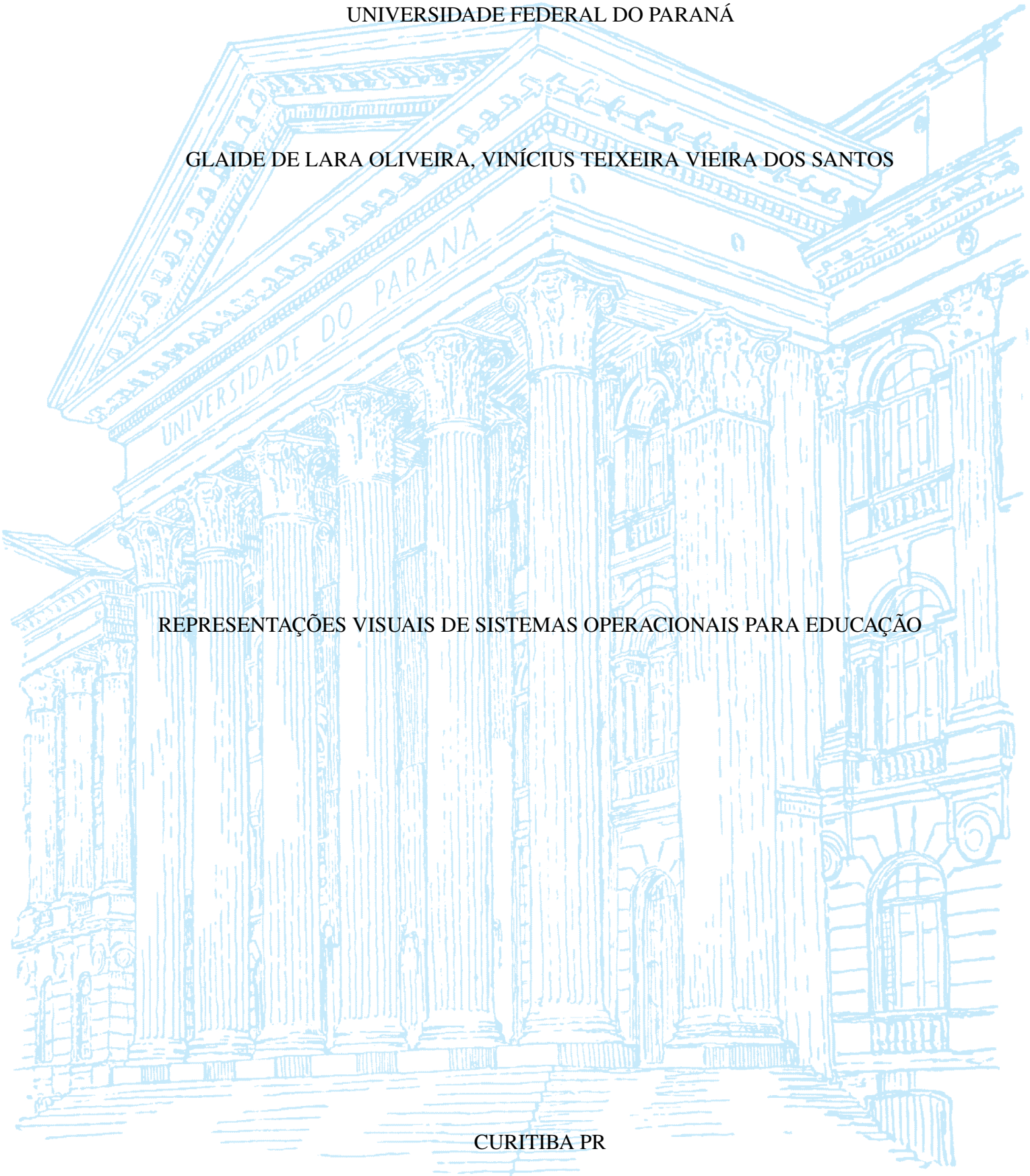
UNIVERSIDADE FEDERAL DO PARANÁ

GLAIDE DE LARA OLIVEIRA, VINÍCIUS TEIXEIRA VIEIRA DOS SANTOS



REPRESENTAÇÕES VISUAIS DE SISTEMAS OPERACIONAIS PARA EDUCAÇÃO

CURITIBA PR

2023

GLAIDE DE LARA OLIVEIRA, VINÍCIUS TEIXEIRA VIEIRA DOS SANTOS

REPRESENTAÇÕES VISUAIS DE SISTEMAS OPERACIONAIS PARA EDUCAÇÃO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Carlos Alberto Maziero.

CURITIBA PR

2023

Ficha catalográfica

Substituir o arquivo `0-iniciais/catalografica.pdf` pela ficha catalográfica fornecida pela Biblioteca da UFPR (PDF em formato A4).

Instruções para obter a ficha catalográfica e fazer o depósito legal da tese/dissertação (contribuição de André Hochuli, abril 2019. Links atualizados Wellton Costa, Nov 2022):

1. Estas instruções se aplicam a dissertações de mestrado e teses de doutorado. Trabalhos de conclusão de curso de graduação e textos de qualificação não precisam segui-las.
2. Verificar se está usando a versão mais recente do modelo do PPGInf e atualizar, se for necessário (`https://gitlab.c3sl.ufpr.br/maziero/tese`).
3. conferir o *checklist* de formato do Sistema de Bibliotecas da UFPR, em `https://bibliotecas.ufpr.br/servicos/normalizacao/`
4. Enviar e-mail para "`referencia.bct@ufpr.br`" com o arquivo PDF da dissertação/tese, solicitando a respectiva ficha catalográfica.
5. Ao receber a ficha, inseri-la em seu documento (substituir o arquivo `0-iniciais/catalografica.pdf` do diretório do modelo).
6. Emitir a Certidão Negativa (CND) de débito junto a biblioteca, em `https://bibliotecas.ufpr.br/servicos/certidao-negativa/`
7. Avisar a secretaria do PPGInf que você está pronto para o depósito. Eles irão mudar sua titulação no SIGA, o que irá liberar uma opção no SIGA pra você fazer o depósito legal.
8. Acesse o SIGA (`http://www.prppg.ufpr.br/siga`) e preencha com cuidado os dados solicitados para o depósito da tese.
9. Aguarde a confirmação da Biblioteca.
10. Após a aprovação do pedido, informe a secretaria do PPGInf que a dissertação/tese foi depositada pela biblioteca. Será então liberado no SIGA um link para a confirmação dos dados para a emissão do diploma.

Universidade Federal do Paraná
Setor de Ciências Exatas
Curso de Ciência da Computação

Ata de Apresentação de Trabalho de Conclusão de Curso 2

Título do Trabalho: REPRESENTAÇÕES VISUAIS DE SISTEMAS OPERACIONAIS PARA EDUCAÇÃO

Autor(es):

GRR_____ Nome: GLAIDE DE LARA OLIVEIRA

GRR_____ Nome: VINICIUS TEIXEIRA VIEIRA DOS SANTOS

Apresentação: Data: 11 / 12 / 23 Hora: 16:00 Local: Sala 101 - DINF/UFPR

Orientador: CARLOS ALBERTO MAZIERO

Membro 1: BRUNO MÜLLER JR (PROF)

Membro 2: CAIO CARVALHO (DOUTORANDO/UFPR)

(nome) (assinatura)

AVALIAÇÃO – Produto escrito		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Conteúdo	(00-40)				
Referência Bibliográfica	(00-10)				
Formato	(00-05)				
AVALIAÇÃO – Apresentação Oral					
Domínio do Assunto	(00-15)				
Desenvolvimento do Assunto	(00-05)				
Técnica de Apresentação	(00-03)				
Uso do Tempo	(00-02)				
AVALIAÇÃO – Desenvolvimento					
Nota do Orientador	(00-20)		**********	**********	
NOTA FINAL		**********	**********	**********	- 90 -

Os pesos indicados são sugestões.

Conforme decisão do colegiado do curso de Ciência da Computação, a entrega dos documentos comprobatório de trabalho de Conclusão de Curso 2 deve deve respeitar os seguintes procedimentos: o orientador deve abrir um processo no Sistema Eletrônico de Informações (SEI – UFPR); Selecionar o tipo: *Graduação: Trabalho Conclusão de Curso*; informar os interessados: nome do aluno e o nome do orientador; anexar esta ata escaneada e a versão final do PDF da monografia do aluno; Tramitar o processo para CCOMP (Coordenação de Ciência da Computação).

Dedicamos este trabalho a todos os que nos ajudaram ao longo desta caminhada.
Dedico este trabalho à memória da minha amada avó, Maria do Carmo, uma mulher forte e inspiradora.
Glaide

AGRADECIMENTOS

Gostaria de expressar minha gratidão ao meu parceiro e amigo neste projeto, Vinícius. Sua dedicação, apoio e paciência foram fundamentais para o sucesso deste trabalho. Sem a sua colaboração, tenho certeza de que não teria alcançado um resultado de tamanha qualidade. Reconheço e valorizo imensamente seu papel nesta jornada.

Além disso, quero estender meus agradecimentos aos meus pais, Marcia e Gleiton, que sempre batalharam para tornar possível minha entrada em uma universidade pública, e cujo amor e apoio constante foram a motivação que me guiou durante este processo.

Agradeço também ao meu namorado Arthur, por todo seu apoio e companheirismo, especialmente por estar ao meu lado nos momentos decisivos, como nos dias de prova do vestibular para a faculdade.

Aos meus amigos, quero expressar minha gratidão por compartilharem comigo essa jornada e por todo o apoio e incentivo ao longo do caminho.

Por fim, expresso meu profundo agradecimento ao Professor Carlos Alberto Maziero por sua orientação, conselhos valiosos e incentivo constante. Sua dedicação exemplar como educador foi crucial para o desenvolvimento deste trabalho. Sua abordagem pedagógica nos inspira a acreditar em um ensino de qualidade.

