidades, na maioria das vezes, é feito de forma que o aluno somente recebe as informações passivamente, sem possibilidade de interagir com os conteúdos. O conhecimento normalmente é transmitido de forma teórica e são passados exercícios de fixação em alguns casos, o que pode não ser suficiente para determinados conteúdos extremamente práticos, que demandam uma forma mais clara de visualização e uma forma de interação com o conteúdo. As metodologias ativas, como a simulação, podem auxiliar no aprendizado do aluno, já que tornam possível a interação do aluno com o conteúdo. A proposta deste trabalho é desenvolver o sistema SYM (*Structure Your Mind*), um simulador para demonstrar o funcionamento dos algoritmos de listas encadeadas, pilhas e filas, que são tipos de estruturas de dados ensinadas na disciplina de estrutura de dados, conforme definido nos projetos pedagógicos dos cursos da área de computação. Estes conteúdos apresentam um maior grau de complexidade em relação ao conteúdo de muitas outras disciplinas dos cursos desta área, e podem parecer bastante abstratos e pouco claros quando transmitidos com o uso de uma técnica tradicional de ensino. O simulador visa permitir a utilização tanto pelo aluno quanto pelo professor, para que haja uma maior interação do aluno com o conteúdo e uma maior facilidade para o professor conduzir as aulas, uma vez que atuará como uma espécie de orientador.

Palavras-chave: Metodologias ativas, Simulador, Estrutura de Dados, Listas encadeadas

ABSTRACT

Traditionally, teaching in schools and universities is most often done in a way that the student only receives the information passively, with no possibility of interacting with the content. Knowledge is usually transmitted theoretically and fixation exercises are passed in some cases, which may not be enough for certain extremely practical content, which requires a clearer visualization and a way of interacting with the content. Active methodologies, such as simulation, can aid in student learning, since they make it possible for the student to interact with the content. The purpose of this work is to develop the SYM (Structure Your Mind) system, a simulator to demonstrate the operation of linked list, stack and queue algorithms, which are types of data structures taught in the data structure discipline, as defined in the projects courses in the computing area. These contents present a greater degree of complexity in relation to the content of many other subjects of the courses of this area, and may seem very abstract and unclear when transmitted with the use of a traditional teaching technique. The simulator aims to allow the use of both the student and the teacher, so that there is a greater interaction of the student with the content and a greater ease for the teacher to lead the classes, since it will act as a kind of advisor.

Keywords: Active methodologies, Simulation, Data Structures, Linked lists

LISTA DE ILUSTRAÇÕES

Figura 1: Interface gráfica do SOSim 16
Figura 2: Tela principal do SIACC 17
Figura 3: Interface do SimDeCs 18
Figura 4: Exemplo de vetor 21
Figura 5: Exemplo de matriz 22
Figura 6: Exemplo de registro 22
Figura 7: Exemplo de lista encadeada 23
Figura 8: Lista duplamente encadeada 23
Figura 9: Estrutura de uma fila 24
Figura 10: Estrutura de uma pilha 25
Figura 11: Exemplos de árvores 25
Figura 12: Exemplo de árvore binária com suas subárvores 26
Figura 13: Tela principal do VisuAlgo 28
Figura 14: Pseudocódigo para inserção de elemento no VisuAlgo 28
Figura 15: Menu de inserção de elemento 29
Figura 16: Operação de inserção em uma lista encadeada no VisuAlgo 30
Figura 17: Página principal do *Data Structure Visualizations* 31
Figura 18: Tela da versão Java do *Data Structure Visualizations* 31
Figura 19: Lista encadeada criada no *Data Structure Visualizations* em Java 32
Figura 20: Pilha criada no *Data Structure Visualizations online* 33
Figura 21: Diagrama de casos de uso 34
Figura 22: Tela do SYM em funcionamento 36
Figura 23: Área de código do SYM com um código em execução 37
Figura 24: Parte da área gráfica mostrando uma lista simplesmente encadeada após uma inserção 38
Figura 25: Área de memória após a alocação do elemento 743 39
Figura 26: Tela do sistema em seu estado inicial 40
Figura 27: Caixa de inserção de elemento 41
Figura 28: SYM durante a execução de uma inserção 41
Figura 29: Caixa de remoção de elemento 42
Figura 30: Sistema durante uma operação de remoção em uma pilha 42
Figura 31: Pesquisa em uma lista simplesmente encadeada desordenada 43

Figura 32: Caixa de geração de lista aleatória 43
Figura 33: Barra inferior do SYM 44
Figura 34: Exemplo de código Javascript puro para criação de um retângulo SVG 46
Figura 35: Exemplo de código Javascript com SVG.js para criação de um retângulo SVG 46
Figura 36: Resultado da primeira pergunta da pesquisa 48
Figura 37: Resultado da segunda pergunta da pesquisa 49
Figura 38: Resultado da terceira pergunta da pesquisa 50
Figura 39: Resultados da quarta pergunta da pesquisa 50

LISTA DE TABELAS

Tabela 1: Tipos primitivos de dados 20
Tabela 2: Rotações da árvore AVL 26
Tabela 3: Sugestões apresentadas na pesquisa e informação sobre sua implementação 51

LISTA DE ABREVIATURAS E SIGLAS

CSS - *Cascade Style Sheets*

EAD - Ensino a Distância

FIFO - *First In, First Out*

HTTP - *HyperText Transference Protocol*

HTML - *HyperText Markup Language*

SEBRAE - Serviço Brasileiro de Apoio às Micro e Pequenas Empresas

SIACC - Sistema Interdisciplinar de Análise de Casos Clínicos

SimDeCS - Simulador Inteligente para a Tomada de Decisão em Cuidados de Saúde

SYM - *Structure Your Mind*

SUMÁRIO

1 INTRODUÇÃO 12
2 TECNOLOGIAS NA EDUCAÇÃO 14
2.1 METODOLOGIAS ATIVAS 14
2.2 JOGOS E SIMULADORES NO AUXÍLIO DA APRENDIZAGEM 15
3 O ENSINO DA DISCIPLINA ESTRUTURA DE DADOS 19
3.1 ALGORITMOS 19
3.2 ESTRUTURAS DE DADOS 20
3.2.1 VETORES E MATRIZES 21
3.2.2 REGISTROS 22
3.2.3 LISTAS ENCADEADAS 22
3.2.4 FILAS 24
3.2.5 PILHAS 24
3.2.6 ÁRVORES 25
3.3 FERRAMENTAS QUE AUXILIAM NO APRENDIZADO DE ESTRUTURA DE DADOS 27
3.3.1 VISUALGO 27
3.3.2 DATA STRUCTURE VISUALIZATIONS 30
4. O SIMULADOR SYM 34
4.1 MOTIVAÇÃO DO PROJETO 34
4.2 DESCRIÇÃO DO FUNCIONAMENTO 35
4.3 TECNOLOGIAS UTILIZADAS 44
4.3.1 HTML5 44
4.3.2 CSS 45
4.3.3 Javascript 45
4.3.4 jQuery 45
4.3.5 SVG.js 46
4.4 PESQUISA DE OPINIÃO 48
5 CONCLUSÃO E TRABALHOS FUTUROS 52
REFERÊNCIAS 54

1 INTRODUÇÃO

Há muitos anos, o ensino é realizado de maneira passiva, onde o aluno permanece a maior parte do tempo observando o que o professor explica no quadro, realiza perguntas, e ao final da explicação, em alguns casos, são passados exercícios para auxiliar na fixação dos conteúdos ensinados. Como complemento, quando possível, normalmente são realizados trabalhos, individuais ou em grupo, para permitir um melhor contato do aluno com a disciplina.

Com o decorrer do tempo, foram surgindo ferramentas e metodologias para auxiliar na transmissão dos conteúdos, já que o quadro negro não era suficiente para demonstrar certos detalhes visuais. Uma das tecnologias mais antigas e muito utilizada até poucos anos atrás é o retroprojetor, equipamento que permite a projeção e ampliação de transparências, onde está o conteúdo que o professor deseja exibir, em paredes ou telas apropriadas. Nelas, podem ser colocados textos ou imagens, aumentando as possibilidades no ensino, uma vez que é possível uma melhor visualização de certos detalhes do conteúdo que não ficam tão claros se apresentados somente de maneira textual. Posteriormente, houve o crescimento do uso do *data show*, ferramenta que pode ser ligada a um computador ou notebook para a exibição de imagens ampliadas, de maneira semelhante ao retroprojetor, porém utilizando-se de tecnologias mais modernas e sem a necessidade do uso de transparências. O *data show*, no entanto, não costuma ter todo o seu potencial aproveitado, pois, na maioria das vezes, é utilizado somente como um substituto evoluído do quadro negro, exibindo slides ou animações improvisadas, quando poderia servir para o uso de metodologias que possibilitem uma visualização mais clara e prática dos conteúdos.

Portanto, o simples uso do *data show* ou tecnologias semelhantes pode não ser suficiente para a assimilação de certos conteúdos que são extremamente práticos. Atualmente, com as tecnologias que estão disponíveis, é possível a utilização de técnicas mais interativas, chamadas metodologias ativas, para a facilitação do aprendizado. Uma dessas metodologias é a simulação, que permite que o ensino seja feito através de um sistema onde o próprio aluno possa interagir e conseguir compreender de maneira prática o que está sendo ensinado, com menor necessidade de intervenção do professor. Neste trabalho será desenvolvido um sistema simulador, denominado SYM (*Structure Your Mind*), para auxiliar no aprendizado do funcionamento dos algoritmos das listas encadeadas e seus tipos especiais, como as pilhas e filas, as quais são conceitos ensinados na disciplina de estrutura de dados, dos cursos de graduação em computação, como Ciência da Computação, Engenharia da Computação e Sistemas de Informação. O ensino dessa disciplina costuma ser um desafio para a didática, tendo em vista

que muitos alunos apresentam dificuldade de compreensão, devido à maior complexidade da programação dessas estruturas, especialmente quando o ensino ocorre do modo tradicional, pela falta de uma maneira de visualização do que está ocorrendo em cada linha do algoritmo. O uso do simulador permitirá a interação e demonstrará passo a passo de forma visual o funcionamento das principais operações que podem ser realizadas nas estruturas, como as inserções e remoções.

Este trabalho será dividido em cinco capítulos. No capítulo 2, será apresentado com maiores detalhes o uso de tecnologias e de metodologias ativas na educação. No capítulo 3, será explicado como funciona o ensino da disciplina e quais as principais dificuldades apresentadas durante seu aprendizado, além de serem apresentados conceitos sobre algoritmos, estruturas de dados. Ainda neste capítulo será explicado o funcionamento das listas encadeadas, pilhas, e filas, que serão objeto deste trabalho, e das árvores binárias. No capítulo 4 será apresentado com detalhes todo o funcionamento do sistema, além das tecnologias utilizadas em seu desenvolvimento, como linguagens de marcação, programação e bibliotecas. Por fim, no capítulo 5, serão apresentadas as conclusões obtidas neste trabalho e as propostas de trabalhos futuros.

2 TECNOLOGIAS NA EDUCAÇÃO

Atualmente, a tecnologia tem grande presença em diversas áreas da sociedade. Dificilmente se realiza alguma tarefa sem a presença de algum tipo de tecnologia para facilitar ou complementar. Do mesmo modo, as tecnologias vêm sendo fortemente incorporadas na educação, área em que, por muitos anos, predominou o modo de ensino tradicional, onde o aluno somente recebe as informações através do professor, sem ter a possibilidade de interagir de algum modo com elas. Para possibilitar uma maior interação durante o aprendizado, podem ser utilizadas as metodologias ativas de aprendizagem, métodos que possibilitam que o aluno interaja com o conteúdo e se torne o agente principal no aprendizado, tendo o professor como orientador.

2.1 METODOLOGIAS ATIVAS

Bastos (2006) define metodologias ativas como “processos interativos de conhecimento, análise, estudos, pesquisas e decisões individuais ou coletivas, com a finalidade de encontrar soluções para um problema”. Ainda segundo o autor, o aluno busca o conhecimento por conta própria, e o professor deixa de fazer o papel de único transmissor do conteúdo e passa a ser orientador, facilitando o aprendizado para que o próprio aluno consiga decidir como atingir o objetivo proposto. Algumas das metodologias ativas mais comuns são a simulação, o seminário e o estudo de caso.

A simulação é uma metodologia ativa que permite que uma pessoa experimente um evento real com o propósito de aprender, avaliar ou entender determinadas situações. Como ferramenta de ensino, é fundamentada na metodologia da Aprendizagem Baseada em Problemas, e é definida como uma metodologia que reproduz situações reais permitindo ao aluno um papel ativo na aquisição dos conceitos necessários para a compreensão e resolução do problema, enquanto que o professor adota uma postura de condutor ou facilitador (COSTA et. al., 2005)

Segundo Masetto (2012), o nome seminário normalmente é atribuído indevidamente a um trabalho onde os alunos fazem um resumo de um assunto determinado pelo professor e apresentam-no à turma, ficando o professor, na maioria das vezes, como expectador, pouco interferindo na apresentação. O autor defende que, em sua concepção original, o seminário consiste em um trabalho em grupo dividido em duas partes, onde a primeira parte trata-se da distribuição de assuntos para os grupos, para que seja realizada uma pesquisa, orientada pelo professor e a segunda é a realização de uma espécie de mesa-redonda com um representante de

cada grupo, onde é debatido um assunto relacionado aos que foram pesquisados pelos alunos, mas que não foi pesquisado diretamente, podendo ter a participação do restante da turma. Isso possibilita uma maior interação dos alunos com o conteúdo, e faz com que seja necessário ir além do que foi estudado.

Um estudo de caso é definido por Masetto (2012) como uma técnica que tem por objetivo colocar o aluno em contato com situações profissionais, por meio de situações reais, quando é trazido um problema real para que os alunos proponham soluções, ou por situações simuladas, onde o professor cria situações de acordo com o que foi ensinado, com o mesmo objetivo. Normalmente é realizado em grupos, o que possibilita o trabalho em equipe e a troca de informações entre os alunos, sendo, assim, uma ótima forma de aplicar os conteúdos vistos em aula para a solução do problema.

Neste trabalho será utilizada a simulação como metodologia, visando criar uma maneira de transmitir o conteúdo de forma visual e interativa, para possibilitar ao aluno uma melhor compreensão do conteúdo.

2.2 JOGOS E SIMULADORES NO AUXÍLIO DA APRENDIZAGEM

Apesar de ferramentas de auxílio ao aprendizado, como os jogos educacionais, estarem a cada dia mais presentes no ensino, elas têm maior predominância nos ensinos fundamental e médio. Assim, quando o estudante chega ao ensino superior, pode encontrar dificuldade ao se deparar com uma enorme quantidade de disciplinas que, dependendo do curso, são extremamente teóricas, com pouco ou nenhum uso de formas práticas de transmissão do conteúdo. É necessária uma maior presença de ferramentas que auxiliem no aprendizado no ensino superior, como os simuladores e os jogos educacionais.

No ensino fundamental, a disciplina de matemática possui diversos jogos, tendo sido apresentados dois deles por Grando (2000) em sua tese de doutorado. A autora fez um experimento com alunos de 6ª série utilizando jogos de estratégia como o Contig 60, que, segundo a autora, trabalha o cálculo mental das quatro operações básicas, expressões numéricas e propriedades aritméticas e o Nim, que trabalha os conceitos de divisibilidade e múltiplos dos números, além de também trabalhar o cálculo mental. Como resultado, foi obtido um grande envolvimento dos alunos com as atividades, que demonstraram alegria ao vencer cada desafio e trabalharam, em muitos casos, assuntos matemáticos que iam além dos abordados pelo jogo, ou resgataram conteúdos que já haviam sido trabalhados anteriormente, ao mesmo tempo em que estavam envolvidos no jogo. (GRANDO, 2000). A experiência demonstrou que com o uso

de metodologias ativas, como é o caso dos jogos, os estudantes puderam obter um resultado muito melhor do que o que poderia ser obtido através de aulas expositivas comuns.

Já no campo dos simuladores existem aplicações para o aprendizado em diversas áreas, sendo a área da medicina a detentora da maior parte delas, além da área do empreendedorismo. Na área da computação, apesar de existirem simuladores, a grande maioria somente é desenvolvida como monografia ou tese de conclusão de curso, não havendo aplicação prática. Além disso, por meio de pesquisas, pode-se notar que há escassez de simuladores na área de estrutura de dados, sendo mais predominantes na área de sistemas operacionais.