A todos vocês, agradeço por fazerem parte deste importante capítulo da minha vida acadêmica.

Glaide

Agradeço primeiramente a todos os professores que estiveram presentes ao longo desta jornada acadêmica, em especial a nosso professor e orientador Carlos Alberto Maziero, que nos auxiliou e proporcionou desenvolver este trabalho, e o professor Eduardo Todt, que me fez interessar nas áreas de robótica, Internet das Coisas e eletrônica. Agradeço a todos por seu tempo e conhecimento para orientar e incentivar meu crescimento intelectual.

Aos colegas de curso agradeço pela troca de experiências, pelo companheirismo e por tudo em que me ajudaram. Cada conversa e debate enriqueceram meu aprendizado e me ajudaram a amadurecer como acadêmico e como pessoa.

Expresso minha gratidão aos meus pais e todos familiares pelo apoio incondicional, compreensão e encorajamento durante toda essa trajetória. Em especial, agradeço minha mãe Marizilda e meu pai José, a maior inspiração de minha vida. Seu amor e suporte foram a minha âncora nos momentos mais desafiadores.

Agradeço a meus amigos por todo apoio que tive durante a graduação, em especial meus amigos de infância Gabriel e William, meus amigos de longa data Gabriela, Isabele e João Octácvio, meus amigos que conheci durante o bacharelado Eduardo, Fernanda, Lara e Luana e meus amigos de trabalho Ana Paula, Camila, Gabriela, Isabela, Lucas, Pamela, Vinícius e Wellington. Obrigado por estarem ao meu lado, compartilhando tanto os momentos de estudo quanto os de descontração, tornando esta jornada ainda mais especial.

Por fim, agradeço à minha dupla, Glaide. Este trabalho não existira sem sua presença e persistência. Uma pessoa ímpar e extraordinária, batalhadora e merecedora que conquistou e ainda conquistará muito durante sua vida, além de, claro, ser uma grande amiga.

Vinícius

RESUMO

Este trabalho aborda a importância da representação visual e interação na educação e como a tecnologia pode auxiliar nesse processo. Serão apresentados recursos digitais disponíveis para tornar a aprendizagem mais interativa e imersiva, além de propor uma abordagem para o ensino de mecanismos de Sistemas Operacionais. O trabalho inclui uma análise comparativa de ferramentas de simulação e visualização e pretende trazer uma ferramenta para educadores e estudantes interessados em aprimorar a compreensão de conceitos complexos por meio de representações visuais e tecnologia.

Palavras-chave: Sistemas operacionais. Interação. Aprendizado. Apoio visual.

ABSTRACT

This work addresses the importance of visual representation and interaction in education, and how technology can assist in this process. Digital resources will be presented to make learning more interactive and immersive, in addition to proposing an approach to teaching Operating Systems mechanisms. The work includes a comparative analysis of simulation and visualization tools and aims to provide a tool for educators and students interested in enhancing the understanding of complex concepts through visual representations and technology.

Keywords: Operating Systems. Interaction. Learning. Visual support.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

CPU	Central Processing Unit
GUI	Graphical User Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SVG	Scalable Vector Graphic
SO	Sistemas Operacionais
STEM	Science, Technology, Engineering and Mathematics
USP	Universidade de São Paulo
SJF	Shortest Job First
RR	Round-Robin
FCFS	First Come, First Served

SUMÁRIO

1 INTRODUÇÃO

A representação visual desempenha um papel importante na educação, auxiliando na compreensão de diversos assuntos e permitindo que alunos e educadores explorem e compreendam conceitos complexos por meio de imagens, gráficos, diagramas e outras formas visuais. Isso estimula a aprendizagem ativa (Brame, 2016), incentivando a participação e o envolvimento dos alunos no processo educacional.

Com o avanço da tecnologia, as representações visuais na educação expandiram-se além das tradicionais ilustrações em papel e desenhos no quadro-negro. Recursos digitais de interação com o usuário como animações, vídeos, realidade virtual e realidade aumentada proporcionam experiências imersivas e interativas, que estimulam a exploração e o pensamento crítico dos alunos.

Assim, a combinação da representação visual com a interação do usuário é uma forma mais eficaz de exemplificar, demonstrar, melhor compreender, identificar padrões e estimular o aprendizado de alunos da área da computação, especialmente considerando que são vistos muitos conceitos abstratos e intangíveis, tornando-se uma ferramenta valiosa para o aprendizado.

Neste contexto, propõe-se uma abordagem para o aprendizado, simulando métodos, algoritmos e conceitos centrados na área de Sistemas Operacionais (SOs) utilizando representações visuais e interação do usuário. O objetivo é criar um conceito de sistema acessível a estudantes, que utilize animações na tela e seja interativo, auxiliando e estimulando os estudantes durante o aprendizado da matéria.

2 FUNDAMENTAÇÃO TEÓRICA

Para produzir um sistema de apoio à educação, é necessário compreender conceitos relacionados sobre como ocorre o processo de aprendizado, como o apoio visual e a interação podem influenciar e instigar alunos na compreensão de novos conteúdos e na representação clara dos diferentes mecanismos presentes em SOs.

2.1 APRENDIZADO, APOIO VISUAL, INTERAÇÃO E SIMULAÇÃO

Este capítulo explora o papel do suporte visual, simulação e aprendizado interativo no campo da educação. Ele enfatiza como esses elementos podem aprimorar significativamente a experiência de aprendizado, especialmente no domínio da educação em ciência da computação. O capítulo aborda os seguintes pontos-chave: apoio visual, aprendizado, interação e simulação.

2.1.1 Apoio visual

O apoio visual refere-se ao uso de elementos visuais para transmitir informações, ideias ou conceitos de forma complementar ou suplementar, podendo variar entre imagens, gráficos, diagramas, desenhos, animações e muito mais. O objetivo é fornecer uma representação visual clara e concisa, que complementa o texto ou a fala, tornando a informação mais atrativa, compreensível e disponível para o público-alvo.

No caso da educação, o apoio visual desempenha um papel importante para enriquecer o processo de aprendizagem e exercendo várias funções, tais como:

- Comunicação efetiva: O uso de imagens e gráficos ajuda a transmitir informações complexas de forma mais acessível e compreensível, permitindo a explicação de conceitos abstratos, a simplificação de ideias complexas e a ilustração de relações entre diferentes elementos (Stokes, 2002; Shabiralyani et al., 2015).

- Reforço da compreensão: Elementos visuais facilitam a conexão de conceitos abstratos a exemplos concretos, permitindo que os alunos façam associações e estabeleçam relações entre diferentes partes do conhecimento.

- Estímulo à memória: Quando explorados adequadamente, os recursos visuais podem ser mais facilmente lembrados e recuperados posteriormente, auxiliando os alunos na retenção de conceitos e detalhes importantes (Raiyn, 2016).

- Engajamento dos alunos: Elementos visuais e interação com os mesmos têm a capacidade de capturar a atenção dos alunos, aumentar o interesse e o envolvimento no processo de aprendizagem, tornando a experiência educacional mais interativa e dinâmica (Grissom et al., 2003).

- Acessibilidade: O apoio visual oferece uma linguagem universal, permitindo que alunos com diferentes origens linguísticas, habilidades cognitivas ou deficiência auditiva se beneficiem do processo de aprendizagem.

Ao buscar informações no Google Scholar, IEEE Xplore, ResearchGate e Semantic Scholar sobre o apoio visual no aprendizado de disciplinas relacionadas à área STEM (acrônimo

em inglês para "*science, technology, engineering and mathematics*"), deparamos com um estudo de Moreno et al. (2011) que se propôs a realizar experimentos para avaliar se alunos do ensino médio e início do ensino superior obtêm melhores resultados ao resolver problemas usando representações concretas e abstratas.

O experimento empregou problemas simples de física elétrica, que envolvem o cálculo da resistência total em circuitos e outras questões correlatas. Os resultados obtidos indicam que o uso de representações visuais, sejam elas abstratas ou concretas, pode levar a uma melhoria substancial na capacidade dos alunos de resolver exercícios e compreender as representações utilizadas (Brame, 2016).

No âmbito da informática, identificamos diversos exemplos de simulação de algoritmos, como vídeos que demonstram algoritmos de ordenação, protocolos de comunicação e outros (Tree, 2020; RA3Player, 2013). Além disso, encontramos ferramentas que não apenas apresentam representações visuais simulando a execução de algoritmos, mas também permitem que o usuário interaja diretamente com o conteúdo apresentado. Exemplos dessas ferramentas incluem o *Scratch* (Scratch, 2018) e sites da web (Halim et al., 2011; Galles, 2011).

2.1.2 Aprendizado

Em um estudo destinado ao ensino de programação para crianças por meio da Realidade Aumentada (Yi-Ming Kao e Ruan, 2022), foi desenvolvido um ambiente interativo que empregasse personagens animados e a construção de código por meio da montagem de "blocos de código"e da definição de ações para esses personagens. Um ponto relevante abordado no estudo é a carga cognitiva (Sweller et al., 2019) imposta ao aluno, que é dividida em três tipos:

- Carga cognitiva intrínseca: relaciona-se com a complexidade do conteúdo ou da atividade. Ao elaborar o currículo e as tarefas, é essencial equilibrar a carga intrínseca, o que pode ser alcançado subdividindo o plano de aprendizagem em "subesquemas"para garantir que essa carga não seja excessiva, pois isso poderia resultar em sobrecarga cognitiva e dificultar a compreensão.

- Carga cognitiva extrínseca: refere-se à carga adicional imposta pelas instruções, organização do material ou pela forma como as tarefas são apresentadas. Diferentes abordagens na apresentação de um mesmo conteúdo podem induzir diferentes níveis de carga cognitiva nos alunos. É necessário fornecer orientações claras e estruturas bem definidas para reduzir a carga extrínseca e permitir que os alunos se concentrem no conteúdo essencial.

- Carga cognitiva pertinente: diz respeito à carga que favorece a aprendizagem significativa, envolvendo o processamento profundo e a elaboração das informações. A distribuição adequada da carga cognitiva pertinente incentiva os alunos a estabelecer conexões entre novos conhecimentos e o conhecimento prévio, a fazer associações e a aplicar o que estão aprendendo em contextos relevantes, fortalecendo a compreensão e a retenção a longo prazo.

Se o material de aprendizagem não for devidamente projetado, pode ocorrer o uso excessivo dos recursos da memória de trabalho do aluno, resultando em uma carga cognitiva extrínseca elevada. Por outro lado, uma instrução bem elaborada tem a capacidade de direcionar a atenção dos alunos para o processo cognitivo de aprendizado de conteúdos relacionados à construção de esquemas. Isso ilustra a necessidade de desenvolver um sistema que equilibre cada tipo de carga cognitiva, permitindo aos alunos manter o foco durante o aprendizado e armazenar as informações apresentadas na memória de longo prazo.

2.1.3 Interação e simulação

Sistemas e várias formas que oferecem representações visuais e simulações passo a passo de algoritmos e estruturas de dados são amplamente reconhecidos, como exemplificado nos casos mencionados na seção 2.1.1. Entretanto, apenas uma minoria deles proporciona aos usuários a oportunidade de interagir ativamente e obter *feedback* sobre as informações que inserem.

Neste contexto, apresentamos três exemplos de sistemas que têm como objetivo elucidar o funcionamento de um determinado conteúdo na área de computação, seja um algoritmo, uma estrutura de dados ou qualquer outro conceito. Esses sistemas utilizam representações visuais para demonstrar o funcionamento e, ao mesmo tempo, permitem que o usuário interaja com o sistema, podendo ajustar parâmetros, escolher ações que o sistema deve executar ou realizar qualquer outra ação que influencie o conteúdo exibido.

O primeiro deles é o site Data Structure Visualizations, desenvolvido pelo professor David Galles da Universidade de São Francisco, Califórnia (Galles, 2011). O site apresenta animações interativas de dezenas de algoritmos e estruturas de dados vistos durante uma graduação em Ciência da Computação (como grafos, algoritmos recursivos e algoritmos de ordenação) com entradas e botões de ação a se realizar no algoritmo sendo visto (como alterar, inserir e remover elementos), além de controles para a animação. Porém, pode-se destacar como ponto negativo que não é feita uma descrição de como estes algoritmos funcionam ou então apresentados exemplos de implementação de seu código, apesar de que as animações possam ser auto-explicativas.

Desenvolvido com JavaScript e o elemento canvas presente no HTML5, o site foi pensado de forma que permita que suas animações possam ser vistas nos mais diversos navegadores e dispositivos modernos, fazendo com que seja acessível digitalmente, de forma a eliminar barreiras na disponibilidade de conteúdo. O autor também disponibiliza um tutorial e biblioteca para que visitantes possam criar suas próprias representações visuais. Um exemplo de representação de algoritmos é disponibilizado no site e pode ser visto na figura 2.1. Utilizando o algoritmo de árvore binária de busca, temos destacados em vermelho controles do algoritmo - que neste caso permitem inserir, deletar, buscar e imprimir valores na árvore - e em azul controles da animação

Figura 2.1: Um exemplo de visualização do algoritmo de árvore binária de busca disponibilizado no site do Data Structure Visualizations. Estão destacados os controles do algoritmo simulado e os de animação em vermelho e azul, respectivamente.

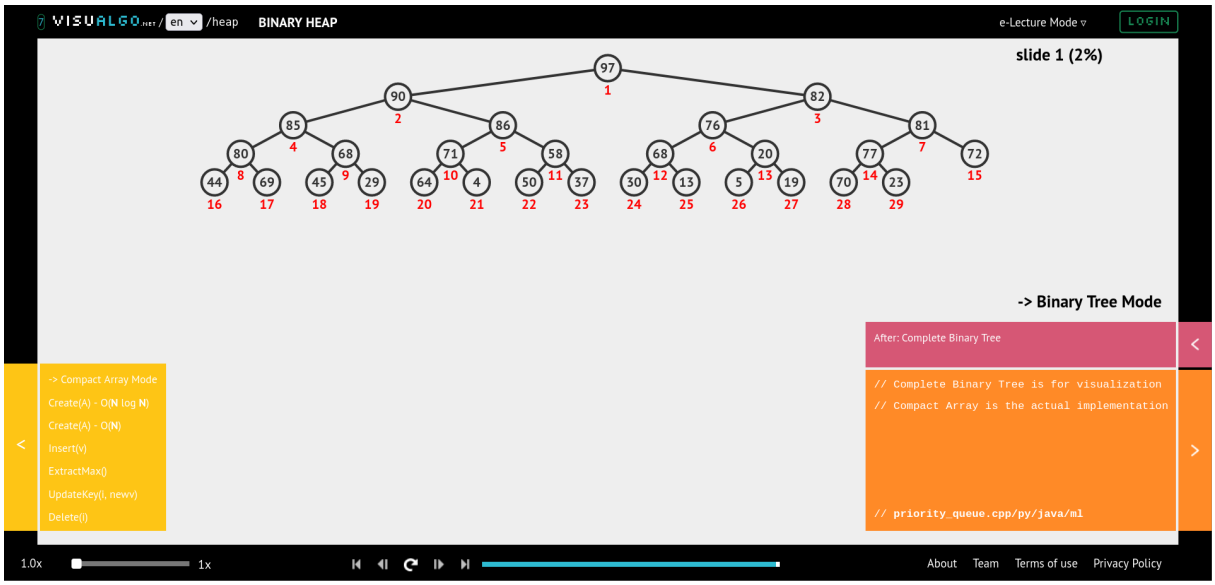


Figura 2.2: Um exemplo da representação do algoritmo de Heap Binária no VisuAlgo.

Figura 2.3: Trabalho por alunos da USP exemplificando o sistema produtor/consumidor (de França Cabrini et al., 2015).

do algoritmo - pausar, pular, voltar e avançar a animação, alterar a velocidade de animação e tamanho do canvas localizado no centro.

Feito na Universidade Nacional de Singapura, tem-se também o VisuAlgo (Halim et al., 2011). Semelhante ao Data Structure Visualizations, este site apresenta animações interativas de vários de algoritmos e estruturas de dados e permite que seus usuários tenham controle em suas animações. Em adição a isso, ao abrir a página do conteúdo desejado são apresentados conceitos e definições sobre o algoritmo/estrutura de dados escolhido e possui um menu à direita que permite exibir um exemplo de código que conforme é executada a animação, são destacadas as linhas de código executadas. VisuAlgo também possui uma interface mais moderna, colorida e subjetivamente mais atrativa de forma que também mantém compatibilidade com dispositivos móveis. Atualmente a página possui representações visuais para 24 algoritmos. Na figura 2.2 encontramos uma captura de tela do algoritmo de Heap Binária em que no centro da tela a animação do algoritmo com as funções do algoritmo que podem ser simuladas no canto inferior esquerdo (na divisória em amarelo) que, ao serem selecionadas, adicionam um exemplo de

código desta função e comentários de sua execução à direita (nas divisórias laranja e vermelha, respectivamente). Opções de controle da animação também são encontradas na parte inferior da tela.

Finalmente, foi identificado um diretório contendo diversos projetos realizados por alunos do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) durante a disciplina de Sistemas Operacionais, lecionada em 2015 pelo professor Paulo Sérgio Lopes de Souza[1]. Esses projetos apresentam diferentes representações visuais para algoritmos e mecanismos utilizados por sistemas operacionais.

Similar aos exemplos mencionados anteriormente, a maioria desses projetos está disponível online na forma de sites. Além disso, esses trabalhos adotam diversos métodos para transmitir informações aos alunos: alguns se baseiam exclusivamente em animações com etapas explicadas em "passo a passo"(B. Andrião et al., 2015), outros incorporam animações que permitem a entrada de informações pelos usuários (de França Cabrini et al., 2015), e alguns recorrem a jogos para ilustrar problemas que são resolvidos em sistemas operacionais (Scalet Bicalho et al., 2015). Na figura 2.3 temos um exemplo de algoritmo do sistema produtor/consumidor com código e botão de execução de cada um de seus tipos de processo. A cada clique no botão é simulada a execução de uma linha de seu código acima e clicando no botão "Desfazer"o último código simulado retorna ao estado de sua linha anterior e para demonstrar um *buffer* sendo ocupado foi utilizada a figura de uma pilha que eventualmente é preenchida.

2.2 DIFICULDADE DE APRENDIZADO NA COMPUTAÇÃO

A origem de erros durante o aprendizado na área de computação pode surgir por diferentes motivos e variar em cada caso individual, mas duas das origens mais comumente citadas (e fortemente relacionadas entre si) em estudos com alunos (Karavirta et al., 2013; Seppälä et al., 2006) são a Teoria de Reparação (*Repair Theory*) e a Solução Imitativa de Problemas (*Imitative Problem Solving*).

A Teoria de Reparação sugere que estudantes que já aprenderam uma atividade procedural inicial, ao aprenderem uma nova atividade, acabem "reparando"(consertando) seus erros fazendo uma correlação não existente entre estas atividades distintas por conta de viés presente em exemplos, material didático ou informações passadas por instrutores. Esta teoria foi inicialmente descrita por Brown e VanLehn (1980), em um estudo no qual os autores buscavam uma teoria generativa para erros (*bugs*) em *skills* procedurais para predizer quais *bugs* sistemáticos ocorrerão no comportamento dos alunos que estão aprendendo dada habilidade. Infelizmente não foi possível expressar de forma generalizada todos os erros cometidos por alunos, já que alunos alternam seus erros cometidos em um pequeno espaço de tempo, um fenômeno chamado pelos autores de "migração de erros"(*bug migration*).