Dentre os simuladores encontrados na área da computação está o software SOSim, desenvolvido como tese de mestrado e descrito pelo autor com o objetivo de ser "um simulador gráfico (SOsim) que sirva de ferramenta visual de suporte efetivo para o ensino e aprendizado dos conceitos e técnicas implementados nos sistemas operacionais" (MAIA, 2001). O sistema demonstra visualmente diversos conceitos, como criação, visualização e escalonamento de processos, gerenciamento de memória, memória virtual, além de muitos outros conceitos relacionados à disciplina. A figura 1 demonstra a interface do SOsim.

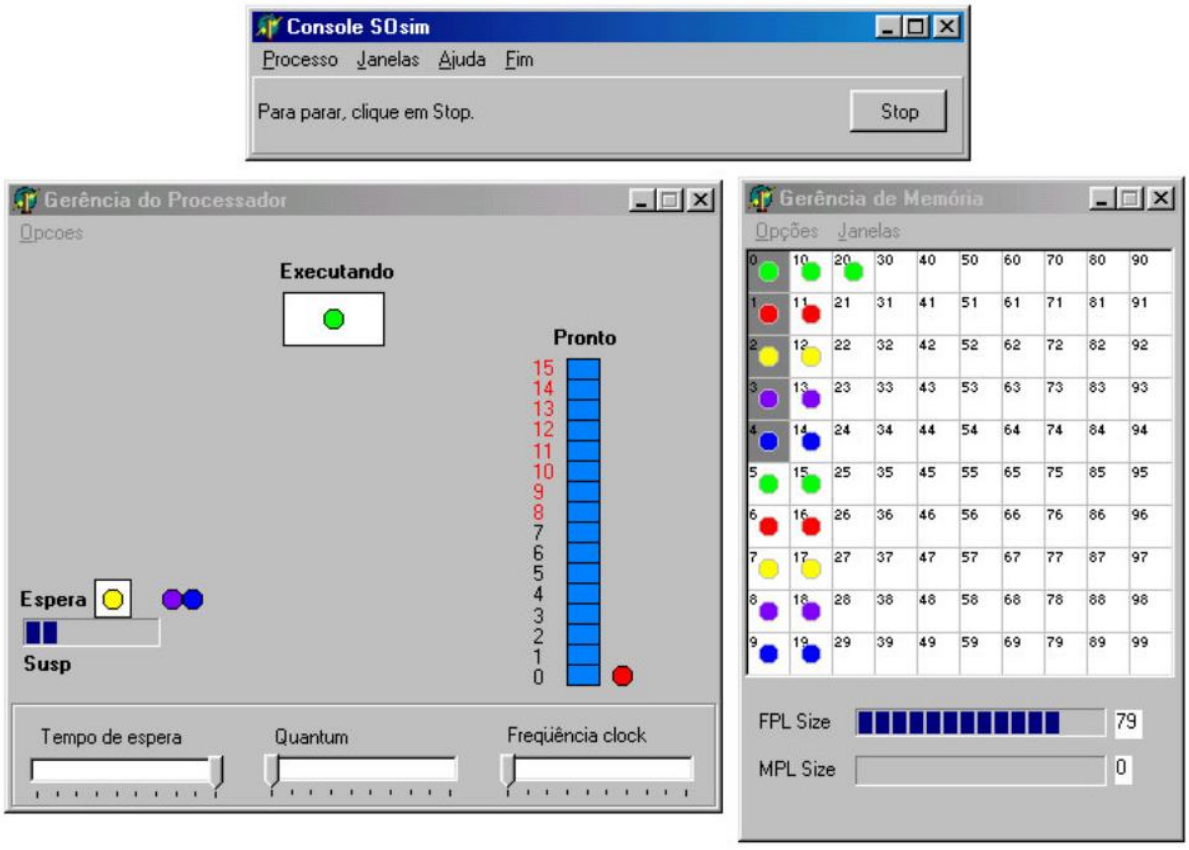


Figura 1: Interface gráfica do SOSim
Fonte: MAIA (2001)

Na área do empreendedorismo existe um projeto bastante conhecido já em funcionamento, que é o Desafio Universitário, desenvolvido pelo Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE), que visa incentivar o desenvolvimento das habilidades de empreendedorismo aos universitários, através de jogos, difundindo conceitos relacionados à área.

Na área da medicina, duas ferramentas são frequentemente utilizadas: o Sistema Interdisciplinar de Análise de Casos Clínicos (SIACC) e o Simulador Inteligente para a Tomada de Decisão em Cuidados de Saúde (SimDeCS).

O SIACC, em uso em instituições de ensino na área desde 2009, simula um paciente virtual, permitindo o estudo das principais patologias vivenciadas na rotina médica. (LIMA et. al., 2015). Os casos a serem trabalhados no SIACC são modelados de forma que constem a história, os exames, dados de laboratório e as imagens. No decorrer da simulação, são exibidos os conteúdos relacionados e são feitas perguntas com links relacionados aos assuntos trabalhados, para que o aluno possa desenvolver o raciocínio diagnóstico. A figura 2 mostra a tela principal do sistema. (FLORES et. al., 2011)

Figura 2: Tela principal do SIACC
Fonte: MILLÃO et. al. (2017)

O SimDeCS também permite a simulação de um paciente virtual, porém se apresenta na forma de um jogo sério, através de fases, servindo de apoio ao aprendizado, permitindo

desenvolver diversas habilidades, dentre elas as habilidades técnicas e o raciocínio diagnóstico. Nele o aluno pode acompanhar o estado de saúde do paciente, obter exames e, a partir dos dados, apresentar o diagnóstico. A interface do SimDeCs é apresentada na figura 3. (LIMA et. al., 2015; FLORES et. al., 2011)

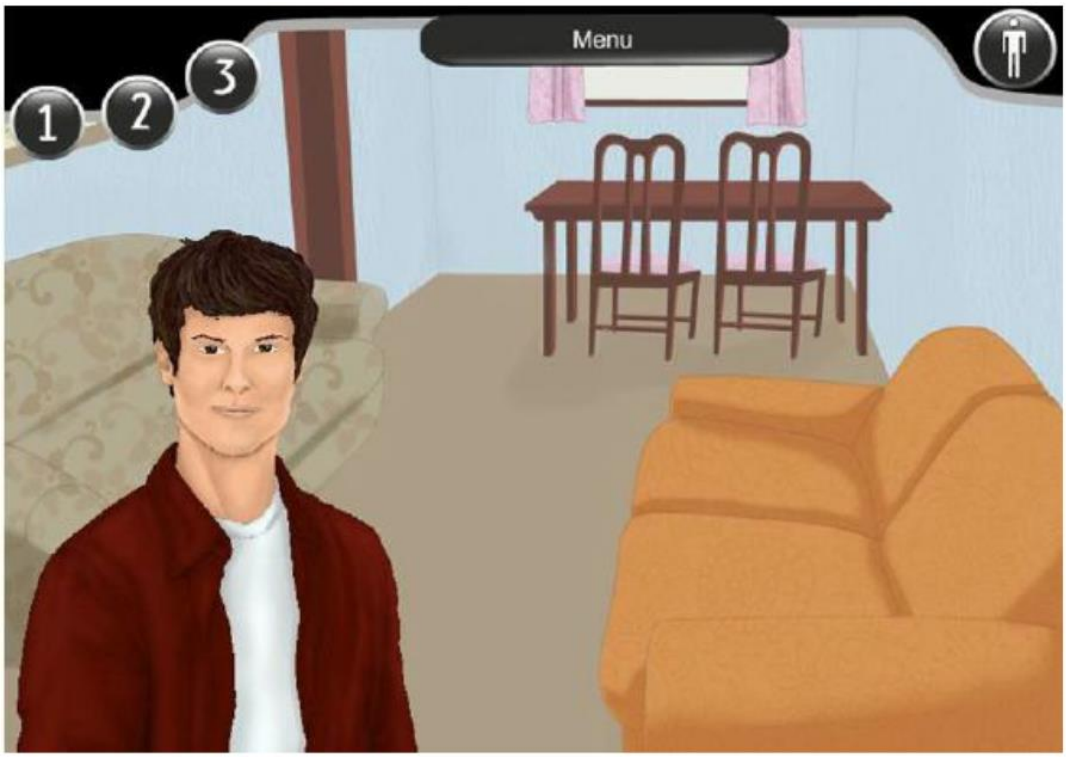


Figura 3: Interface do SimDeCs
Fonte: BEZ et. al. (2012)

Considerando a dificuldade de aprendizado de conteúdos mais práticos, tendo em vista que muitos detalhes não ficam claros em uma aula expositiva da disciplina estrutura de dados, ministrada nos cursos de graduação e pós-graduação na área de computação, será desenvolvido neste trabalho um simulador para auxílio no ensino e aprendizado dos principais algoritmos da disciplina de estrutura de dados.

3 O ENSINO DA DISCIPLINA ESTRUTURA DE DADOS

A disciplina estrutura de dados possui uma grande diferença no grau de complexidade em relação a muitas outras disciplinas dos cursos de computação. Compreender programação, por si só, é algo em que muitos alunos apresentam dificuldades. No caso dessa disciplina, é necessário um aprendizado mais profundo de programação, uma vez que utiliza estruturas de programação que normalmente não são ensinadas nas disciplinas iniciais dos cursos.

Além do que já foi mencionado, o ensino da disciplina de estrutura de dados normalmente é realizado na forma tradicional, em aulas expositivas teóricas, sendo a única parte prática a criação e execução de algoritmos, o que não torna clara a visualização do que está ocorrendo, uma vez que a execução dos algoritmos se resume à manipulação de variáveis internas, sem retornos visuais. Com isso, a única forma de o aluno compreender o funcionamento é tentar simular mentalmente o funcionamento, o que pode ser bastante confuso para muitos.

A forma tradicional de ensino pode ser um dos fatores pelos quais os índices de reprovação na disciplina costumam ser bastante altos, pois, em algumas vezes, o aluno já não obteve uma boa base de ensino desde as disciplinas iniciais do curso, ou simplesmente decorou o mínimo necessário para obter aprovação, sem realmente compreender o conteúdo. Portanto, é importante que haja uma forma de permitir a visualização do que ocorre durante a execução dos algoritmos, para que se torne mais fácil o aprendizado, e para que o aluno não tenha que se limitar a imaginar o que ocorre dentro de cada estrutura, ou visualizar imagens estáticas, que muitas vezes são pouco esclarecedoras, levando-se em conta que há vários passos até que se obtenha o resultado que é mostrado. Isso é possível por meio da simulação dos algoritmos, com o uso de recursos visuais e animações.

Este capítulo visa apresentar uma síntese de alguns conceitos sobre algoritmos e estruturas de dados, além de apresentar alguns exemplos de simuladores similares que envolvem a área de estrutura de dados e alguns detalhes de seu funcionamento.

3.1 ALGORITMOS

Algoritmos podem ser definidos como “uma sequência de passos que visam a atingir um objetivo bem definido” (FORBELLONE e EBESPÄCHER, 2005).

Em computação, os algoritmos são objeto de estudo da lógica de programação, que tem o objetivo de criar sequências de passos para solução de problemas visando sua utilização para

a criação de programas através de uma linguagem de programação.

Um algoritmo pode armazenar dados utilizando dois tipos de estruturas, que são denominadas variáveis e constantes. Uma constante é um dado que não tem seu valor alterado durante toda a execução do algoritmo. Já uma variável pode ser definida como um dado que pode ter seu valor modificado no decorrer da execução do algoritmo. Em um programa, ambas as estruturas são armazenadas em espaços na memória do computador, recebendo um nome que permita sua localização. (ASCENCIO e CAMPOS, 2002; FORBELLONE e EBESPÄCHER, 2005)

As variáveis e constantes são criadas por meio da declaração, onde recebem um nome e podem ser associadas a tipos de dados, que especificam o tipo de conteúdo que pode ser armazenado nas mesmas e as operações que podem ser realizadas sobre elas. Os principais tipos de dados, que fazem parte da própria linguagem de programação, são denominados tipos primitivos.

Os tipos primitivos de dados possuem tamanhos predefinidos e são destinados ao armazenamento de números inteiros ou decimais, textos ou valores lógicos (verdadeiro ou falso). A tabela 1 apresenta os principais tipos primitivos de dados.

Tabela 1: Tipos primitivos de dados
Fonte: ASCENCIO e CAMPOS (2002)

Tipo	**Descrição**
Inteiro	Representa os números positivos ou negativos sem parte decimal.
Real	Representa os números positivos ou negativos com parte decimal.
Lógico	Representa os valores *verdadeiro* ou *falso*.
Caracter	Representa um único caracter ou um conjunto de caracteres, como letras, números sem finalidade de cálculo e caracteres especiais.

A partir dos tipos primitivos de dados, é possível a criação de outros tipos de dados pelo próprio usuário, a partir de conjuntos de dados com um ou mais tipos primitivos. Esses tipos de dados são denominados tipos abstratos de dados.

3.2 ESTRUTURAS DE DADOS

Em certos casos, o conceito de tipo primitivo de dado pode não ser suficiente para

determinar o tipo da informação a ser armazenada. Isso ocorre porque, em certos casos, é necessário trabalhar com várias informações em conjunto, por possuírem alguma ligação em comum, como, por exemplo, quando se trata de informações que descrevem um mesmo elemento. Nesse tipo de situação é necessário o uso de um tipo de dados que possibilite o armazenamento de um conjunto de dados, do mesmo tipo ou não, chamado tipo abstrato de dados (LAUREANO, 2008; PEREIRA, 1996). Uma estrutura de dados é a representação de um tipo abstrato de dados (ASCENCIO e ARAÚJO, 2010).

Estruturas de dados que utilizam somente um tipo de dado são chamadas de variáveis compostas homogêneas. Correspondem a várias posições de memória com o mesmo tipo de dado identificadas por um índice individual. Como exemplo, tem-se os vetores e matrizes (LAUREANO, 2008).

3.2.1 VETORES E MATRIZES

O vetor é uma variável composta homogênea que possui uma só dimensão, o que faz com que necessite de um único índice para endereçar seus elementos. Armazena um conjunto de valores de um mesmo tipo na mesma variável, e identifica-os através de seu índice individual (LAUREANO, 2008). A figura 4 demonstra um exemplo de vetor.

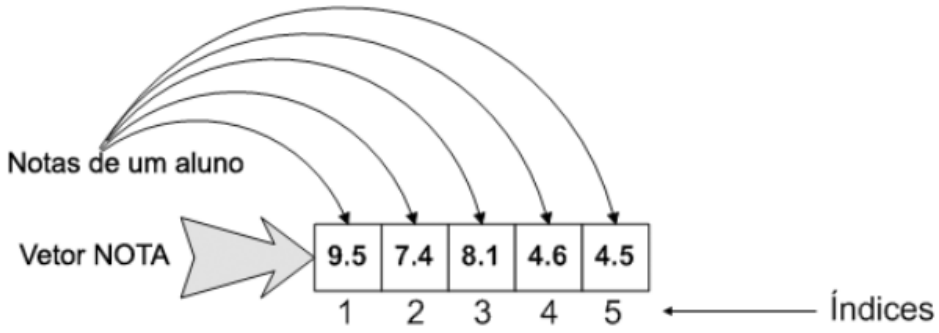


Figura 4: Exemplo de vetor
Fonte: LAUREANO (2008)

A matriz possui funcionamento semelhante ao do vetor, porém possui duas dimensões, o que significa que a mesma divide seus elementos em linhas e colunas, e, portanto, faz-se necessário o uso de dois índices para localizar seus elementos. A figura 5 demonstra um exemplo de matriz (LAUREANO, 2008).