Já a Solução Imitativa de Problemas (Robertson e Kahney, 1996; Robertson, 2000) afirma que iniciantes usam da imitação como seu principal método de resolução de problemas ao aprender sobre um novo domínio desconhecido. A resolução de problemas por imitação pode explicar grande parte da evidência de que a transferência análoga, mesmo dentro de um domínio, costuma ser difícil de ocorrer.

Um estudo mais recente sobre como mudanças na Interface de Usuário podem afetar a performance de estudantes (Karavirta et al., 2013) também corrobora o papel da Teoria de Reparação e Solução Imitativa de Problemas. Buscando saber se alunos preferem o uso de uma UI mais simples e interação e se cometem os mesmos equívocos descritos em estudos anteriormente realizados em 2006 (Seppälä et al., 2006), foram repetidos os estudos de 2006 e acrescidas novas

[1] http://www.lasdpc.icmc.usp.br/ ssc640/grad/

tarefas para mais de 100 alunos de Ciência da Computação no curso de Algoritmos e Estruturas de Dados realizarem. A comparação entre interfaces foi feita entre os exercícios dos estudos anteriores que utilizavam o TRAKLA2 (Korhonen et al., 2009), enquanto os exercícios novos utilizavam o JSAV, um novo sistema feito com JavaScript e HTML mais pensado em dispositivos com telas *touch*. Ao final, os alunos apresentaram erros semelhantes em ambas as pesquisas – que corrobora o papel da Teoria de Reparação e Solução Imitativa de Problemas – e apesar de não ter uma clara influência nos resultados, a mudança da interface rendeu elogios por parte dos alunos.

2.3 SISTEMAS OPERACIONAIS

Para dar inicio à explicação sobre o que é um sistema operacional, é importante entender que ele desempenha um papel fundamental em qualquer dispositivo computacional, seja um computador pessoal, um *smartphone*, um *tablet* ou até mesmo um sistema embarcado, como um caixa eletrônico ou uma máquina de lavar roupas inteligente. Um sistema operacional é um software complexo que atua como uma camada intermediária entre os aplicativos e o hardware de um dispositivo.

Em termos simples, pode-se afirmar que um sistema operacional é o responsável por gerenciar e coordenar todas as atividades do computador. Ele fornece uma interface entre o usuário e o hardware, como visto na figura 2.4, permitindo que os programas sejam executados de maneira eficiente e controlando os recursos disponíveis.

Um sistema operacional possui várias funções essenciais. Segundo Maziero (2020), suas principais funcionalidades são:

- Gerência do processador, que visa distribuir a capacidade de processamento.
- Gerência da memória, que controla como os programas e os dados são armazenados na memória principal.
- Gerência de dispositivos, que busca implementar a integração de cada dispositivo por meio de *drivers*.
- Gerência de arquivos, que é responsável por controlar como os arquivos são armazenados, organizados e acessados.
- Gerência de proteção, que é responsável por garantir a segurança e proteção dos recursos do sistema contra ameaças e acessos não autorizados.

Um sistema operacional também oferece serviços de rede, permitindo que os dispositivos se conectem e troquem informações em uma rede local ou na internet. Ele desempenha um papel crítico na segurança do sistema, protegendo-o contra ameaças externas e internas, como vírus e acessos não autorizados.

Existem diferentes tipos de sistemas operacionais, incluindo Windows, macOS, Linux, Android, iOS, entre outros. Cada um possui suas características específicas e é projetado para atender às necessidades dos diferentes dispositivos e usuários.

Em resumo, um sistema operacional é um software complexo que atua como o cérebro de um dispositivo computacional, coordenando todas as suas operações e fornecendo uma interface para que os usuários interajam com ele. Ele desempenha um papel crucial na execução eficiente de programas, gerenciamento de recursos e garantia da segurança e estabilidade do sistema.

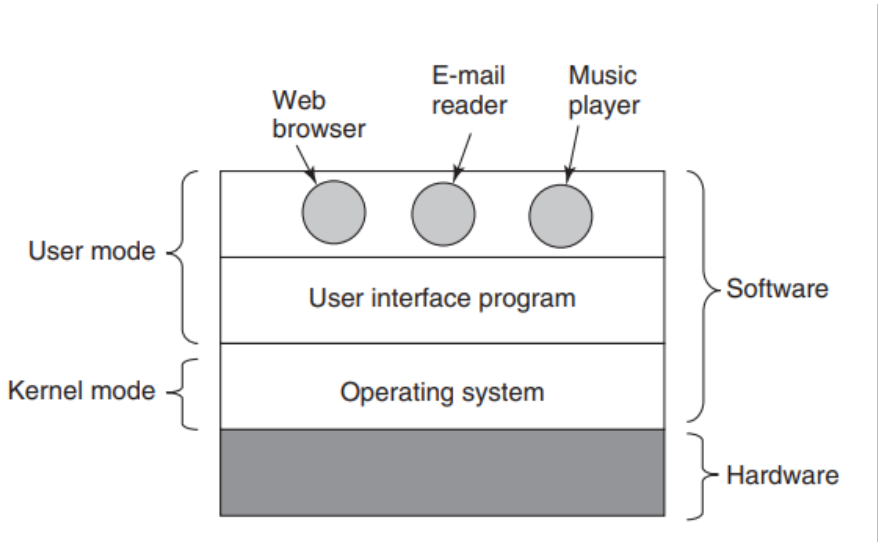


Figura 2.4: Sistema operacional (Tanenbaum e Bos, 2014)

2.3.1 Mecanismos de sistemas operacionais

Para uma melhor compreensão do propósito deste trabalho, é importante ter um entendimento mais aprofundado de alguns conceitos de sistemas operacionais, tais como tarefas, *threads* e concorrência. Ao adquirir esse conhecimento, torna-se possível compreender o nível de complexidade envolvido e por que os alunos de graduação frequentemente enfrentam dificuldades ao aprender esse tema.

Concorrência é um conceito importante na ciência da computação que se refere à execução simultânea de várias partes de um programa. Isso significa que diferentes partes do programa podem ser executadas ao mesmo tempo, em paralelo, em vez de serem executadas em uma sequência linear. A concorrência é usada para melhorar o desempenho dos programas, permitindo que várias tarefas sejam executadas ao mesmo tempo, em vez de esperar que uma tarefa seja concluída antes de iniciar outra.

No entanto, o uso de concorrência também pode levar a problemas, como condições de corrida e impasses, que ocorrem quando duas ou mais partes do programa tentam acessar o mesmo recurso ao mesmo tempo, o que pode levar a resultados imprevisíveis ou a um bloqueio completo do programa. Para evitar esses problemas, é necessário sincronizar o acesso aos recursos compartilhados, garantindo que apenas uma parte do programa possa acessá-los de cada vez. A sincronização pode ser realizada por meio de técnicas como exclusão mútua, semáforos e monitores que garantem que a seção critica (trecho de código que podem ocasionar condições de corrida) será acessada apenas por quem está no "direito"de acessar.

De acordo com Strömbäck et al. (2018), a concorrência é frequentemente percebida como um assunto difícil pelos estudantes de ciência da computação. Isso ocorre porque a concorrência envolve a execução simultânea de várias partes de um programa, o que pode ser difícil de entender e visualizar. Além disso, como já foi dito, a concorrência pode levar a problemas como condições de corrida e impasses, que podem ser difíceis de detectar e corrigir, principalmente considerando alunos que estão em um momento de aprendizagem e entendimento do assunto. Outro desafio é que a concorrência é uma área abstrata e teórica, o que pode tornar difícil para os estudantes entenderem como ela se aplica na prática. Para superar esses desafios, os autores sugerem que os professores devem abordar a concorrência por meio de uma combinação de ensino teórico e prático, incluindo a realização de exercícios práticos e a discussão de exemplos concretos de programas concorrentes.

De acordo com Netzer e Miller (1992), uma condição de corrida (ou "*race condition*", em inglês) é um tipo de "bug"de software que ocorre quando duas ou mais *threads* ou processos acessam dados ou recursos compartilhados em uma ordem inesperada, resultando em um comportamento imprevisível ou resultados incorretos. No contexto de programas paralelos com memória compartilhada, as condições de corrida podem ocorrer quando várias *threads* tentam acessar e modificar as mesmas variáveis compartilhadas simultaneamente, sem uma sincronização adequada. De acordo com Flanagan et al. (2005) , "uma condição de corrida ocorre quando duas *threads* manipulam uma estrutura de dados compartilhada simultaneamente, sem sincronização. Condições de corrida são erros comuns em programas *multithreaded* e, como são dependentes de tempo, são notoriamente difíceis de detectar usando testes."

Um exemplo comum de condição de corrida ocorre quando duas *threads* tentam atualizar simultaneamente o mesmo valor de uma variável compartilhada. Caso elas não estejam sincronizadas corretamente, o resultado final pode ser imprevisível, dependendo da ordem em que as operações são executadas. Por exemplo, se a primeira *thread* ler o valor da variável, adicionar 1 e escrever o resultado de volta, enquanto a segunda *thread* fizer o mesmo, o valor final da variável pode ser 1 ou 2, dependendo da ordem em que as operações forem executadas. Esse comportamento imprevisível pode levar a problemas difíceis de detectar e corrigir.

2.4 CONSIDERAÇÕES

Em suma, a criação de um sistema de apoio à educação com suporte visual deve levar em consideração diversos fatores para garantir eficácia e engajamento. Em primeiro lugar, a adaptação aos diferentes estilos de aprendizagem, oferecendo variedade de formatos visuais e interativos para atender às preferências dos alunos. A usabilidade é um fator a ser priorizado, assegurando que o sistema seja intuitivo e o mais acessível possível a todos os usuários - especialmente se tratando de conceitos abstratos. A criação de conteúdo deve ser pedagogicamente sólida, alinhada aos objetivos educacionais, e as informações visuais devem ser claras e informativas. A interatividade deve ser bem projetada, promovendo a participação ativa dos alunos e proporcionando *feedback* imediato para reforçar a compreensão.