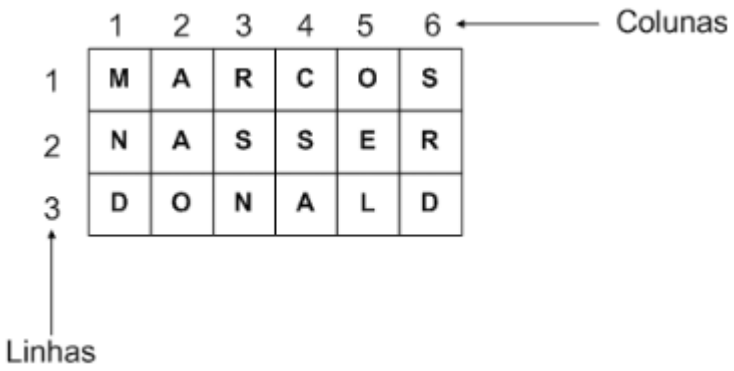


Figura 5: Exemplo de matriz
Fonte: LAUREANO (2008)

3.2.2 REGISTROS

Já as estruturas de dados que utilizam mais de um tipo de dado são chamadas de variáveis compostas heterogêneas, e também são conhecidas como registros. Visam agrupar dados que não são do mesmo tipo, mas guardam algum tipo de relação. No registro, os conjuntos de valores armazenados são identificados por um nome único, que identifica o conjunto de posições de memória, e por nomes individuais atribuídos a cada valor, no lugar de índices dentro de uma mesma variável, como ocorre nas variáveis compostas homogêneas (LAUREANO, 2008). Um exemplo de registro é demonstrado na figura 6.

conta	num	conta.num
	saldo	conta.saldo
	nome	conta.nome

Figura 6: Exemplo de registro
Fonte: (ASCENCIO e CAMPOS, 2002)

3.2.3 LISTAS ENCADEADAS

Os vetores, matrizes e registros armazenam os dados de forma sequencial na memória, o que permite que seus elementos sejam facilmente localizados através de seus índices ou nomes de identificação. Estas estruturas possuem, na maioria das vezes, tamanho fixo ao serem declaradas, com exceção dos vetores criados com o uso de alocação dinâmica de memória, que têm o tamanho definido durante a execução do programa. Em certos casos, pode ser necessário criar listas que permitam o acréscimo ou exclusão constante de elementos após a sua criação, o que é extremamente difícil com o uso de vetores dinamicamente alocados, já que para a

realização dessas operações, pode ser necessária a movimentação manual de vários elementos. Para solucionar esse problema, existe a estrutura de dados denominada lista encadeada, que é composta por elementos interligados alocados em locais distintos da memória, onde cada elemento possui seu conteúdo e uma referência para o elemento seguinte. De acordo com o tipo da lista, pode ser possível adicionar ou remover itens em qualquer posição da lista, o que faz com que a lista seja muito mais flexível que um vetor ou matriz. No entanto não é possível acessar todos os elementos diretamente, pois não há um índice individual, como ocorre nos vetores, sendo necessário percorrer os elementos anteriores para que se possa localizar um específico. (FORBELLONE e EBERSPÄCHER, 2005). A figura 7 mostra um exemplo de lista, onde é possível visualizar que é possível remover um dos elementos sem que se prejudique o restante da lista, bastando modificar a referência no elemento anterior

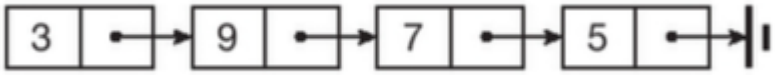


Figura 7: Exemplo de lista encadeada
Fonte: ASCENCIO e ARAÚJO (2010)

Existem diversos tipos específicos de lista, com diferentes características. Alguns dos principais tipos de lista são a lista simplesmente encadeada, a lista duplamente encadeada e a lista ordenada.

Uma lista simplesmente encadeada possui nós compostos de um valor e de um único ponteiro, que aponta para o elemento seguinte, e permite inserção e remoção em qualquer parte. A lista simplesmente encadeada é a mostrada na figura 7.

A lista duplamente encadeada armazena nós compostos de um valor e de referências tanto para o nó anterior quanto para o posterior, e também permite inserção e remoção em todas as posições. Possibilita uma maior facilidade na navegação entre os nós, dado que não é necessário armazenar a referência para o elemento anterior em variáveis temporárias, como ocorre com as listas simplesmente encadeadas durante operações de busca. A figura 8 mostra uma lista duplamente encadeada (ASCENSIO e ARAÚJO, 2010).

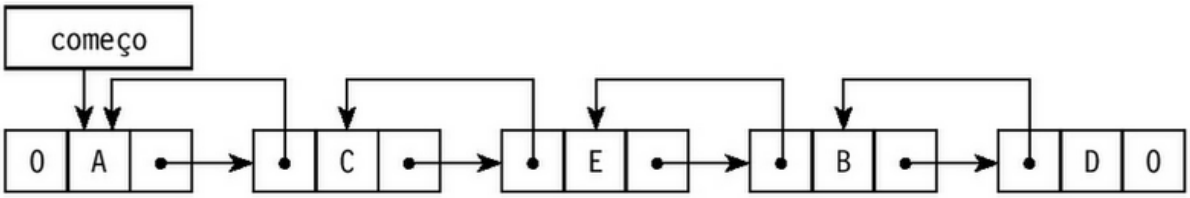


Figura 8: Lista duplamente encadeada
Fonte: ASCENCIO e ARAÚJO (2010)

Lista ordenada é um tipo de lista onde os elementos devem ser inseridos obedecendo a ordem crescente de valor, o que significa que a posição não é escolhida pelo usuário, e sim determinada de acordo com o valor inserido. Uma lista ordenada pode ser simplesmente ou duplamente encadeada.

3.2.4 FILAS

Filas são estruturas de dados que se comportam de maneira semelhante a uma fila real, com regras específicas de entrada e saída de elementos. Uma fila possui basicamente as mesmas operações que uma lista, com algumas limitações nas operações de inserção e remoção. Em uma fila, as inserções sempre são feitas no final, e as remoções são feitas sempre no início, o que faz com que o primeiro elemento inserido seja sempre o primeiro a ser removido. Essa política de entrada e saída recebe o nome de FIFO – *First In, First Out* (primeiro a entrar, primeiro a sair). A figura 9 mostra a estrutura de uma fila. (FORBELLONE e EBERSPÄCHER, 2005; ASCENSIO e ARAÚJO, 2010).

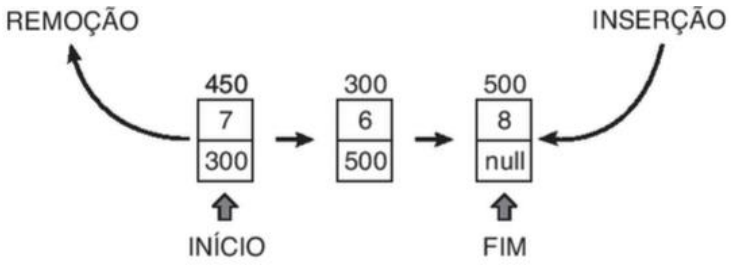


Figura 9: Estrutura de uma fila
Fonte: ASCENCIO e ARAÚJO (2010)

3.2.5 PILHAS

Uma pilha, assim como a fila, é outra estrutura que funciona de maneira semelhante à da lista, possuindo também a grande maioria das operações da mesma, porém possui uma forma distinta de inserção de elementos. Em uma pilha, os elementos somente podem ser inseridos ou removidos através do início, denominado topo da pilha. A política de entrada e saída de elementos da pilha é denominada LIFO – *Last In, First Out* (último a entrar, primeiro a sair). A estrutura de uma pilha é mostrada na figura 10. (FORBELLONE e EBERSPÄCHER, 2005; ASCENSIO e ARAÚJO, 2010)

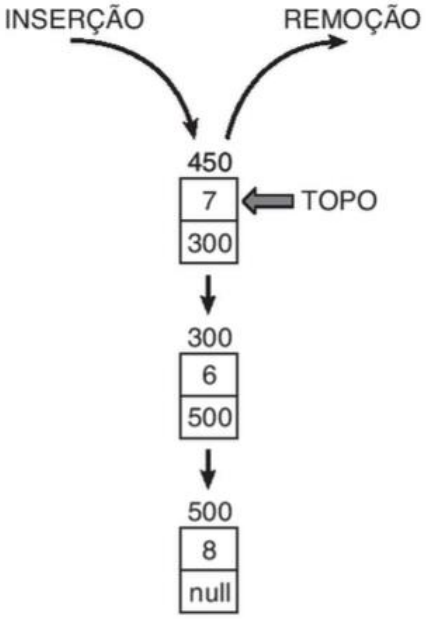


Figura 10: Estrutura de uma pilha
Fonte: ASCENCIO e ARAÚJO (2010)

3.2.6 ÁRVORES

Segundo Forbellone e Eberspächer (2005), a árvore "é uma lista na qual cada elemento possui dois ou mais sucessores, porém todos os elementos possuem apenas um antecessor". Os elementos da árvore, diferentemente da lista, possuem uma relação de hierarquia. Cada elemento da árvore é chamado de nó. O primeiro nó é chamado de raiz da árvore. Os sucessores de um nó são chamados filhos, os antecessores são chamados pais, e o elemento que não possui filhos é chamado folha (FORBELLONE e EBERSPÄCHER, 2005). A figura 11 apresenta representações de árvores.

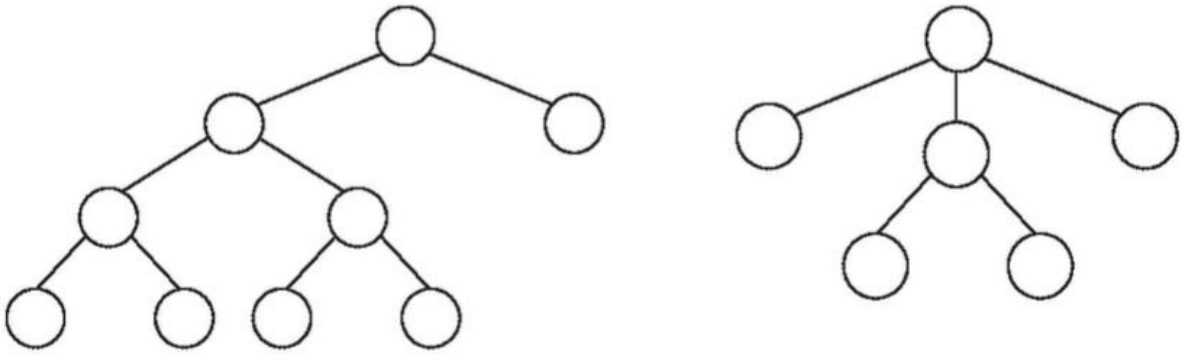


Figura 11: Exemplos de árvores
Fonte: ASCENCIO e ARAÚJO (2010)

Uma árvore binária é uma árvore que pode ser divida em três subconjuntos: 1º subconjunto (raiz), 2º subconjunto (subárvore à direita) e 3º subconjunto (subárvore à esquerda) (ASCENSIO e ARAÚJO, 2010), A figura 12 mostra uma árvore binária e suas subárvores.

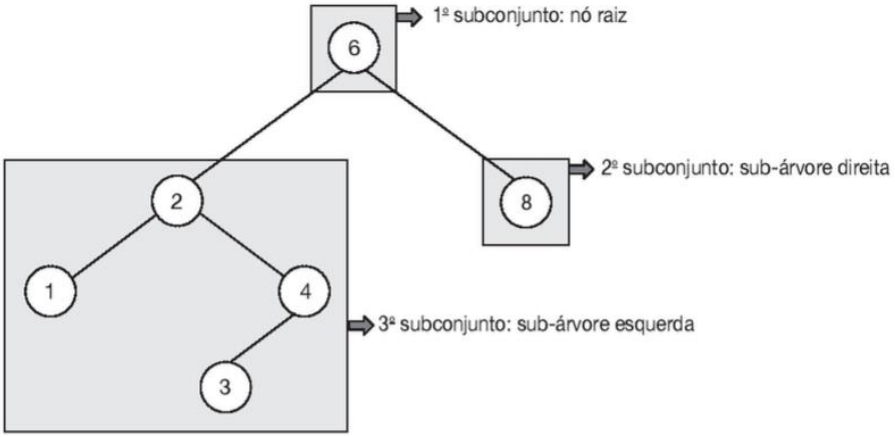


Figura 12: Exemplo de árvore binária com suas subárvores
Fonte: ASCENCIO e ARAÚJO (2010)

Em uma árvore binária de busca, (ASCENSIO e ARAÚJO, 2010).

As inserções e remoções em uma árvore binária são feitas de acordo com sua relação com o elemento antecessor. Após algumas inserções e remoções, é comum que um dos lados da árvore ou uma subárvore de um nó qualquer fique desproporcionalmente maior que o outro lado. Alguns tipos de árvore pode ter o tempo de busca prejudicado por esse desequilíbrio, como é o caso das árvores binárias de busca, que são utilizadas para agilizar a busca de elementos, e são organizadas de forma que todos os nós da subárvore direita são maiores que o nó raiz, enquanto que todos os da subárvore esquerda são menores que o nó raiz. Para evitar esse problema, as árvores binárias de busca utilizam o chamado balanceamento, que mantém o equilíbrio da árvore a cada inserção ou remoção de elementos. A árvore AVL é um exemplo de árvore que utiliza o conceito de balanceamento.

A árvore AVL é uma árvore binária de busca balanceada criada em 1962 por Adelson-Velsky e Landis, que deram origem ao nome. Nas árvores AVL, a diferença de altura entre as subárvores direita e esquerda só pode ser de 1, 0 ou -1. Quando ocorre um valor diferente, é necessário realizar o balanceamento (ASCENCIO e ARAÚJO, 2010). O balanceamento nas árvores AVL é realizado através das rotações, conforme a tabela 2 a seguir.

Tabela 2: Rotações da árvore AVL
Fonte: ASCENCIO e ARAÚJO (2010)

Diferença da altura de um nó	Diferença de altura do nó filho do nó desbalanceado	Tipo de rotação
2	1	Simples à esquerda

	0	Simples à esquerda
	-1	Dupla com filho para a direita e pai para a esquerda
-2	1	Dupla com filho para a esquerda e pai para a direita
	0	Simples à direita
	-1	Simples à direita

3.3 FERRAMENTAS QUE AUXILIAM NO APRENDIZADO DE ESTRUTURA DE DADOS

Durante as pesquisas para este trabalho foram localizados dois exemplos de ferramentas que também possuem a finalidade de demonstrar visualmente, por meio da simulação, o funcionamento de algumas estruturas de dados. A primeira ferramenta encontrada foi o VisuAlgo, que possui um grande número de estruturas e funcionalidades, e a segunda foi o visualizador de estruturas de dados da Universidade de San Francisco, que possui um número bastante limitado de funcionalidades, tendo como objetivo somente exibir as animações da criação e manipulação das estruturas.

3.3.1 VISUALGO

O VisuAlgo[1] é uma ferramenta desenvolvida no ano de 2011 pelo professor Steven Halim com o objetivo de auxiliar seus alunos na compreensão de algoritmos e estruturas de dados. Dispõe de simulações de diversas estruturas de dados, como listas, árvores e tabelas *hash*, e todas as estruturas disponíveis podem ser acessadas a partir da página principal do site, que é demonstrada na figura 13.

[1] https://visualgo.net

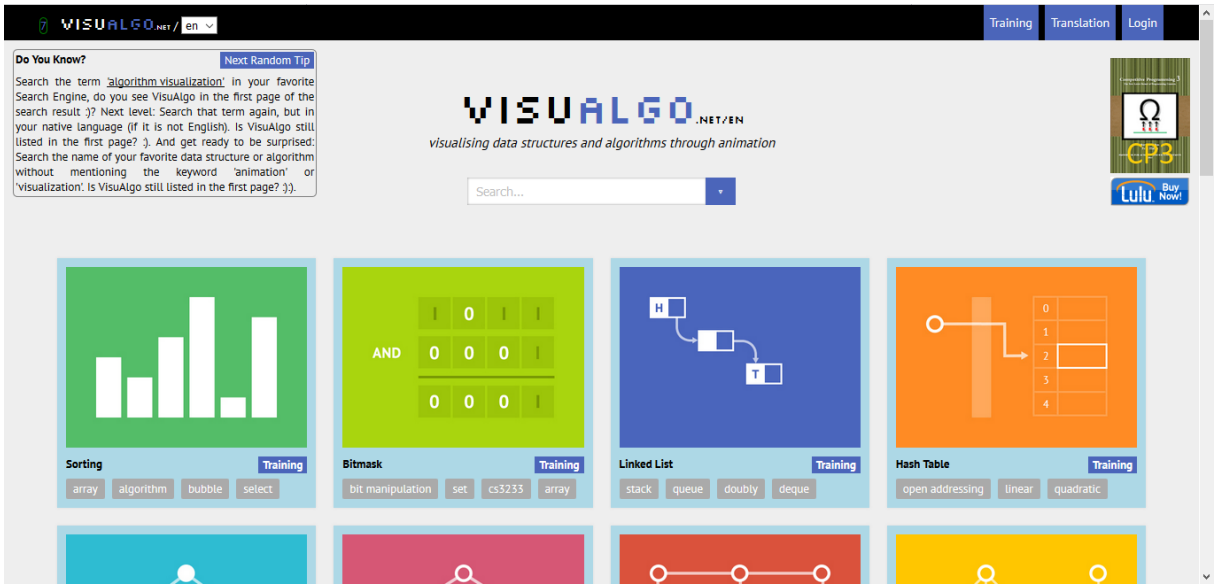


Figura 13: Tela principal do VisuAlgo
Fonte: Autor

Esta ferramenta disponibiliza, para as listas encadeadas, as operações de criação da lista e de inserção, remoção e pesquisa de elementos. Possui um grande número de recursos, simulando cada uma das operações, utilizando animações para mostrar cada passo, e indicação de alguns dos ponteiros utilizados. O foco principal da ferramenta é na demonstração visual do comportamento das estruturas, possuindo um menor foco nos algoritmos, disponibilizando pequenos pseudocódigos para as demonstrações. A figura 14 demonstra um exemplo de pseudocódigo de inserção de elemento em uma lista simplesmente encadeada.

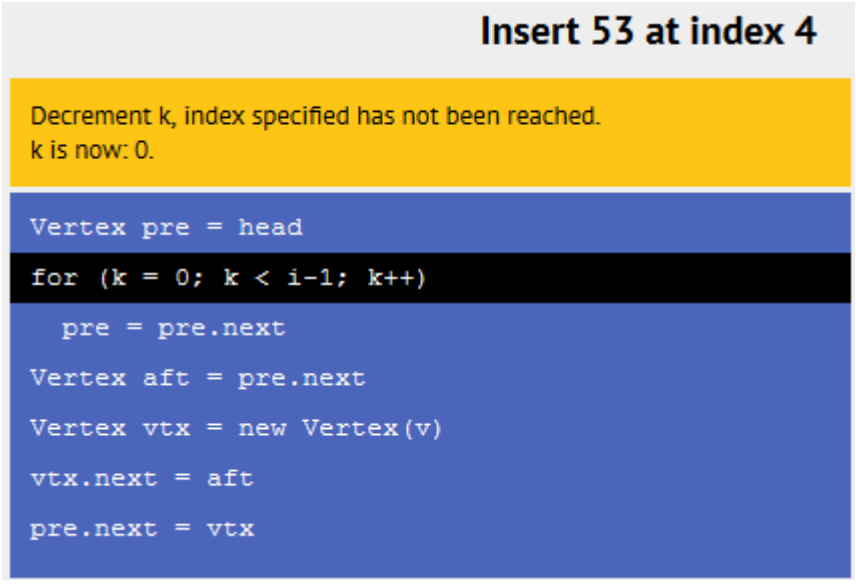


Figura 14: Pseudocódigo para inserção de elemento no VisuAlgo
Fonte: Autor

Além disso, o VisuAlgo, não dispõe de nenhum recurso que demonstre visualmente

como cada estrutura é armazenada na memória, representando cada elemento no pseudocódigo como instância de um objeto de uma classe denominada “Vertex”, que não tem sua estrutura descrita, não sendo possível determinar de maneira clara quais são seus atributos. Também não disponibiliza informações sobre a estrutura da lista, utilizando os ponteiros *head* e *tail*, que indicam início e fim da lista, sem referência de como os mesmos foram criados.

Outro ponto importante a ser observado, é que, apesar de os ponteiros auxiliares serem demonstrados, representados por *pre* e *aft* (anterior e seguinte, respectivamente), sua visualização não é clara durante a simulação, limitando-se apenas a textos inseridos abaixo de cada elemento sem uso de animação, o que, mesmo em velocidade baixa, dificulta a visualização da transição entre cada passo do pseudocódigo pelo usuário.

Para realizar uma inserção em uma lista no VisuAlgo deve-se clicar no menu na lateral esquerda, selecionar a opção *Insert*, e informar se deseja incluir no início, no final ou em uma posição específica no meio da lista, informar o valor do elemento, e, se for o caso, a posição, e clicar em *Go*, como mostra a figura 15.

Figura 15: Menu de inserção de elemento
Fonte: Autor

Após clicar no botão *Go*, é iniciada a operação de inserção, mostrada na figura 16, com a busca pela posição informada e posterior criação e inserção do elemento, exibindo os passos percorridos por meio do uso de animações, e associando os passos ao pseudocódigo exibido na lateral direita.

As demais operações, como busca e remoção de elementos, funcionam do mesmo modo que a inserção, sendo possível controlar a velocidade ou controlar manualmente o avanço dos passos de cada operação. No VisuAlgo não é possível utilizar continuamente a mesma lista caso todos os elementos sejam removidos, já que ao remover o último elemento o sistema não disponibiliza mais os ponteiros de início e fim da lista e não permite que sejam incluídos novos elementos. Isso faz com que seja necessário criar uma nova lista para voltar a inserir elementos.

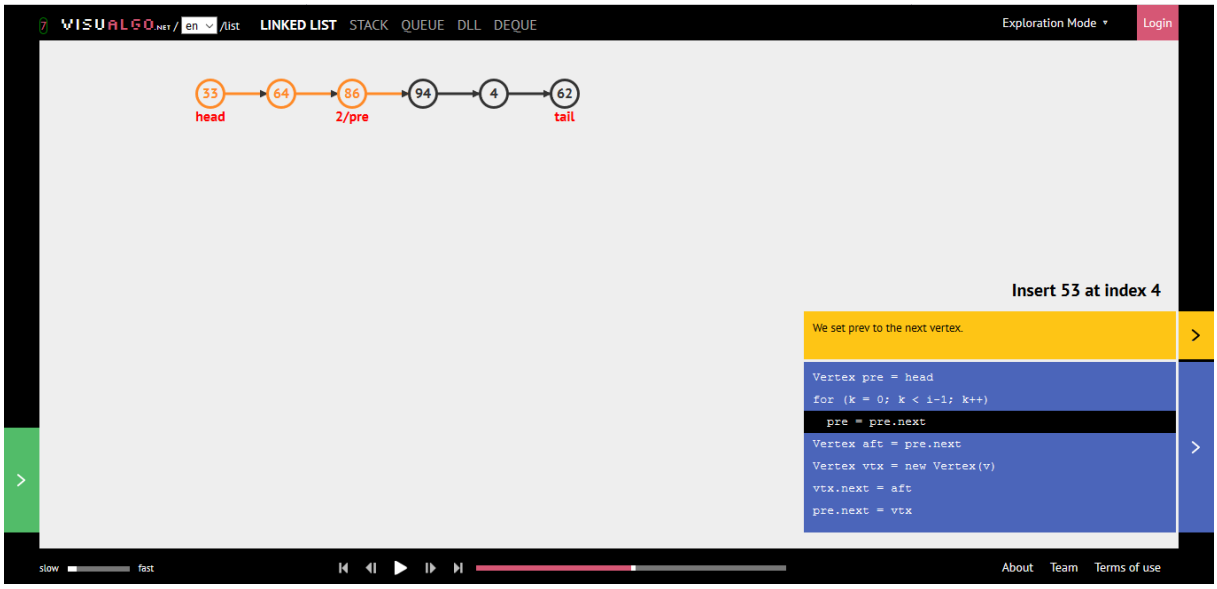


Figura 16: Operação de inserção em uma lista encadeada no VisuAlgo
Fonte: Autor

O VisuAlgo disponibiliza para simulação, no caso das listas, a lista simplesmente encadeada, lista duplamente encadeada, pilha, fila e deque. No entanto, não disponibiliza a lista simplesmente encadeada ordenada, permitindo somente a criação de uma lista simplesmente encadeada aleatória gerada já em ordem crescente e de uma única vez, sem demonstração dos passos percorridos para sua criação. Além disso, quando um elemento é inserido manualmente após a criação da lista, o funcionamento é o mesmo de uma lista simplesmente encadeada desordenada.

O funcionamento do VisuAlgo é exclusivamente *online*, o que dificulta o uso em sala de aula em casos de ausência de conexão à internet. Disponibiliza também um cadastro para os usuários, que possui como finalidade disponibilizar alguns recursos adicionais, como tradução em outros idiomas.

3.3.2 DATA STRUCTURE VISUALIZATIONS

O sistema *Data Structure Visualizations*[2] foi desenvolvido em 2011 pelo professor David Galles, da Universidade de San Francisco, localizada na Califórnia, nos Estados Unidos. Disponibiliza, também, visualização por meio de animações para diversas estruturas de dados, como listas encadeadas, pilhas, filas e árvores. A página principal do sistema, onde são mostradas todas as estruturas disponíveis, é apresentada na figura 17.

[2] https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

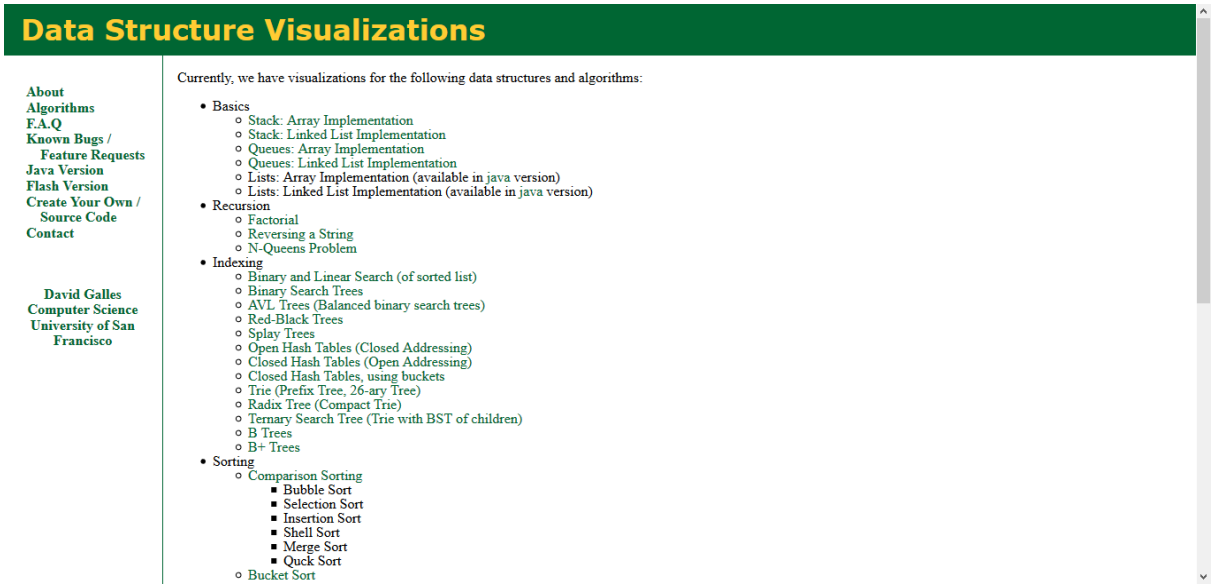


Figura 17: Página principal do *Data Structure Visualizations*
Fonte: Autor

O sistema possui a versão *online*, mais recente e desenvolvida em Javascript, e *offline*, desenvolvida em Java. Algumas estruturas, como é o caso da lista encadeada, somente estão disponíveis na versão em Java, que, além de ser bastante incompleta, já não é mais atualizada, segundo informação disponibilizada no site pelo próprio autor. Além disso, a versão em Java necessita da instalação do Java para sua utilização, ao contrário do SYM, que não necessita de nenhum programa além de um navegador web. A versão Java do *Data Structure Visualizations* é mostrada na figura 18.

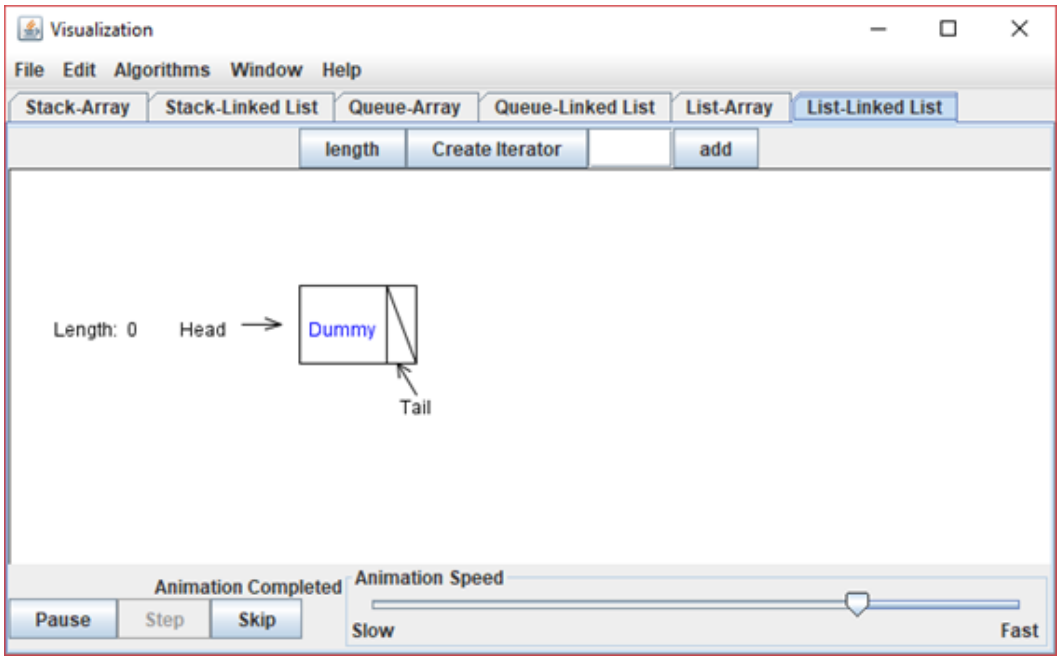


Figura 18: Tela da versão Java do *Data Structure Visualizations*
Fonte: Autor

A versão Java é a única a disponibilizar a lista encadeada, que possui um número muito reduzido de funcionalidades, não possibilitando a busca de elementos, além de possuir um funcionamento pouco claro, sendo difícil determinar, em alguns casos, onde cada elemento vai ser inserido. Além disso, não mostra o caminho percorrido até o local da inserção do elemento, indicando somente qual é o elemento anterior por meio de uma seta, que muitas vezes não condiz com o local onde o elemento foi inserido. A figura 19 apresenta uma lista encadeada criada no *Data Structure Visualizations* em Java.

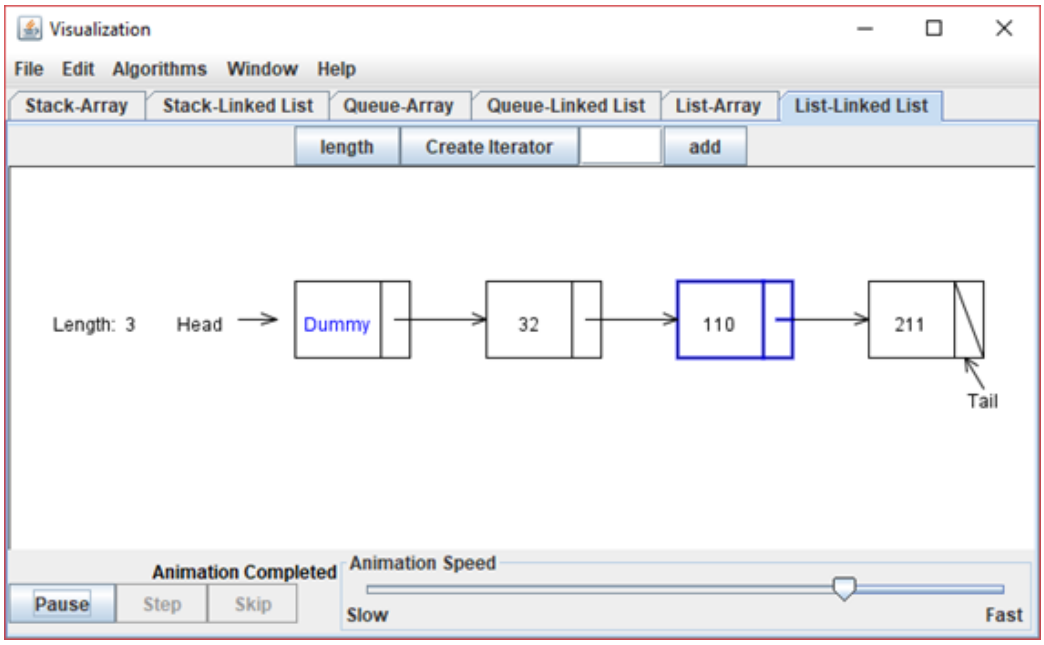


Figura 19: Lista encadeada criada no *Data Structure Visualizations* em Java
Fonte: Autor

A versão online do *Data Structure Visualizations*, além de não possuir a lista encadeada entre as estruturas disponíveis para simulação, também não dispõe da operação de busca nas estruturas de pilha e fila, limitando-se à inserção e remoção, que são realizadas de forma bastante simplificada, unicamente criando o elemento e vinculando-o à estrutura. Em ambas as versões, possibilita o controle manual do avanço dos passos e da velocidade da execução da animação. Não dispõe de nenhuma forma de visualização dos algoritmos de cada estrutura durante a execução da simulação, nem de visualização da alocação dos elementos da memória, em nenhuma das versões. Uma pilha criada no *Data Structure Visualizations online* é mostrada na figura 20.