Por fim, é importante envolver educadores e alunos no processo de desenvolvimento, coletando *feedback* contínuo para aprimorar o sistema e garantir que atenda às necessidades específicas do ambiente educacional, além de ser importante para identificar preferências e possíveis desafios de usabilidade que podem não ser óbvios para os desenvolvedores.

3 ESTADO DA ARTE

Com base em nossas pesquisas, simulação de sistemas operacionais têm se mostrado promissor no campo da educação, onde a tecnologia da informação desempenha um papel fundamental. Neste capítulo serão abordados trabalhos da área de sistemas operacionais no campo educacional.

3.1 TRABALHOS ANTERIORES

Estudos recentes na área de aprendizado online e educação em programação têm explorado a eficácia do conteúdo instrucional e o uso de representações visuais para aprimorar as experiências de aprendizado.

Em um estudo realizado por Nariman (2021) sobre a efetividade do conteúdo de aprendizado online, os resultados indicam que a maioria dos estudantes expressou satisfação com vários aspectos do sistema de e-learning. Os níveis mais altos de satisfação estavam associados ao design sistemático de conteúdo em vídeo, à utilidade dos vídeos, a livros digitais úteis, à clara definição de objetivos nos livros digitais e ao uso apropriado de cores nos livros digitais. No entanto, o estudo não forneceu percepções específicas sobre o impacto de instruções interativas no e-learning, especialmente para cursos de programação.

Outro estudo realizado Hundhausen e Douglas (2000), centrado em metodologias de aprendizado, comparou o uso de representações visuais de algoritmos no ensino de programação. Os autores conduziram um experimento com estudantes de ciência da computação, onde um grupo utilizou software de visualização de algoritmos (AV), enquanto outro grupo criou suas próprias visualizações. Ambos os grupos aprenderam o algoritmo QuickSelect e foram avaliados em suas habilidades de acompanhamento e desenvolvimento de programas. Os resultados mostraram diferenças mínimas entre os grupos, mas os autores sugerem que estudos adicionais podem revelar resultados aprimorados para o grupo que criou suas próprias representações.

3.1.1 Trabalhos com ferramentas implementadas

A evolução da educação em sistemas operacionais tem testemunhado avanços significativos através de ferramentas inovadoras recentemente introduzidas. Esta revisão examinará mais detalhadamente essas abordagens e seu potencial impacto na educação em sistemas operacionais.

3.1.1.1 Visual OS

Um outro estudo recente de Hill e Gokhale (2005) abordou especificamente o uso da tecnologia da informação para o ensino de sistemas operacionais, com ênfase nos processos de sincronização. Os autores desse estudo propuseram uma ferramenta interativa inovadora chamada Visual OS, que proporciona a visualização de diversos algoritmos relacionados ao tema, sua arquitetura foi desenvolvida conforme mostra a figura 3.1. Essa ferramenta permite ao usuário selecionar o algoritmo desejado, bem como inserir e controlar as entradas, a figura 3.2 ilustra a ação de criar um novo projeto.

Para avaliar a eficácia da ferramenta Visual OS, um experimento foi conduzido em uma universidade, no qual os alunos participantes foram divididos em dois grupos distintos. O primeiro grupo teve acesso à ferramenta durante o processo de aprendizagem, enquanto o segundo grupo seguiu o método tradicional de ensino. Os resultados foram analisados com base nos

índices de compreensão alcançados por cada grupo. Os resultados desse experimento revelaram que os alunos que utilizaram a ferramenta Visual OS apresentaram índices significativamente maiores de compreensão em relação aos que não tiveram acesso a ela. A possibilidade de interagir com os algoritmos e visualizar seu funcionamento em tempo real proporcionou aos alunos uma compreensão mais aprofundada dos processos de sincronização em sistemas operacionais.

Esta pesquisa representa um avanço no campo da educação em sistemas operacionais, demonstrando como o uso da tecnologia da informação pode contribuir para um aprendizado mais eficiente e eficaz. Visual OS oferece uma abordagem interativa que permite aos alunos explorar e experimentar conceitos complexos de forma prática e visualmente estimulante.

Considerando o impacto positivo observado no experimento realizado em uma universidade, esta ferramenta pode ser uma adição valiosa ao currículo de cursos relacionados a sistemas operacionais. Além disso, essa abordagem pode ser estendida para outras áreas de estudo, proporcionando um ambiente de aprendizado mais dinâmico e efetivo para os estudantes.

No entanto, é importante ressaltar que essa pesquisa representa apenas o início de um campo promissor. Há potencial para aprimoramentos futuros do Visual OS, bem como para a exploração de outros aspectos relevantes da simulação de sistemas operacionais. A contínua colaboração entre educadores, pesquisadores e profissionais de tecnologia da informação é fundamental para impulsionar ainda mais o estado da arte nesse campo e maximizar seu impacto na educação.

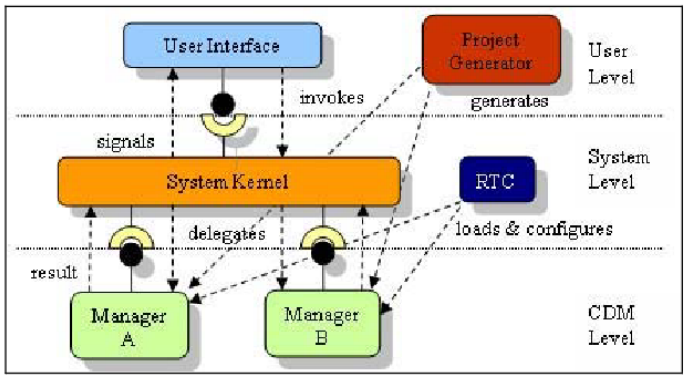


Figura 3.1: Visual OS: diagrama de três níveis da arquitetura em sua forma montada. (Hill e Gokhale, 2005)

3.1.1.2 Design Modular Visual de Sistemas Operacionais

Em seu artigo intitulado "Um *Framework* para Design Modular Visual de Sistemas Operacionais Educacionais", Al-Oudat (2017) propõe uma nova direção promissora nesse campo, introduzindo um método totalmente baseado na construção visual de sistemas operacionais educacionais. Um *framework* é uma ferramenta ou conjunto de ferramentas que fornece uma base para o desenvolvimento de software. Ele oferece uma estrutura organizada para a construção de aplicativos e simplifica tarefas comuns, com uma estrutura pré-definida, abstração de tarefas comuns, manipulação de dados, entre outros.

Projetado para facilitar o aprendizado de sistemas operacionais, esta ferramenta permite que os usuários construam seus próprios sistemas através de uma interface visual. A plataforma de desenvolvimento consiste em blocos-chave que podem ser arrastados e soltos em um painel de trabalho, proporcionando aos usuários uma maneira intuitiva de projetar e montar seus sistemas operacionais personalizados.

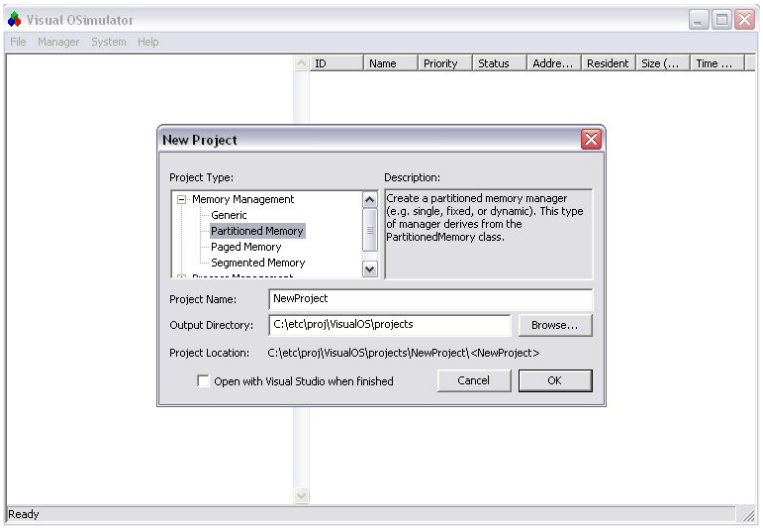


Figura 3.2: Visual OS: criação de um novo projeto. (Hill e Gokhale, 2005)

Uma das principais vantagens é a mitigação da complexidade associada à programação de sistemas operacionais tradicionais. Em vez de entrar em detalhes intricados de codificação e programação, os usuários podem se concentrar no design e na estruturação de seus sistemas, selecionando e ajustando os blocos de acordo com suas necessidades específicas.

Além disso, este *framework* apresenta-se como uma valiosa ferramenta educacional para o ensino de sistemas operacionais. A programação visual proporciona uma abordagem envolvente para os alunos, permitindo-lhes compreender e experimentar diretamente os princípios subjacentes aos sistemas operacionais. Ao interagir com os blocos modulares e visualizar as alterações em tempo real, os estudantes podem aprofundar seu conhecimento teórico enquanto ganham experiência prática na construção de sistemas operacionais.

Essa abordagem visual também pode ser benéfica para estudantes com diferentes estilos de aprendizagem, tornando o ensino de sistemas operacionais mais inclusivo e adaptável. Ao remover algumas das barreiras tradicionais associadas à programação de baixo nível, o trabalho de Al-Oudat amplia o acesso e o envolvimento de estudantes com diversos níveis de habilidade técnica.

Em resumo, Al-Oudat representa uma contribuição significativa para o estado da arte da simulação de sistemas operacionais. Sua abordagem baseada na construção visual oferece uma maneira mais intuitiva e envolvente de aprender e desenvolver sistemas operacionais. Ao mitigar a complexidade da programação de baixo nível, esse *framework* se destaca como uma ferramenta educacional promissora para o ensino de sistemas operacionais, capacitando os alunos a explorar conceitos fundamentais por meio da programação visual.