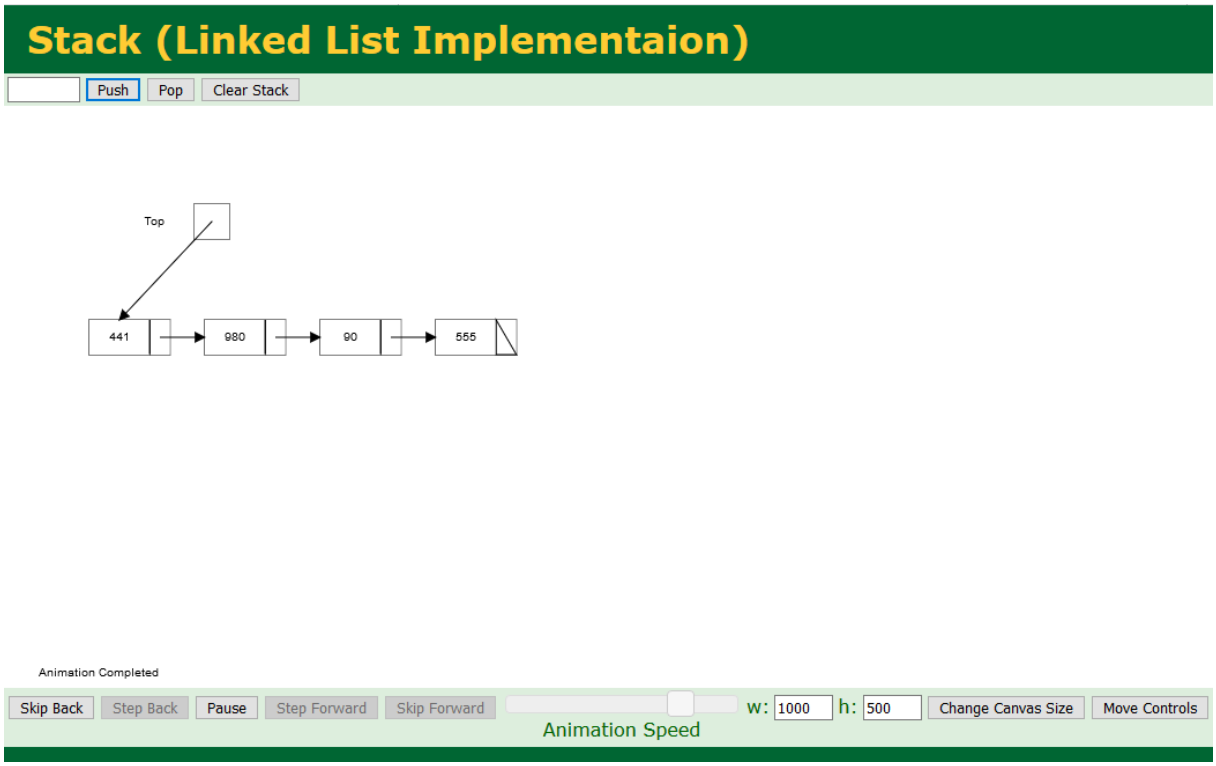


Figura 20: Pilha criada no *Data Structure Visualizations online*
Fonte: Autor

4. O SIMULADOR SYM

Este trabalho tem como objetivo desenvolver um simulador que permita a compreensão do funcionamento dos algoritmos das estruturas de dados do tipo lista encadeada e seus tipos especiais, como as pilhas e filas, que fazem parte da disciplina de estrutura de dados, com a visualização passo a passo da execução do código, para que seja facilitada a compreensão e seja menos necessária a intervenção do professor no aprendizado.

Normalmente, o conteúdo é exposto de maneira teórica, partindo-se em seguida para exercícios de programação, sem que o aluno tenha conseguido assimilar bem o funcionamento. Isso pode ocorrer não só devido à forma tradicional de ensino, mas também por conta do tempo que é possível destinar aos assuntos, levando-se em conta que a disciplina dispõe de muitos conteúdos que necessitam de uma quantidade razoável de tempo. Com o simulador, será possível não só a utilização pelo aluno e pelo professor durante as aulas para facilitação do aprendizado, como o uso posterior pelo próprio aluno durante seus estudos individuais, o que deve permitir um rendimento maior das turmas. A figura 21 demonstra o diagrama de casos de uso do sistema.

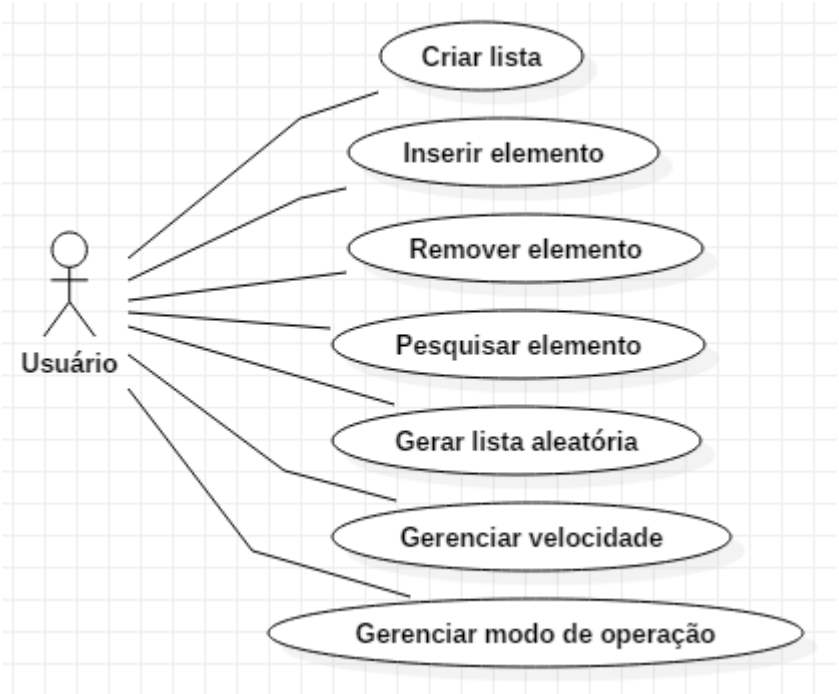


Figura 21: Diagrama de casos de uso
Fonte: Autor

4.1 MOTIVAÇÃO DO PROJETO

Apesar da existência de simuladores com objetivos semelhantes, como os que foram mencionados no capítulo anterior, o SYM apresenta como principal diferencial o fato de possuir

foco na execução completa do código, e não na simples simulação sem código ou utilizando pseudocódigo como mero complemento, como é o caso dos simuladores citados. Com isso, todas as partes dos códigos de cada operação realizada, inclusive variáveis que indicam início e fim da estrutura e variáveis auxiliares de busca, são representadas visualmente, para que o aluno possa compreender o que cada uma delas representa na estrutura. Além disso, também há uma representação simplificada da alocação dos elementos da estrutura na memória, para que o aluno possa compreender como funciona a alocação dinâmica dos nós. Também são exibidas as variáveis temporárias utilizadas para contagem de índices na área de memória.

Considerando que boa parte da dificuldade dos alunos costuma estar exatamente na compreensão do código, o SYM pode ser extremamente útil para melhorar o rendimento das turmas da disciplina estrutura de dados, complementando o que foi ensinado pelo professor, e permitindo que o aluno possa aprender mais com uma menor intervenção do professor, inclusive fora da sala de aula.

O SYM permite que o usuário tenha controle da execução do código, podendo selecionar quatro velocidades diferentes ou utilizar a execução manual, que permite que o aluno avance a execução no momento que desejar, sem depender de intervalos pré-determinados no sistema, o que permite que o usuário adapte o funcionamento do sistema conforme sua necessidade, para que possa obter maior aproveitamento dos recursos do sistema para o aprendizado.

Outra vantagem do SYM é o fato de ter sido desenvolvido para plataforma *web*, e não *desktop*, o que torna desnecessária a instalação de programas adicionais, bastando um navegador *web* para executá-lo. Isso permite que o sistema seja facilmente utilizado em qualquer sistema operacional sem necessidade de nenhum tipo de adaptação. Além disso, o SYM não necessita de conexão à internet, uma vez que é totalmente desenvolvido em tecnologias *front-end*, como a linguagem Javascript, e utiliza somente bibliotecas locais, que acompanham o sistema. Isso o torna muito mais acessível, evitando que o uso seja impossibilitado nas aulas até mesmo em momentos de instabilidade na conexão à internet e facilitando o uso posterior pelos próprios alunos para fins de estudo.

4.2 DESCRIÇÃO DO FUNCIONAMENTO

O SYM tem sua tela dividida verticalmente em três partes: o cabeçalho, onde se encontram as principais opções de controle do sistema, como a escolha do tipo de estrutura e a seleção das operações desejadas; o meio, onde são demonstradas todas as partes da simulação, desde o código até a memória; e o rodapé, onde se encontram algumas opções auxiliares de

controle, como o controle de velocidade e o de avanço manual ou automático do sistema.

O sistema proporciona três formas diferentes de visão de cada uma das estruturas de dados disponíveis, dispostas em três áreas, que demonstram o código, a representação visual e a memória, possibilitando que o usuário adquira um conhecimento completo sobre cada uma das estruturas. A figura 22 apresenta o sistema com todas as áreas em funcionamento.

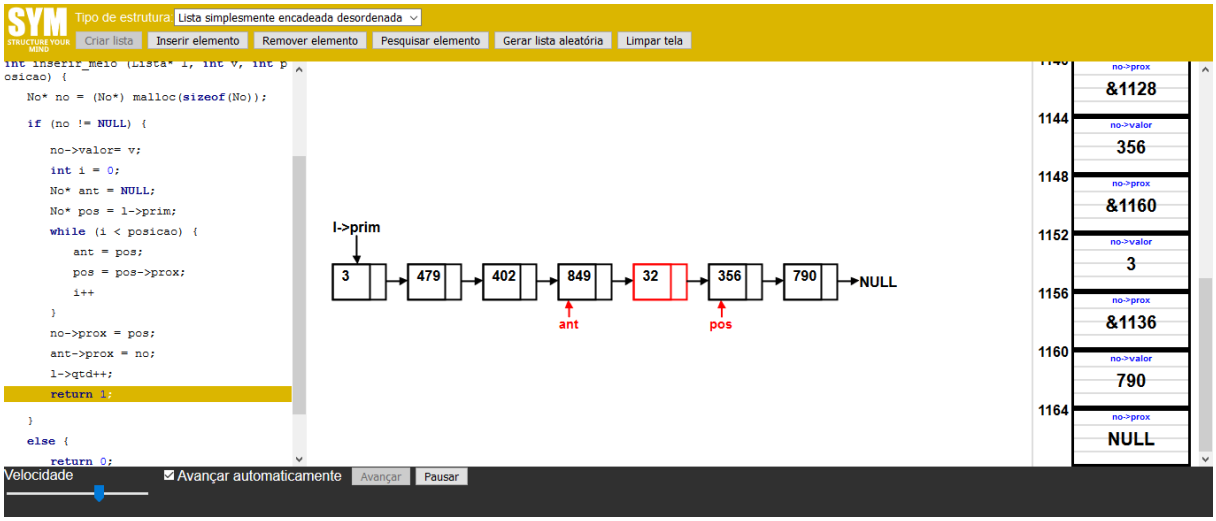
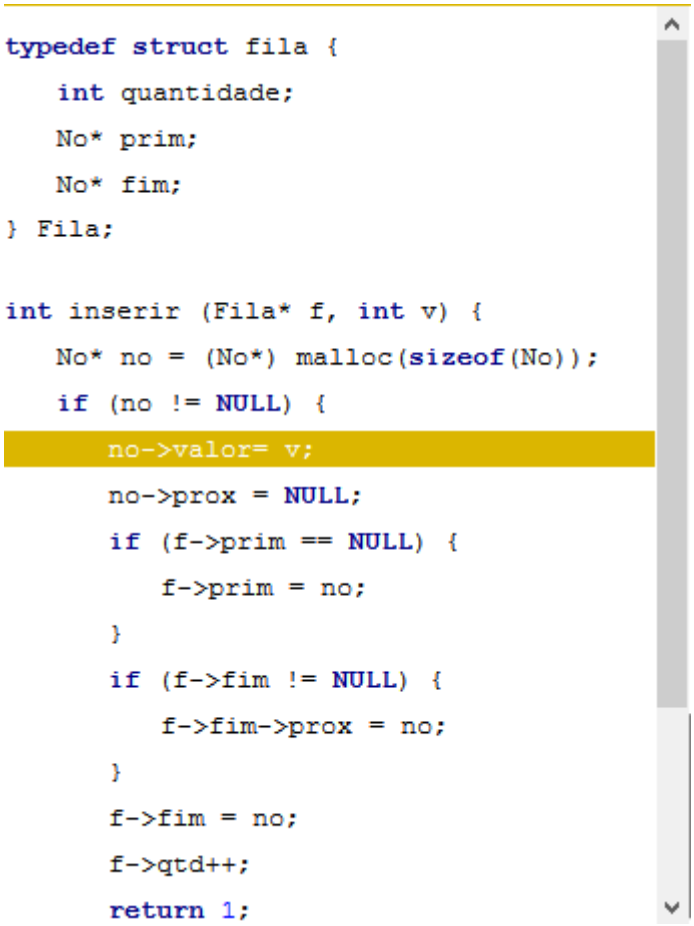


Figura 22: Tela do SYM em funcionamento
Fonte: Autor

A primeira delas é a visão de código, disponível na área branca à esquerda da tela, onde o usuário pode visualizar linha a linha o código da operação que está sendo realizada, carregado no momento em que o usuário solicita a operação, enquanto acompanha a respectiva execução nas outras duas visões. A linha que está sendo executada é marcada por um cursor na cor amarela, da mesma maneira que ocorre em sistemas que possibilitam *debug* de código, como IDEs (*Integrated Development Environment*) e alguns navegadores web que dispõem do recurso. Isso possibilita que o usuário obtenha um melhor aprendizado sobre o que acontece a cada passo do código, já que muitas vezes isso não fica claro o suficiente em uma aula expositiva comum, e torna a exposição do conteúdo em sala muito mais simples, não sendo necessário que o professor demonstre o código de maneira estática em várias etapas. Nessa área, o principal diferencial do SYM em relação à aplicação VisuAlgo, apresentada no capítulo 4 é a demonstração de um código completo em uma linguagem de programação real, no lugar de pseudocódigo. A área de código é demonstrada na figura 23.



```
typedef struct fila {
    int quantidade;
    No* prim;
    No* fim;
} Fila;

int inserir (Fila* f, int v) {
    No* no = (No*) malloc(sizeof(No));
    if (no != NULL) {
        no->valor= v;
        no->prox = NULL;
        if (f->prim == NULL) {
            f->prim = no;
        }
        if (f->fim != NULL) {
            f->fim->prox = no;
        }
        f->fim = no;
        f->qtd++;
        return 1;
```

Figura 23: Área de código do SYM com um código em execução
Fonte: Autor

A segunda visão disponível no sistema é a da simulação gráfica, mostrada na figura 24, e localizada no meio da tela, onde é possível visualizar graficamente tudo que ocorre internamente em toda a estrutura após cada passo dado na execução do código marcado pelo cursor no momento, o que proporciona uma forma muito mais didática de aprendizado. Cada elemento inserido na estrutura é representado por um retângulo, que contém o valor armazenado no nó. As ligações entre os nós são representadas por setas, que se movimentam sempre que necessário para manter os nós ligados e, consequentemente, em conformidade com o que está sendo executado no código.

Os ponteiros são indicados por setas com indicação textual de qual ponteiro representam, como a seta que indica o início da estrutura, indicada na figura 24 como “l->prim”. Os ponteiros que representam variáveis temporárias auxiliares são indicados na cor vermelha, e se movimentam constantemente, por meio de animação, conforme seu valor é alterado no código. Além disso, o SYM sempre mantém representação gráfica de tudo que é necessário durante a execução, demonstrando inclusive quando o nó ou ponteiro aponta para NULL,

situação em que o ponteiro está vazio. A forma como os ponteiros são mostrados e movimentados é um diferencial em relação a ambas as aplicações apresentadas anteriormente, uma vez que possuem somente demonstração textual.

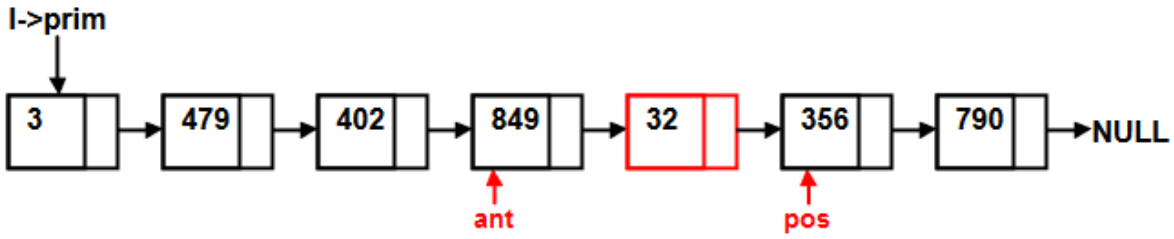


Figura 24: Parte da área gráfica mostrando uma lista simplesmente encadeada após uma inserção
Fonte: Autor

Por fim, a terceira visão possível é a do comportamento da memória durante a execução do código, localizada à direita da tela, onde é possível visualizar a alocação dos espaços necessários para cada um dos nós, que é feita de maneira aleatória visando permitir que o aluno perceba que a alocação não ocorre de maneira contígua, como no caso dos vetores. No momento que uma estrutura é criada, a área de memória é preenchida com vários espaços de memória gerados com endereços aleatórios, onde a alocação das variáveis auxiliares e das referentes à estrutura da lista é realizada nas primeiras posições, e, conforme a criação dos elementos, a alocação dos nós é realizada em qualquer um dos demais endereços.

Cada espaço de memória recebe, também, o nome da variável que armazena, visando facilitar a identificação pelo aluno. A figura 25 mostra a área de memória após a inserção do elemento 743 em uma lista simplesmente encadeada desordenada, que tem sua área destacada na cor vermelha até que ocorra a próxima operação. Os endereços de memória são representados por meio do caractere “&”, seguido do número. É possível observar que, apesar de o elemento destacado ser o último inserido, há nós inseridos anteriormente que ocupam posições totalmente distintas, antes e depois da área alocada para o nó.

É importante ressaltar que a alocação é realizada de maneira extremamente simplificada, unicamente com a finalidade de permitir que o aluno visualize a alocação aleatória e a interligação entre os elementos, não seguindo nenhum dos algoritmos de alocação utilizados em sistemas operacionais.

Outra informação importante é o fato de que esse é um dos principais diferenciais do SYM, tendo em vista nenhuma das aplicações apresentadas possui uma forma de visualização do armazenamento estruturas dentro da memória.

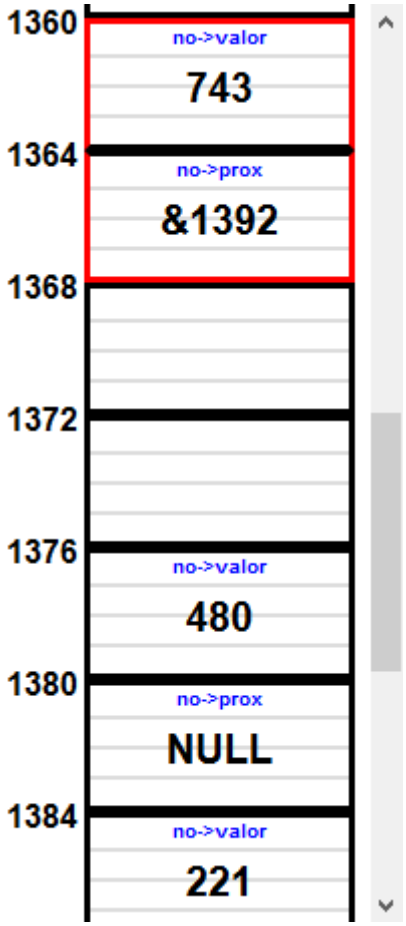


Figura 25: Área de memória após a alocação do elemento 743
Fonte: Autor

Para iniciar a simulação, o usuário poderá escolher um dos tipos de estrutura disponíveis no sistema na caixa “Tipo de estrutura”, localizada na parte superior esquerda do sistema. Os tipos disponíveis são:

- Lista simplesmente encadeada desordenada
- Lista simplesmente encadeada ordenada
- Lista duplamente encadeada
- Pilha
- Fila

Ao ser iniciado, por padrão, o sistema seleciona a estrutura “lista simplesmente encadeada desordenada” para a simulação. Com base na estrutura selecionada, é necessário que o usuário crie a estrutura antes de realizar qualquer operação. Isso é possível através do botão “Criar lista”, localizado abaixo da caixa de seleção do tipo de estrutura. A figura 26 exibe a tela do sistema em seu estado inicial.

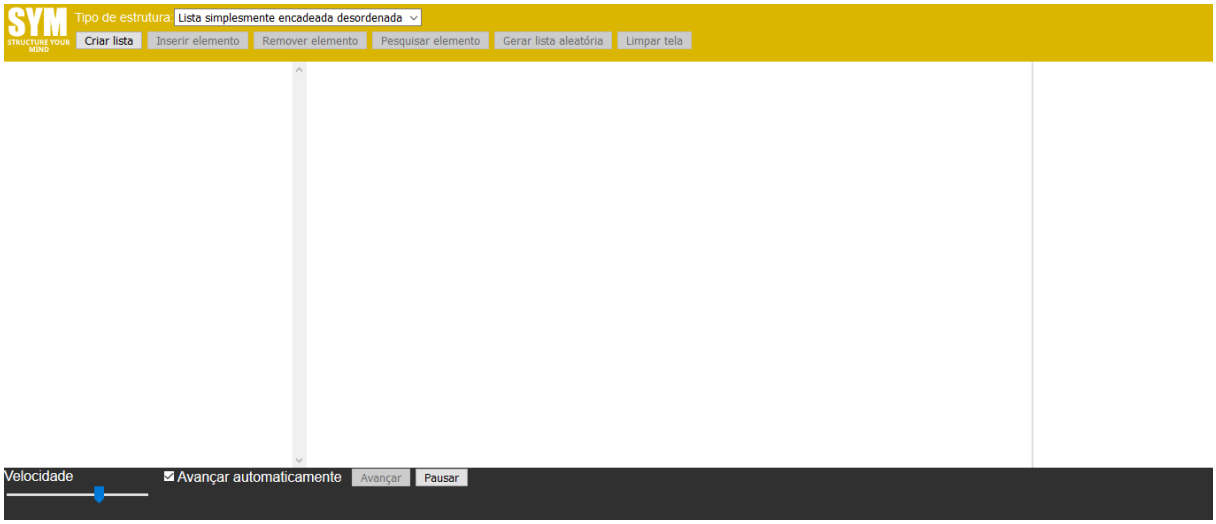


Figura 26: Tela do sistema em seu estado inicial
Fonte: Autor

Ao ser criada a estrutura, será inserido o ponteiro referente ao início e, caso se trate de uma fila, ao fim da lista, que apontarão para NULL neste momento. Após isso, passará a ser possível realizar uma das operações disponíveis no sistema: inserção, remoção e pesquisa, além da geração de elementos aleatórios. Em todos os casos, a estrutura selecionada terá um limite de 7 elementos, uma vez que a finalidade do sistema é unicamente facilitar o aprendizado.

Para qualquer simulação no sistema, será possível que o usuário controle a velocidade da execução automática ou que desative a execução automática e controle manualmente o avanço da simulação, através dos controles localizados na barra inferior, para que seja possível um melhor acompanhamento de cada etapa que está ocorrendo. Também é possível que o usuário pause a execução durante a execução automática.

Ao selecionar a opção “Inserir elemento”, será aberta uma pequena caixa, mostrada na figura 27, onde o usuário deverá informar o valor do elemento a ser inserido, e, caso se trate de uma lista simplesmente encadeada desordenada ou duplamente encadeada, poderá escolher a posição, que poderá ser no início, no meio ou no fim. No caso de inserção no meio é necessário que o usuário informe a posição desejada, que tem início em 0. O valor do elemento deve estar entre 0 e 999. Caso a estrutura seja uma pilha, fila ou lista simplesmente encadeada ordenada, o sistema não exibirá os campos referentes à posição da inserção, uma vez que não há possibilidade de escolha da posição nessas estruturas. Depois de informados o valor e, caso necessário, a posição, o usuário deverá clicar no botão “Inserir elemento” para que o sistema inicie a simulação da inserção.

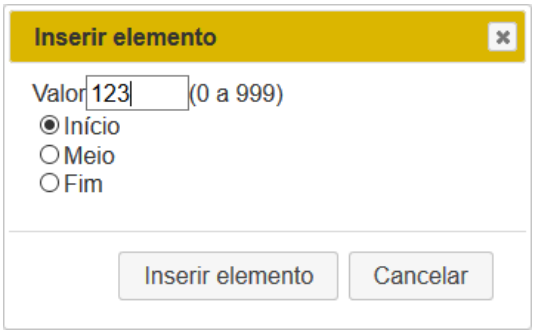


Figura 27: Caixa de inserção de elemento
Fonte: Autor

Iniciada a simulação, primeiramente, o código em linguagem C referente à inserção na estrutura escolhida será inserido na caixa destinada a esse fim, localizada no lado esquerdo da tela. Em seguida, será iniciada a execução passo a passo do código, desde o início da criação do nó, passando pela busca pela posição e chegando ao fim com o retorno da função.

No momento da criação do nó, demonstrada na figura 28, é alocado um espaço na área de memória destacado pela cor vermelha, e é criado um retângulo na área gráfica que, inicialmente, também possui a cor vermelha. Posteriormente, é inserido o valor do elemento, e, em seguida, são criados, caso necessário, os ponteiros auxiliares de busca, que possuem indicadores na cor vermelha. Se necessário, será realizada a busca pela posição onde o elemento será inserido, com a movimentação dos ponteiros conforme os valores das variáveis forem alterados. Em seguida, o novo elemento será colocado abaixo de sua posição, e os elementos seguintes, caso existam, serão movidos para a direita. Por fim, o novo elemento passará a apontar para seu elemento seguinte, o elemento anterior passa a apontar para o novo elemento, e o novo elemento será movido para a altura normal em sua posição.

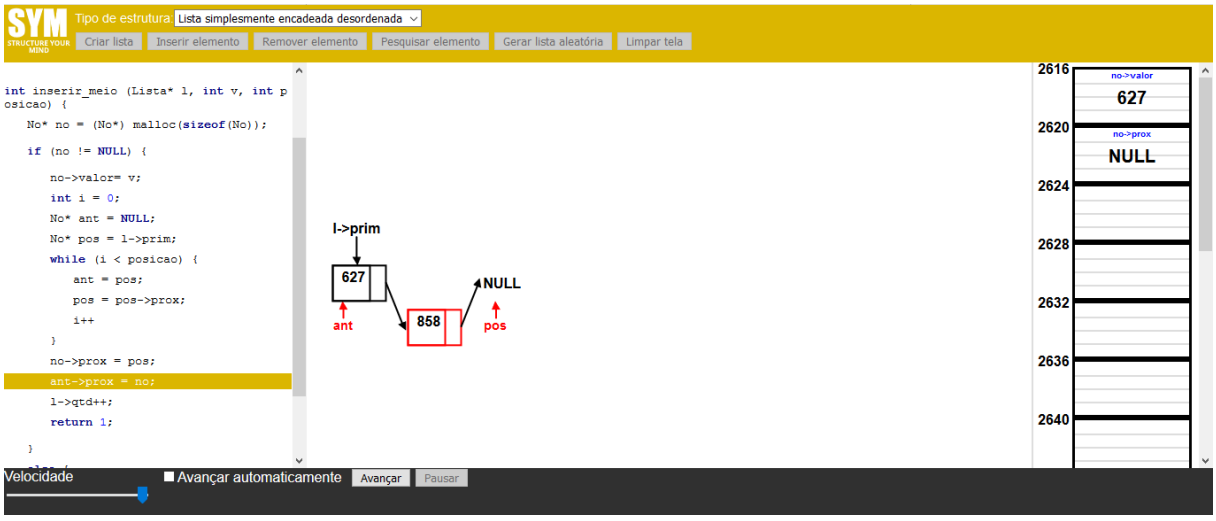


Figura 28: SYM durante a execução de uma inserção

Fonte: Autor

Ao clicar em “Remover elemento”, caso a estrutura selecionada não seja uma pilha ou fila, será exibida uma pequena caixa da mesma forma que ocorre na inserção, solicitando o valor do elemento a ser removido. Caso se trate de uma pilha ou fila, o sistema dará prosseguimento à remoção sem solicitar informação do usuário, por conta da impossibilidade de seleção do elemento a ser removido, sendo removido o primeiro elemento em ambas as estruturas. A caixa exibida é demonstrada na figura 29.

Figura 29: Caixa de remoção de elemento
Fonte: Autor

Da mesma forma que na inserção, na operação de remoção, demonstrada na figura 30, o código em C será carregado e, caso não se trate de uma pilha ou fila, será iniciada a busca pelo elemento informado, sendo indicado visualmente cada elemento percorrido. Caso o elemento seja encontrado, o mesmo será sinalizado e, posteriormente deslocado para baixo. Caso o elemento não seja encontrado, será exibida uma mensagem com a informação. No momento em que a linha de código correspondente à liberação de memória é atingida, a área de memória correspondente é esvaziada e a representação gráfica do elemento é removida.

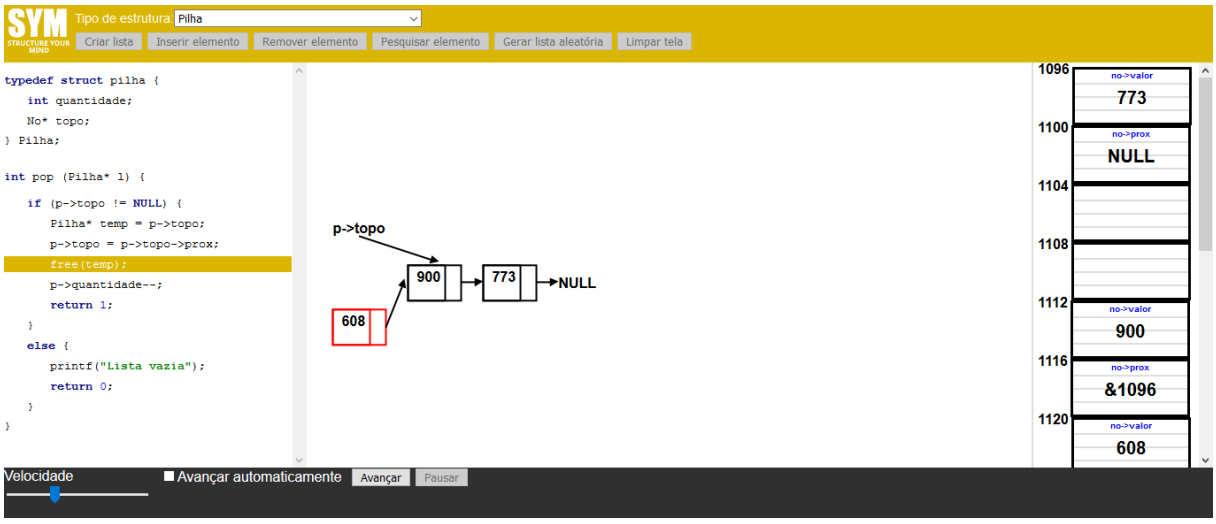


Figura 30: Sistema durante uma operação de remoção em uma pilha

Fonte: Autor

Quando a opção “Pesquisar elemento” é escolhida, o sistema solicita o valor do elemento a ser localizado, em uma caixa idêntica à de remoção de elemento. Ao ser informado, o código de pesquisa em C é carregado e o sistema inicia a busca pelo elemento. Cada elemento percorrido é sinalizado por um marcador, até que seja encontrado o elemento, quando será exibida uma mensagem indicando a posição onde o elemento foi encontrado, ou uma mensagem indicando que o elemento não foi localizado. Como nas outras operações, todos os passos são realizados de forma sincronizada com o código carregado. A figura 31 mostra uma operação de pesquisa no sistema.