3.1.1.3 SimulaRSO

Pacheco e Costa, em seu trabalho de conclusão de curso, desenvolveram o simulador SimulaRSO (Pacheco e Costa, 2011), um projeto que tem como objetivo simular o comportamento dos principais algoritmos de escalonamento de processos, requisições de disco e paginação de memória virtual em sistemas operacionais. O projeto foi desenvolvido como uma ferramenta online de ensino que auxilia o aluno durante as aulas de Sistemas Operacionais.

O simulador oferece animações gráficas que representam os recursos do sistema operacional, como processos, memória, disco e CPU, e apresenta informações relevantes para estudos analíticos, como o tempo de espera, tempo de execução e tempo de resposta dos

processos. Além disso, o simulador permite que o usuário altere os parâmetros dos algoritmos de escalonamento e paginação, para que possa observar o comportamento do sistema operacional em diferentes cenários.

O projeto foi implementado utilizando alguns Padrões de Projetos na arquitetura do sistema, para permitir a modularização do projeto e facilitar no trabalho de futuras extensões e ou manutenções neste simulador. O simulador pode ser facilmente integrado a qualquer sistema EAD (Ensino a Distância), pelo fato da aplicação ser totalmente acessada pela internet. Na figura 3.3 pode-se visualizar a tela de simulação de escalonamento de processos, demonstrando o tempo de processamento, e exemplificando com uma simulação gráfica do algoritmo Round-Robin.

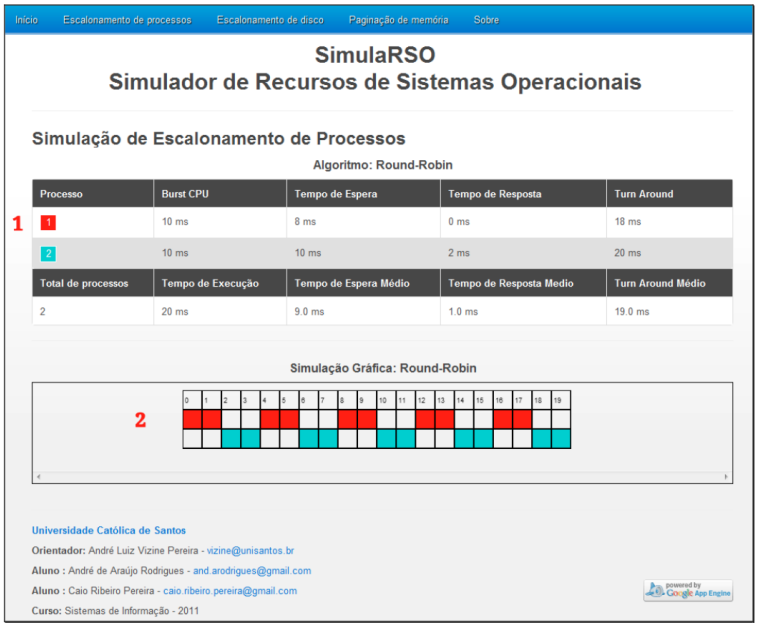


Figura 3.3: SimulaRSO: simulação de escalonamento de processos. (Pacheco e Costa, 2011)

3.1.1.4 WebVizOs

Outro projeto de Pham (2022) é o WebVizOs. WebVizOS é um *framework* web para simulações interativas e visualizações dos principais conceitos e mecanismos do sistema operacional. A arquitetura do sistema consiste em vários componentes, incluindo o simulador principal do sistema operacional, componentes de suporte, como *Graphical User Interface* (GUI), gerenciamento de usuários e submissões, e um banco de dados para armazenar dados de entrada e saída.

Os usuários podem analisar ou aprender um mecanismo em um dos principais componentes do sistema operacional, como gerenciamento de processos, gerenciamento de memória e gerenciamento de E/S, usando códigos pré-construídos do sistema operacional e os dados de entrada padrão para valores de parâmetros que são pré-armazenados ou gerados sob demanda no sistema. Os usuários também podem testar seus próprios códigos do sistema operacional para o mecanismo do sistema operacional fornecido e monitorar seu desempenho, enviando os dados de entrada e saída para o simulador do sistema operacional por meio da GUI baseada na web.

O WebVizOS é projetado para ser aberto e portátil, o que significa que os códigos que rodam em diferentes plataformas podem se conectar a ele. Ele também é interativo, permitindo que os usuários tenham entradas instantâneas no processo de simulação, pausar ou alterar a velocidade da simulação para acompanhar o progresso das atividades do sistema operacional. Em

geral, este é um *framework* abrangente e interativo de simulação e visualização para componentes do sistema operacional que pode ser usado para desenvolvimento, pesquisa e otimização de mecanismos individuais do sistema operacional. A figura 3.4 descreve de forma visual as divisão estrutural do WebVizOS, e na figura 3.5 é possível visualizar a página web do sistema, com algumas configurações de escalonamento de CPU.

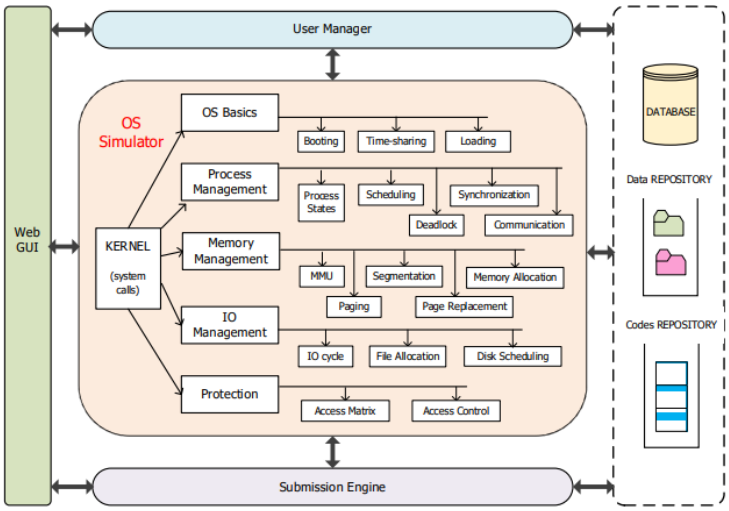


Figura 3.4: Estrutura do WebVizOS (Pham, 2022)

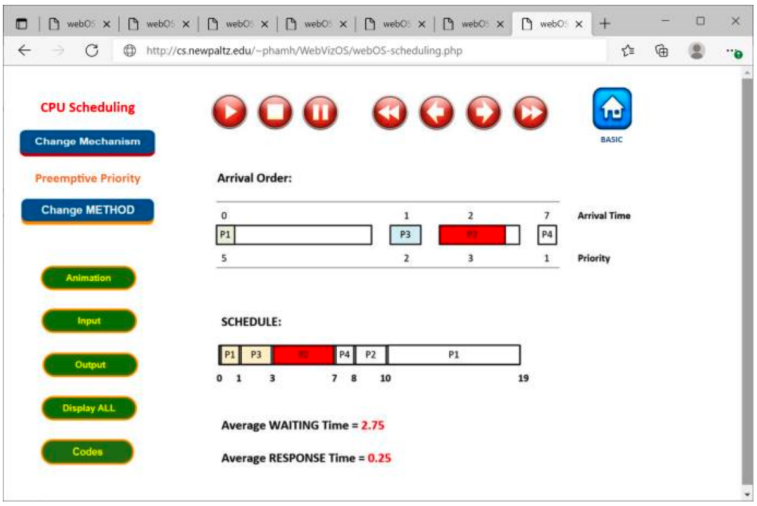


Figura 3.5: WebVizOS: escalonamento de CPU (Pham, 2022)

3.2 ANÁLISE COMPARATIVA

A literatura acadêmica reflete uma crescente preocupação em aprimorar os métodos de ensino. Todos os artigos analisados exploraram abordagens que tinham como objetivo facilitar a compreensão do conteúdo, buscando avaliar se as metodologias adotadas resultaram em uma melhor assimilação do assunto estudado. No entanto, ao direcionarmos nossa atenção para o uso de ferramentas desenvolvidas especificamente para servir como meio de interação em um sistema operacional e outras atividades relacionadas ao tema, pudemos encontrar ferramentas com intuitos diferentes. Podemos resumir as principais ferramentas da seguinte forma:

Visual OS:

Em suma, o usuário pode selecionar o algoritmo desejado como entrada e arrastar para iniciar o funcionamento, observando suas interações. O Visual OS busca simular problemas de sincronização:

- Sistema produtor/consumidor, utilizando *buffer* com e sem limites.
- O problema leitor/escritor, com prioridade de leitura, prioridade de escrita e o modelo clássico.
- O Jantar dos Filósofos.

Comparado aos outros sistemas, oferece uma interação mais limitada, em que o usuário pode somente arrastar o algoritmo selecionado para iniciar sua execução e observar as interações ocorrendo.

WebVizOS:

Recebe algoritmos já pré-construídos como entrada, juntamente com parâmetros, incluindo tabelas e listas de dados necessários para o algoritmo. O usuário pode escolher o tipo de sistema operacional que deseja utilizar, inserir o algoritmo que deseja visualizar e fornecer os dados de entrada e saída esperados para o algoritmo. Além disso, é possível interagir com suas animações, podendo pausá-las e retomá-las, sendo um dos mecanismos de interação mais completo entre os três.

SimulaRSO:

Recebe como entrada informações como o número de processos, o tamanho da memória, a capacidade do disco, as políticas de escalonamento, entre outros. Nele são simulados os seguintes mecanismos:

- Algoritmos de escalonamento de processos.
- Requisições de disco.
- Paginação de memória virtual.