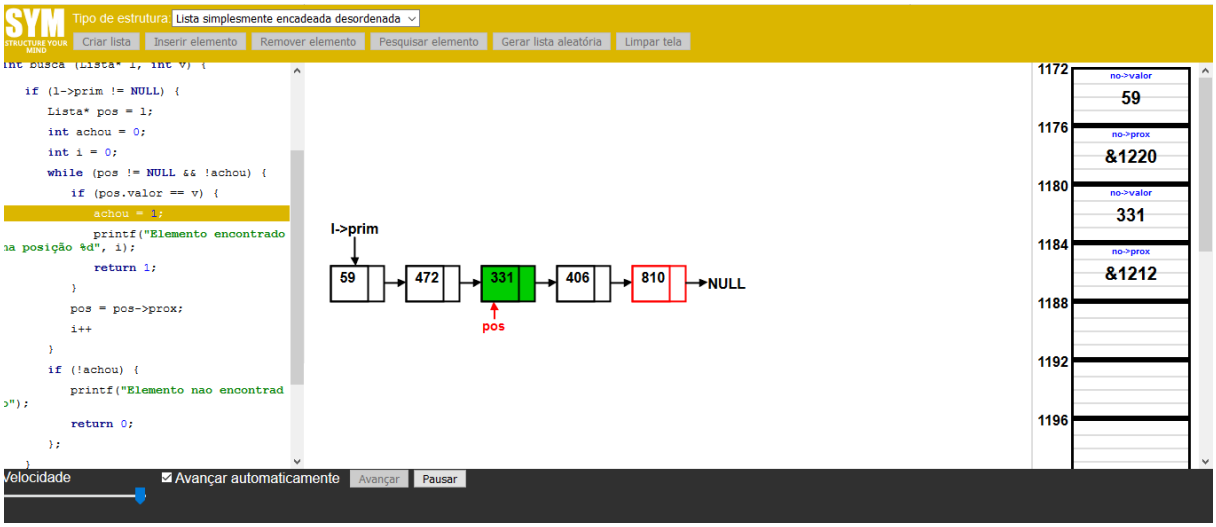


Figura 31: Pesquisa em uma lista simplesmente encadeada desordenada
Fonte: Autor

Quando o usuário clica em “Gerar lista aleatória”, todos os elementos existentes serão apagados, e será exibida a tela exibida na figura 32, onde o usuário deverá informar a quantidade de elementos a serem gerados, limitada a 7. É possível também, exceto em pilhas, filas e listas ordenadas, que o usuário solicite que a inserção seja feita em posições aleatórias, já que, por padrão, a inserção é feita sempre no final.

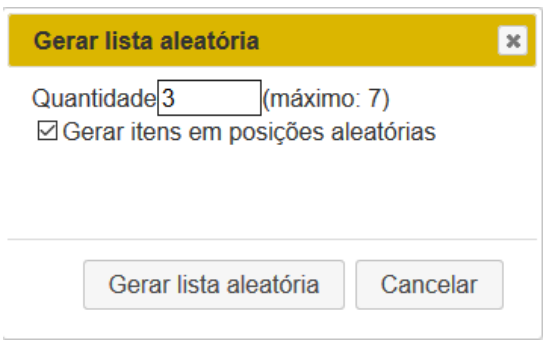


Figura 32: Caixa de geração de lista aleatória

Fonte: Autor

Depois de o usuário solicitar a geração da lista, será gerada a quantidade de elementos informada, utilizando-se de números de 0 a 999 gerados aleatoriamente, repetindo o procedimento de inserção para cada elemento inserido.

Ao clicar na opção “Limpar tela”, o sistema remove todos os elementos existentes, mantendo a lista criada, que passará a apontar para NULL.

Na barra inferior, é possível selecionar a velocidade de execução da simulação do código através da barra deslizante, sendo 4 velocidades disponíveis no total. Também é possível habilitar ou desabilitar a execução automática, através da caixa de seleção, sendo habilitado o botão “Avançar” quando a execução automática estiver desabilitada, para permitir o avanço manual de cada linha de código. A figura 33 mostra a barra inferior do SYM, com os controles de execução.

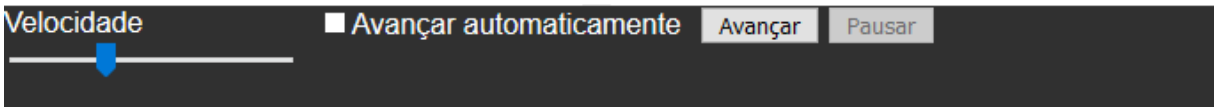


Figura 33: Barra inferior do SYM
Fonte: Autor

4.3 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do SYM, foram utilizadas algumas das principais linguagens de desenvolvimento *front-end*, como HTML, CSS e Javascript. A linguagem de marcação HTML foi utilizada na construção visual do sistema, possibilitando a divisão da página em cabeçalho, conteúdo, rodapé e suas subdivisões. As folhas de estilo CSS foram utilizadas na formatação da aparência de cada um dos elementos da página, determinando suas bordas, cores e posições. A linguagem de programação Javascript foi utilizada em toda a manipulação dos elementos HTML, como as movimentações e animações dos elementos utilizadas para demonstrar o funcionamento de cada uma das estruturas simuladas, além de gerenciar a interação do usuário com a página. A linguagem Javascript foi utilizada com auxílio das bibliotecas SVG.js, responsável por facilitar a manipulação dos elementos gráficos dos nós, e jQuery, utilizada para simplificar o gerenciamento de todas as interações do usuário com os elementos da página.

4.3.1 HTML5

HyperText Markup Language (HTML) é a linguagem de marcação utilizada para a criação de páginas *web*. Foi criada em 1990 por Tim Berners-Lee, com o auxílio de Robert Cailliau, juntamente com o protocolo *HyperText Transference Protocol* (HTTP), utilizado para

a transferência de documentos em uma rede. Foi criada com base na linguagem SGML (*Standard Generalizated Markup Language*), muito utilizada para a estruturação de documentos na época, tendo acrescentado os *links*, recursos que permitem a ligação entre diversas páginas diferentes. (FLATSCHART, 2011)

O HTML possui diversas versões, sendo implementadas diversas melhorias em cada uma delas, e se encontra atualmente na versão 5. A partir da versão atual, passou a haver uma maior preocupação com a semântica do código, ao contrário do que ocorria até o HTML4, com a criação de *tags* que descrevem melhor o conteúdo da página, como as *tags* <*header*> e <*footer*>, que determinam, respectivamente, o cabeçalho e o rodapé da página. Além disso, foram criadas tags como <*canvas*>, destinada à criação de desenhos e animações gráficas.

A tag <*svg*>, utilizada neste trabalho, foi criada antes do HTML5, e, assim como a tag <*canvas*> também é voltada à área gráfica, porém permite melhor controle sobre os elementos, motivo pelo qual foi escolhida.

4.3.2 CSS

Cascade Style Sheets (CSS) são as folhas de estilo das páginas *web*, responsáveis por estilizar a aparência das páginas e elementos HTML. Sua proposta foi criada e apresentada em 1995 por Håkon Wium Lie e Bert Bos, apoiada pelo W3C, consórcio responsável pela padronização da web. Com as folhas de estilo CSS é possível formatar diversas áreas de cada elemento e formatar inúmeros elementos de uma só vez, facilitando o design da página. Também possibilita que o código HTML se torne mais limpo, devido à possibilidade de separação entre a formatação e o código HTML, inclusive em arquivos diferentes, e facilita o reuso da formatação.

Atualmente encontra-se na versão 3, que acrescentou diversos recursos mais avançados e bastante úteis, como novos tipos de organização de elementos e recursos de animação.

4.3.3 Javascript

Javascript é uma linguagem de programação criada no ano de 1995, responsável pelo comportamento das páginas *web*, funcionando no próprio navegador *web* do usuário. Permite o gerenciamento de interações do usuário com a página e com cada um de seus elementos.

4.3.4 jQuery

A biblioteca Javascript jQuery[3] foi criada no ano de 2006 por John Resig, e visa facilitar

[3] https://jquery.com/

o uso de diversas funções bastante importantes do Javascript, como a manipulação de elementos HTML, captura de eventos, requisições Ajax, e diversas outras funcionalidades. É possível manipular uma grande quantidade de elementos com um número bastante reduzido de linhas de código, o que não ocorre quando se programa em Javascript sem o uso a biblioteca, como, por exemplo, a manipulação de todos os objetos que contêm uma classe específica. Além disso, a manipulação de elementos é feita utilizando a sintaxe CSS, o que torna o aprendizado muito mais fácil, já que não é necessário aprender uma sintaxe completamente nova para seu uso.

4.3.5 SVG.js

SVG.js[4] é uma biblioteca Javascript criada no ano de 2012 por Wout Fierens, e visa facilitar o uso dos recursos do elemento HTML <svg>, fornecendo diversas propriedades e funções específicas, como as que permitem a movimentação, rotação ou redimensionamento dos elementos, e a animação de cada uma dessas operações. Uma grande vantagem dessa biblioteca é o fato de não possuir dependência de nenhuma outra biblioteca, o que permite maior portabilidade e a torna muito mais simples de instalar.

O uso da biblioteca reduz consideravelmente a quantidade de código necessária para a criação e manipulação dos elementos SVG em comparação com a quantidade quando se utiliza somente código Javascript puro, como demonstram os códigos nas figuras 34 e 35, que demonstram, respectivamente, códigos que realizam a criação de um retângulo sem e com o uso da biblioteca.

```
var ns = 'http://www.w3.org/2000/svg'
var div = document.getElementById('drawing')
var svg = document.createElementNS(ns, 'svg')
svg.setAttributeNS(null, 'width', '100%')
svg.setAttributeNS(null, 'height', '100%')
div.appendChild(svg)
var rect = document.createElementNS(ns, 'rect')
rect.setAttributeNS(null, 'width', 100)
rect.setAttributeNS(null, 'height', 100)
rect.setAttributeNS(null, 'fill', '#f06')
svg.appendChild(rect)
```

Figura 34: Exemplo de código Javascript puro para criação de um retângulo SVG
Fonte: Site do SVG.js

```
var draw = SVG('drawing')
  , rect = draw.rect(100, 100).fill('#f06')
```

Figura 35: Exemplo de código Javascript com SVG.js para criação de um retângulo SVG

[4] https://svgjs.com

Fonte: Site do SVG.js

A biblioteca dispõe de funções simples para criação e manipulação dos elementos SVG, onde na maioria dos casos somente é necessário informar como parâmetros a posição e tamanho do elemento.

Para criar o elemento HTML <svg> e o controle da biblioteca, utiliza-se a função SVG(id), onde em id é informado um elemento HTML, como uma div, que deve ser substituído pelo elemento <svg>. Deve-se atribuir o retorno dessa função a uma variável, que será utilizada para toda a manipulação dos elementos gráficos SVG. O uso dessa função é mostrado na figura 35.

Após a criação do manipulador dos elementos SVG, deve-se utilizar as funções que criam os elementos gráficos, e estão disponíveis para chamada a partir do manipulador criado inicialmente. Algumas delas são:

- *rect(width, height)*: Serve para a criação de retângulos, sendo o primeiro elemento a largura e o segundo a altura. É bastante utilizada na criação dos nós, já que todos são representados por retângulos.
- *circle(diameter)*: Utilizada para a criação de círculos, sendo o único parâmetro o diâmetro do círculo, e não o raio. Não foi utilizada nesta aplicação.
- *path(string)*: Tem o objetivo de permitir a criação de desenhos livres, sendo o parâmetro uma string que possui regras específicas do elemento HTML <*svg*>, não sendo próprias da biblioteca. É bastante utilizada na criação das setas entre os nós e entre os ponteiros, uma vez que as setas podem possuir diferentes formatos.
- *text(text)*: Possibilita a criação de textos dentro do elemento SVG. É bastante utilizada na aplicação para exibição dos valores dos nós e nomes dos ponteiros, além de outras informações exibidas.

Para a manipulação dos elementos já criados, há as funções que possibilitam a movimentação e preenchimento das bordas e da área, além da função que permite a animação de todas essas operações. As principais funções são:

- *move(x, y)*: Permite a movimentação dos objetos dentro do elemento SVG, sendo o primeiro parâmetro a posição horizontal, e o segundo a posição vertical, ambos iniciando em 0. É necessária sua utilização inclusive na criação dos elementos, uma vez que todos são criados por padrão na posição (0, 0).
- *stroke(color)*: A função *stroke* é utilizada para definir a cor das bordas dos elementos ou das linhas criadas, sendo seu único parâmetro a string RGB que determina a cor desejada.

- *fill(color)*: A função *fill* é utilizada para definir a cor do preenchimento do elemento, sendo seu único parâmetro a string RGB que determina a cor desejada.
- *animate(duration)*: Utilizada para inserir animação nas operações realizadas nos elementos, como nas funções *move*, *stroke* e *fill*. Deve ser inserida uma única vez antes da chamada das funções a serem animadas sendo as chamadas seguintes encadeadas a ela, como: "*draw.rect(100,100).animate(1000).move(500,500)*". Seu único parâmetro é a duração da animação em milissegundos, que por padrão será de 1 segundo (1000ms).

4.4 PESQUISA DE OPINIÃO

No dia 27 de maio de 2019 foi realizada uma apresentação do simulador desenvolvido neste trabalho, no estado em que se encontrava na ocasião, para uma das turmas da disciplina de estrutura de dados II do professor André Lucio. Após uma breve explicação sobre o objetivo do sistema e de seu funcionamento, foi passado um link para os alunos onde poderiam obter uma cópia do sistema para que pudessem testá-lo e avaliá-lo. Para essa avaliação, foi disponibilizado um formulário com uma pesquisa contendo cinco perguntas sobre a aplicação, sendo quatro objetivas e uma aberta para sugestões de mudanças ou novas funcionalidades, caso tivessem. A pesquisa teve um total de 25 participantes.

A primeira pergunta feita na pesquisa foi: "Qual o seu nível de dificuldade para entender os conceitos da disciplina estrutura de dados utilizando a metodologia tradicional de ensino (Teoria em apresentação PowerPoint + desenvolvimento de código no quadro)?", e teve como objetivo embasar a necessidade ou não de um meio mais claro para transmissão do conteúdo nas aulas. A pergunta obteve como maioria absoluta a resposta "Média", com 56%, seguida da resposta "Alto", com 28%, o que é mostrado na figura 36. Isso demonstra que a forma tradicional de ensino pode não possuir uma grande eficiência para uma enorme parte da turma, o que torna necessário o uso de recursos visuais que tornem mais simples o ensino da disciplina.