Enquanto Visual OS e WebVizOS são duas abordagens diferentes para visualizar algoritmos e problemas de sincronização, o SimulaRSO possui um foco voltado para escalonamento de processos, escalonamento de disco e algoritmos de paginação de memória virtual. Oferece uma interação simples e completa, que tenta ser amigável ao usuário e também projetado para ser acessado pela internet, não exigindo que o usuário faça download e instalação local para utilizá-lo. O intuito da ferramente é de, através de simulações interativas e animações gráficas, tornar o aprendizado mais interativo e facilitar a compreensão dos conceitos teóricos apresentados em sala de aula. A tabela 3.1 a seguir apresenta de forma mais detalhada a comparação entre Visual OS, SimulaRSO e WebVizOS.

Aspecto	Área de Foco	Principais Resultados	Resultados Específicos	Inovação ou Ferramenta Introduzida	Impacto Educacional	Aplicabilidade a Outras Áreas	Desafios e Oportunidades de Melhoria
Visual OS	Tecnologia no ensino de SOs	A ferramenta ajudou na compreensão do tema	Usuários tiveram compreensão significativamente maior	Ferramenta visual para SOs	Melhoria na compreensão do aprendizado de SOs	Aplicável ao ensino de outros conceitos de SOs	Melhorias futuras e exploração de aspectos relevantes da simulação de SOs
SimulaRSO	Simulador para escalonamento de processos, requisições de disco e paginação de memória	Testes aplicados com alunos, porém sem resultados conclusivos	Oferece animações gráficas e informações analíticas	Simulador para ensino de SOs	Facilitar o ensino de conceitos de SOs	Aplicável a vários conceitos de SOs	Melhorias futuras e exploração na simulação de SOs
WebVizOs	Framework web para simulações interativas e visualizações de SOs	Suporta simulações interativas e visualizações	Permite análise, aprendizado e teste de mecanismos de SOs	Framework para simulações e visualizações interativas de SOs	Suporta desenvolvimento, pesquisa e otimização de mecanismos de SOs	Adequado para desenvolvimento, pesquisa e otimização de mecanismos de SOs	Abertura e portabilidade entre diferentes plataformas

Tabela 3.1: Tabela comparativa dos principais aspectos das obras citadas

3.2.1 Considerações finais

Em resumo, o Visual OS é voltado para a visualização de problemas de sincronização específicos, enquanto o WebVizOS é mais abrangente, permitindo a visualização de vários algoritmos e fornecendo a capacidade de interagir com as animações. Em conclusão, os três tópicos - Visual OS, WebVizOS e SimulaRSO - compartilham o objetivo comum de fornecer visualizações interativas que facilitem a compreensão e o aprendizado de conceitos relacionados a sistemas operacionais e problemas de sincronização.

Esses trabalhos abordam um desafio significativo no ensino de sistemas operacionais, que é a complexidade desses conceitos e a dificuldade em visualizá-los de forma interativa. É crucial reconhecer a importância de utilizar ferramentas acessíveis e de fácil compreensão para os alunos, a fim de promover uma aprendizagem efetiva.

No âmbito geral, essas ferramentas desempenham um papel fundamental no aprendizado de sistemas operacionais e problemas de sincronização, oferecendo diferentes níveis de interatividade e visualização. Elas fornecem uma abordagem prática que auxilia os estudantes no desenvolvimento de uma compreensão mais profunda desses conceitos complexos. Ao permitir que os alunos observem as interações entre processos, experimentem algoritmos e manipulem as animações, essas ferramentas tornam o aprendizado mais envolvente e atrativo.

Portanto, ao integrar essas visualizações interativas em atividades educacionais, os alunos têm a oportunidade de aprimorar seu conhecimento teórico, enquanto experimentam na prática os princípios subjacentes aos sistemas operacionais e problemas de sincronização. Isso contribui para uma compreensão mais abrangente e uma aplicação mais eficaz desses conceitos no mundo real.

4 PROPOSTA

Como discutido no capítulo 2.2, é comum que alunos de graduação enfrentem dificuldades ao tentar compreender conceitos mais complexos da área de computação, podendo confundir novos conceitos com antigos ou até fazer correlações com conceitos não correlatos, especialmente quando se trata de sistemas operacionais. Além disso, como mencionado no capítulo 2.1.1, de fato, o apoio visual desempenha um papel importante na educação, contribuindo significativamente para que os alunos possam compreender melhor o funcionamento do assunto sendo estudado. Através de recursos visuais, é possível enriquecer o processo de aprendizagem, proporcionando uma representação mais clara e tangível de conceitos abstratos. Isso permite que alunos visualizem as informações de forma mais concreta e estabeleçam conexões mais sólidas entre os conceitos estudados, facilitando assim a compreensão e retenção do conhecimento.

4.1 MOTIVAÇÃO

Sistemas como VisuAlgo (Halim et al., 2011) e Data Structure Visualizations (Galles, 2011) possuem uma construção sólida, com representação visual, interatividade e um sistema de simulação bem elaborados. Apesar disso, estes sites tratam de algoritmos e estruturas de dados mais generalistas, enquanto neste trabalho buscamos tratar mais especificamente de mecanismos de SOs. Por outro lado, apesar de boa parte dos sites desenvolvidos pelos alunos da USP (B. Andrião et al., 2015; de França Cabrini et al., 2015; Scalet Bicalho et al., 2015) tratarem de de algoritmos de sistemas operacionais, nem todos possuem representação visual, interatividade e simulação bem elaborados. E, por fim, devemos ponderar que não foi possível acessar os simuladores vistos no capítulo 3 (Pham, 2022; Pacheco e Costa, 2011) e somente considerar as publicações feitas por seus autores.

Diante disso, o objetivo deste trabalho é desenvolver uma ferramenta capaz de auxiliar estudantes e professores durante o ensino de Sistemas Operacionais, buscando tornar o processo de aprendizagem mais simples e intuitivo. A intenção é de auxiliar na compreensão e tornar estes tópicos de mais fácil dispor aos alunos, através de uma ferramenta que permita aos alunos interagir ativamente, de modo a compreender o passo a passo dos algoritmos e mecanismos implementados.

4.2 ALGORITMOS E MECANISMOS

A seguir, separamos alguns dos mecanismos de SOs que consideramos representar durante o desenvolvimento de nosso trabalho:

4.2.1 Concorrência e impasses

Como discutido no capítulo 2.2.3, a compreensão da concorrência pode ser desafiadora para estudantes universitários. Com o objetivo de facilitar o acesso ao conhecimento e ajudar os alunos a compreender melhor os conceitos relacionados à concorrência, este trabalho tem como meta o desenvolvimento de uma plataforma capaz de lidar com vários algoritmos de solução. No entanto, para estabelecer uma base sólida que possa ser ampliada para diferentes algoritmos, iremos nos concentrar em um dos mecanismos de sincronização relacionados a concorrência: os semáforos.

Um semáforo é um objeto de sincronização usado para controlar o acesso a um recurso compartilhado em um sistema concorrente. Ele é comumente utilizado para proteger uma seção crítica de código ou coordenar o acesso a um recurso compartilhado, como um arquivo ou uma conexão de rede, entre várias *threads* ou processos. Os semáforos podem ser implementados como semáforos binários (com dois estados) ou de contagem (com vários estados), e podem ter escopo local ou global.

Assim, uma vez que uma operação de semáforo é iniciada, é garantido que nenhum outro processo possa acessar o semáforo até que a operação seja concluída ou bloqueada. Dessa forma, o problema de concorrência de acesso à seção crítica é resolvido.

Com isso, separamos dois dos problemas mais comumente apresentados em sala de aula envolvendo impasses:

4.2.1.1 Jantar dos filósofos

O jantar dos filósofos é um problema clássico que explora desafios de concorrência e sincronização. Nele, cinco filósofos estão sentados em uma mesa redonda com hashis entre eles. Cada filósofo precisa pegar dois hashis para comer, mas só pode pegar os hashis adjacentes. O impasse ocorre quando todos os filósofos tentam pegar o hashi à sua esquerda ao mesmo tempo.

Para resolver esse problema, é necessário implementar um algoritmo de sincronização, como o do "hashi único", que permite que os filósofos peguem os hashis de forma ordenada e evitem o impasse. Além disso, é importante estabelecer um limite de tempo para evitar a inanição, garantindo que todos os filósofos tenham a oportunidade de comer. O objetivo é alcançar uma solução que permita que os filósofos jantem de forma eficiente, evitando tanto o impasse quanto a inanição.

4.2.1.2 Sistema Produtor/Consumidor

O problema do produtor/consumidor envolve a comunicação e sincronização entre processos que são dividas entre dois tipos: produtores e consumidores. Produtores são processos responsáveis por gerar dados (ou itens) e os depositar em um buffer compartilhado. Já os processos consumidores são responsáveis por retirar esses itens do buffer e os "consumir".

O objetivo é garantir que os produtores não depositem itens em um buffer cheio e que os consumidores não tentem retirar itens de um buffer vazio. Isso exige a coordenação adequada entre os produtores e consumidores para evitar condições de corrida e garantir a consistência dos dados.

4.2.2 Alocação de Arquivos

A alocação de arquivos em um SO envolve a organização e gerenciamento de espaço em disco para armazenar os arquivos. De acordo com Tanenbaum e Bos (2014), implementadores "estão interessados em como arquivos e diretórios são armazenados, como o espaço em disco é gerenciado e como fazer tudo funcionar de forma eficiente e confiável". Essa busca por uma forma eficiente e confiável faz com que a alocação de arquivos tenha não só múltiplas camadas de implementação como também conceitos abstratos mais complexos.

Pensando sobre a implementação de arquivos, temos então os i-nodes. Utilizado por sistemas estilo Unix, é uma estrutura de dados que lista os atributos e endereços de disco de objetos de arquivos de sistema como arquivos e diretórios que acreditamos ser um bom exemplo de mecanismo de um SO a ser representado em nosso trabalho. A figura 4.1 demostra sua estrutura.