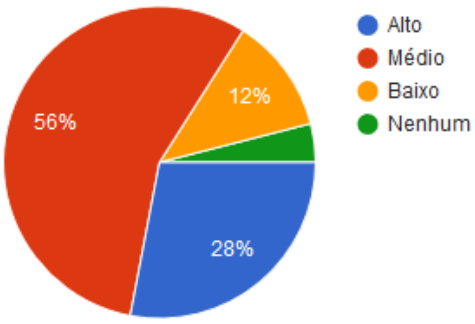


Figura 36: Resultado da primeira pergunta da pesquisa
Fonte: Autor

A segunda pergunta foi uma avaliação do quanto a aplicação poderia contribuir para o ensino da disciplina, com o texto “Em uma escala de 0 a 10, quanto esta aplicação pode contribuir para o aprendizado da disciplina estrutura de dados?”. Todas as notas atribuídas à aplicação foram a partir de 8, tendo a maioria absoluta da turma (64%) atribuído a nota 10, nota máxima dentro da escala da pergunta, o que comprova que o uso de uma aplicação de simulação, e, mais especificamente, o SYM, teria uma grande eficiência no auxílio do aprendizado da disciplina. Em seguida, estão as notas 9, com um total de 32%, e 8, com soma de 4%. Os resultados dessa pergunta condizem com os obtidos na primeira, uma vez que a maioria entende que a forma tradicional não tem uma grande eficiência na transmissão do conteúdo da disciplina. O resultado da segunda pergunta é apresentado na figura 37.

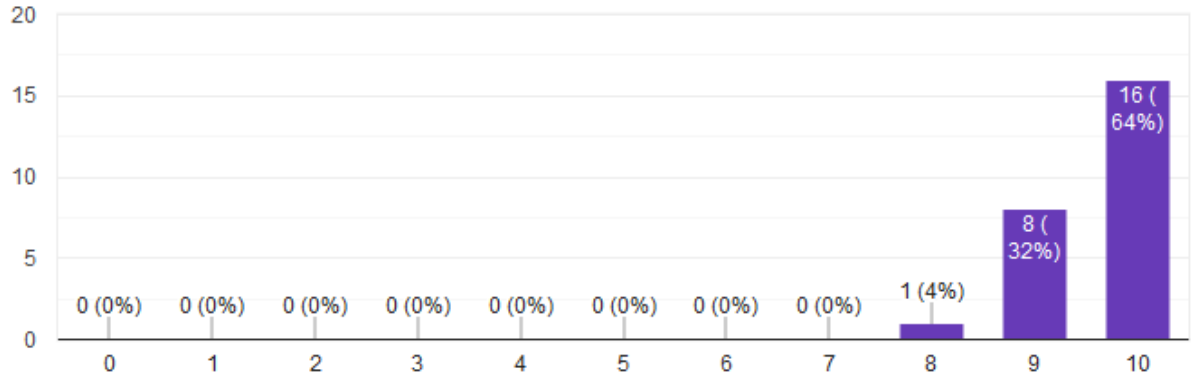


Figura 37: Resultado da segunda pergunta da pesquisa
Fonte: Autor

A interface e usabilidade do sistema foram os itens avaliados na terceira pergunta, que possuía o texto: “Em uma escala de 0 a 10, o quanto você considera a interface dessa aplicação intuitiva e com boa usabilidade?”. A pergunta teve como objetivo sinalizar possíveis dificuldades no uso da aplicação, e, em caso positivo, possibilitar a adoção de medidas para aperfeiçoamento da interface e da usabilidade. A nota atribuída pela maioria dos participantes foi 8, com 36%, seguida de 10, com 28%, e 9, com um total de 20%. Por último, e em menor número, foram atribuídas as notas 5 e 6, com 4% cada, e 7, com 8%.

Uma vez que as notas estão quase integralmente concentradas a partir de 8, pode-se considerar que o sistema obteve uma ótima avaliação, necessitando apenas de alguns ajustes. Cabe ressaltar que alguns recursos não funcionavam corretamente na ocasião da apresentação, o que pode ter sido levado em conta por alguns dos alunos, apesar de esse fato ter sido comunicado desde o início da apresentação. A figura 38 mostra os resultados obtidos na terceira pergunta.

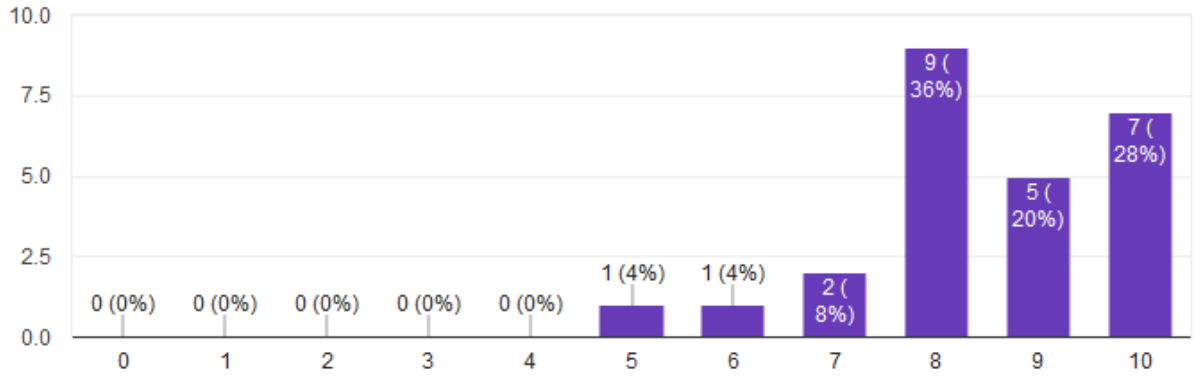


Figura 38: Resultado da terceira pergunta da pesquisa
Fonte: Autor

A penúltima pergunta da pesquisa teve como objetivo obter a opinião dos alunos a respeito de um possível uso da aplicação em Ensino a Distância (EAD), com o texto “Você acha que uma aplicação como essa poderia ser utilizada como ferramenta de aprendizagem para a disciplina estrutura de dados em um ambiente EAD?”. É constante a reclamação sobre a falta de interação com o conteúdo em disciplinas EAD, pois na maioria das vezes o conteúdo é disponibilizado na forma de slides e vídeos, ficando o aluno como mero receptor do conteúdo, o que motivou a inserção da pergunta. Nessa pergunta, especificamente, foi questionado se, em caso positivo, seria possível o uso sem necessidade de auxílio de um professor, com a disponibilização de um tutorial. Todos os alunos responderam positivamente a essa possível forma de uso, tendo a maioria (56%) respondido que seria possível o uso somente com um bom tutorial, sem a necessidade de professor, o que demonstra que a aplicação pode ser bastante eficaz mesmo no uso individual. Além disso, o resultado demonstra que a aplicação seria bastante útil inclusive no aprendizado virtual, e que possui uma boa usabilidade, reforçando as respostas à terceira pergunta, que, em maioria, indicaram que a aplicação possui uma boa interface e usabilidade. O resultado da quarta pergunta é demonstrado na figura 39.

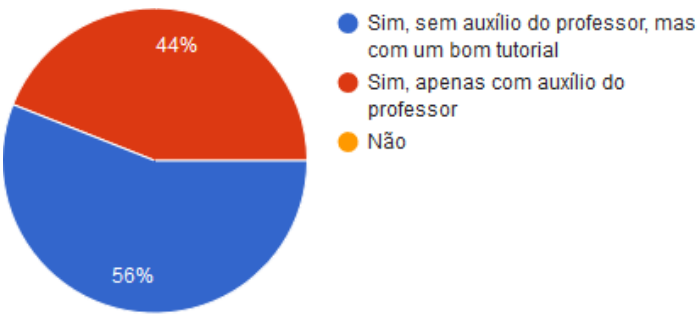


Figura 39: Resultados da quarta pergunta da pesquisa
Fonte: Autor

A última pergunta foi uma oportunidade de os alunos fornecerem sugestões de melhoria,

caso tivessem alguma, com o texto “Você tem alguma sugestão de melhoria para essa aplicação? Qual”. Boa parte dos alunos respondeu à pergunta, tendo alguns somente elogiado e outros fornecido pequenas sugestões que foram bastante úteis. Algumas das sugestões foram implementadas de imediato, dada sua complexidade mínima, e outras foram inseridas nos trabalhos futuros, tendo em vista que não havia tempo hábil para sua implementação, apesar de sua aparente simplicidade. Foi feita uma síntese das sugestões, onde foram agrupadas as sugestões contendo a mesma ideia e suas respectivas quantidades. A tabela 3 mostra as sugestões apresentadas, as quantidades e a informação sobre sua implementação ou não na aplicação.

Tabela 3: Sugestões apresentadas na pesquisa e informação sobre sua implementação
Fonte: Autor

Síntese da sugestão	Quantidade	Situação
Inserção do botão “Pausar”	5	Implementada
Inserção do botão “Voltar”	3	Não implementada
Geração de elementos aleatórios em posições aleatórias	1	Implementada
Descrição textual de cada passo realizado	2	Não implementada
Simular código em mais de uma linguagem de programação	1	Não implementada
Simulação de árvores	1	Não implementada

As sugestões mais simples foram implementadas, como a geração de elementos aleatórios em posições aleatórias, pois até então a inserção era realizada sempre no final da estrutura, e a inserção do botão “pausar”. Algumas das outras sugestões foram incluídas nas possibilidades de trabalhos futuros, como a simulação de árvores, que já estava planejada anteriormente.

5 CONCLUSÃO E TRABALHOS FUTUROS

Com base nas pesquisas realizadas neste trabalho, é possível concluir que há um grande avanço no uso de metodologias ativas na educação, onde o aluno passa a ser o agente principal do aprendizado, com menor intervenção do professor, como é o caso da simulação. Contudo, foi encontrada maior predominância do uso de metodologias ativas na área da medicina, onde os simuladores são frequentemente utilizados para que os alunos vivenciem as situações do cotidiano de um médico. Na área da computação, existem alguns exemplos de simuladores, no entanto, não aparentam ter uso prático, limitando-se à demonstração em trabalhos acadêmicos, sem aparente uso prático.

Neste trabalho foi desenvolvido o SYM, simulador para auxílio no ensino da disciplina estrutura de dados que possibilita a visualização do funcionamento de cada estrutura de forma gráfica, contemplando o comportamento de cada linha de seus respectivos códigos e da memória. A disciplina possui um nível de dificuldade mais elevado que o de muitas disciplinas da área da computação, principalmente se ensinada do modo tradicional, o que é indicado em pesquisa de opinião realizada em uma das turmas da disciplina, onde somente um aluno alegou possuir não possuir nenhum tipo de dificuldade baixo, tendo a maioria afirmado possuir ao menos dificuldade média. A própria implementação da aplicação foi bastante extensa, tendo em vista a quantidade de passos e detalhes que é necessário mostrar, o que reforça a complexidade do conteúdo da disciplina.

Um ponto importante é o fato de a aplicação ter sido desenvolvida totalmente em linguagens *front-end*, como Javascript, o que dispensa a necessidade de instalação de programas específicos, bastando um simples navegador web para o uso. Além disso, a aplicação não necessita de conexão à internet, o que a torna muito mais acessível, não tendo seu funcionamento prejudicado ou impedido por instabilidades na conexão.

Outro ponto importante é a possibilidade de utilização da aplicação em ambientes EAD, já que a principal característica desse tipo de ensino é o aprendizado individual, e a complexidade da disciplina pode ser uma barreira se a exposição for somente teórica. Com o uso do SYM, é possível que o aluno aprenda de forma autônoma, visualizando cada passo do funcionamento das estruturas, o que pode ser extremamente útil em uma forma de ensino como essa, fato apontado na pesquisa de opinião realizada. Na pesquisa, os alunos votaram de forma unânime pela possibilidade de uso nessa forma de ensino, tendo a maioria, inclusive, votado pela possibilidade de uso sem a necessidade de intervenção de um professor, com o uso de um

tutorial, o que demonstra a eficiência da aplicação no aprendizado autônomo.

Com base no exposto, pode-se afirmar que o SYM possui um grande potencial para o auxílio no ensino da disciplina estrutura de dados, tornando as aulas mais interativas e menos teóricas, e possibilitando inclusive o aprendizado de uma forma mais autônoma, fato que também foi apontado pela maioria na pesquisa realizada. É necessário, porém, que haja uma aplicação prática e um maior número de pesquisas de uso para que se possa obter índices mais precisos de avaliação e adequar melhor o sistema a seus usuários, o que será realizado pelo professor André em suas turmas.

Como proposta de trabalhos futuros podem ser citados os seguintes tópicos: manutenção evolutiva para desenvolvimento de simulações de árvores, como binária, AVL e B, utilizando a mesma estrutura da simulação das listas; possibilidade de voltar durante a execução das simulações; descrição textual de cada passo realizado, em forma de *log*.

REFERÊNCIAS

ASCENSIO, Ana Fernandes Gomes; CAMPOS, Edilene Aparecida Veneruchi de. Fundamentos da programação de computadores. São Paulo: Prentice Hall, 2002

ASCENSIO, Ana Fernandes Gomes; ARAÚJO, Graziela Santos de. Estruturas de dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Prentice Hall, 2010

BASTOS, Celso da Cunha. Metodologias ativas. 2006. Disponível em: <http://educacaoemedicina.blogspot.com.br/2006/02/metodologias-ativas.html>. Acesso em: 03 out. 2018.

BEZ, Marta Rosecler; VICARI, Rosa Maria; FLORES, Cecília Dias. Métodos Ativos de Aprendizagem: Simulador de Casos Clínicos – SimDeCs. RETEME, São Paulo, v.2, n. 2, p. 146-166, jan./jun. 2012.

COSTA, Raphael Raniere de Oliveira; MEDEIROS, Soraya Maria de; MARTINS, José Carlos Amado; MENEZES, Rejane Maria Paiva de; ARAÚJO, Marília Souto de. O uso da simulação no contexto da educação e formação em saúde e enfermagem: Uma reflexão acadêmica. Revista Espaço para a Saúde, Londrina, v.16, n. 1, p. 61, jan. 2005.

FLATSCHART, Fábio. HTML 5 – Embarque Imediato. São Paulo: Brasport, 2011

FLORES, Cecilia Dias; BEZ, Marta Rosecler; BRUNO, Rosana Mussoi. O Uso de Simuladores no Ensino da Medicina In VIII Congresso Brasileiro de Ensino Superior a Distância. Anais da ESUD 2011 - VIII Congresso Brasileiro de Ensino Superior a Distância, Ouro Preto, v.1, p. 1-10, out. 2011.

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. Lógica de programação: a construção de algoritmos e estruturas de dados. 3 ed. São Paulo: Prentice Hall, 2005

GRANDO, Regina Célia. O conhecimento matemático e o uso de jogos na sala de aula. Tese de doutorado, Faculdade de Educação, UNICAMP, Campinas, 2000

LAUREANO, Marcos. Estrutura de dados com Algoritmos e C. Rio de Janeiro: Brasport, 2008

LIMA, Alessandro; STAHNKE, Fernando; BARROS, Paulo; BENETTI, Diego; MELLO,

Blanda; BEZ, Marta; CERVI, Gustavo. Projeto para desenvolvimento do Simulador Health Simulator In Computer on the Beach, 2015, São José. Anais do Computer on the Beach 2015. São José, 2015. 279-288. Disponível em <https://siaiap32.univali.br/seer/index.php/acotb/article/view/7043>. Acesso em: 19 out. 2018.

MAIA, LUIZ PAULO. SOsim: Simulador para o Ensino de Sistemas Operacionais. Rio de Janeiro, 2001

MASETTO, Marcos Tarciso. Competência pedagógica do professor universitário. São Paulo: Summus, 2012

MILLÃO, Luzia Fernandes; VIEIRA, Tainara Wink; SANTOS, Natália Domigues dos; SILVA, Ana Paula Scheffer Schell da Silva; FLORES, Cecília Dias. Integração de tecnologias digitais no ensino de enfermagem: criação de um caso clínico sobre úlceras por pressão com o software SIACC. Revista Eletrônica de Comunicação, Informação e Inovação em Saúde, Rio de Janeiro, v. 11, n. 1, jan.-mar. 2017.

PEREIRA, Silvio do Lago: Estruturas de dados fundamentais - Conceitos e Aplicações. São Paulo: Editora Érica, 1996

SOARES, Tays C. A. P., CORDEIRO, Eduardo S., STEFANI Ítalo G. A., TIRELO, Fabio. Uma Proposta Metodológica para o Aprendizado de Algoritmos em Grafos Via Animação Não-Intrusiva de Algoritmos. III Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG 2004). Belo Horizonte, MG, Brasil.