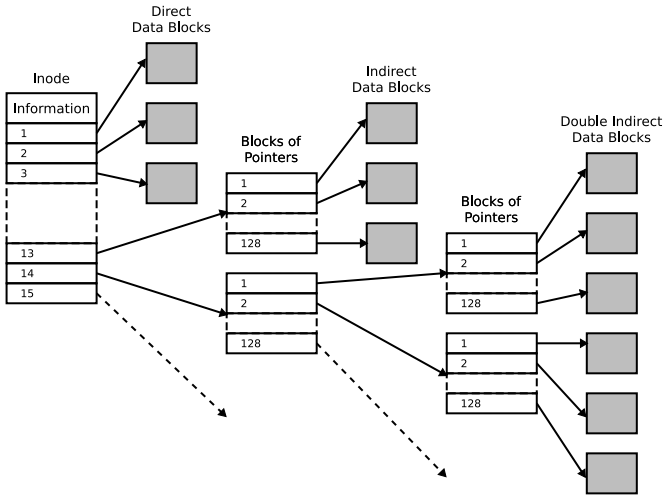


Figura 4.1: Exemplo de uma estrutura de i-nodes (Wikipedia, the free encyclopedia, 2023).

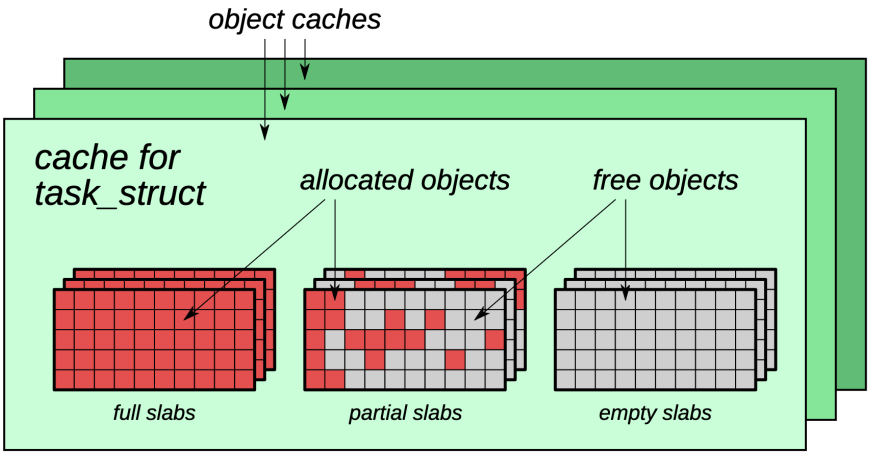


Figura 4.2: Estrutura de caches, slabs e objetos do alocador Slab, Maziero pg. 201 (Maziero, 2020).

4.2.3 Alocador Slab

O alocador Slab é uma técnica de gerenciamento de memória utilizada em sistemas operacionais para alocar e liberar memória de forma eficiente. Ele funciona dividindo a memória em pools ou bancos de memória chamados de "slabs", que são pré-alocados e organizados por tamanho de objeto. Cada slab consiste em uma série de objetos do mesmo tamanho, onde cada objeto possui um cabeçalho e os dados propriamente ditos.

Quando um objeto precisa ser alocado, o alocador Slab verifica se há um slab disponível no pool correspondente ao tamanho do objeto desejado. Se houver, o objeto é retirado do slab e retornado ao solicitante. Caso contrário, o alocador Slab solicita uma nova página de memória do sistema operacional, cria um novo slab para o tamanho do objeto e retorna o objeto solicitado, evitando o desperdício de memória, pois os slabs são reutilizados quando os objetos são liberados, eliminando a necessidade de alocar e desalocar memória constantemente. Além disso, a técnica do Slab permite um acesso mais rápido à memória, já que os objetos do mesmo tamanho estão agrupados, reduzindo a fragmentação e melhorando a eficiência da *cache*. Um exemplo de uso do algoritmo pode ser visto na figura 4.2.

4.2.4 Escalonamento de tarefas

Um escalonador de tarefas é o componente do Sistema Operacional responsável por gerenciar o uso do processador por múltiplos processos ou *threads* concorrentemente. Ele determina a ordem em que os processos são executados e quanto tempo cada um deles pode utilizar a CPU. O objetivo principal de um escalonador é otimizar o desempenho do sistema, minimizando o tempo ocioso da CPU e garantindo uma distribuição justa de recursos entre os processos. Escalonadores podem ser implementados usando diferentes algoritmos de escalonamento, cada um com suas próprias estratégias e critérios de decisão.

4.3 CONSIDERAÇÕES

Durante o processo de pesquisa sobre trabalhos relacionados ao tema, identificou-se várias implementações de diferentes algoritmos, todas com o propósito de mostrar o funcionamento de cada um. No entanto, uma observação relevante em tais projetos foi a ausência de uma interação mais robusta com os algoritmos, bem como a limitação no acesso às suas implementações. Isto gerou uma lacuna, a qual destacou a necessidade de desenvolver algo que proporcionasse uma experiência mais interativa e disponível ao usuário.

Os algoritmos mencionados possuem renome no cenário acadêmico de sistemas operacionais. Ao apresentar suas implementações de maneira visual e interativa, antecipa-se que o processo de aprendizado nessa área possa ser simplificado, preenchendo, assim, a lacuna atual existente em projetos desenvolvidos.

Assim sendo, o propósito deste trabalho é conceber uma plataforma que proporcione uma experiência mais interativa ao usuário, permitindo acesso à algoritmos de sistemas operacionais de forma mais simples.

5 DETALHAMENTO

Este capítulo destina-se a fornecer uma visão abrangente da solução desenvolvida, a qual a nomeamos de ViSO, destacando os principais objetivos, arquitetura geral e tecnologias empregadas. A solução proposta busca não apenas atender às necessidades descritas ao decorrer deste Trabalho, mas também oferecer um primeiro passo de se oferecer um sistema simples, consistente e disponível em língua portuguesa que possa ser reutilizado ou aprimorado futuramente, além de ser desenvolvido com tecnologias atuais. A página inicial resultante pode ser vista na figura 5.1.

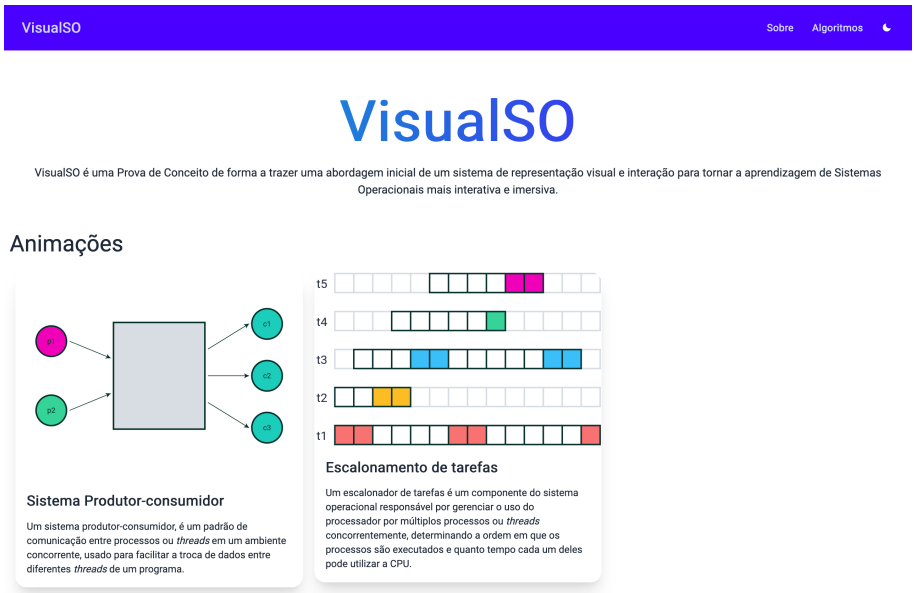


Figura 5.1: Página inicial do site.

5.1 ESCOLHA DE TECNOLOGIAS

Primeiramente, foi necessário escolher qual plataforma utilizar para desenvolvermos nossa aplicação: *web*, *mobile* ou *desktop*. Considerando o contexto específico de animações interativas, as aplicações *web* têm a grande vantagem de maior possibilidade de acesso, potencialmente funcionando em qualquer navegador moderno sem a necessidade de instalação de novos aplicativos, além de funcionarem independentemente de modelos dos dispositivos do usuário e seu respectivo sistema operacional. Essa amplitude de alcance proporciona uma plataforma eficiente para a distribuição de conteúdo interativo, atendendo a uma audiência diversificada. A compatibilidade multiplataforma é um fator importante, pois as aplicações *web* são desenvolvidas para funcionar de maneira consistente em diversos navegadores e dispositivos. Essa abordagem simplifica o processo de desenvolvimento, pois desenvolvedores podem focar em um único código-base, garantindo uma experiência homogênea para os usuários, independentemente do meio de acesso.

A flexibilidade e facilidade de atualização também são aspectos importantes diante desta escolha. Diferentemente das aplicações nativas, as atualizações em aplicações *web* são

mais ágeis, permitindo aos usuários acessar automaticamente as versões mais recentes sem a necessidade de downloads ou instalações complexas. Isso possibilita uma implementação e adesão por parte dos usuários mais rápida de novas funcionalidades e animações, mantendo a aplicação sempre atualizada.

Também considerando a popularização de desenvolvimento de *sites*, ferramentas e bibliotecas específicas para animações utilizando CSS3, SVG e bibliotecas JavaScript especializadas oferecem recursos robustos para a criação de animações.

Essas características tornam a escolha de desenvolver animações interativas em aplicações *web* uma estratégia envolvente e eficaz para proporcionar experiências dinâmicas e acessíveis na internet.

5.1.1 Vue.js e Nuxt.js

Escolhida a plataforma, foi então necessário escolher qual linguagem de programação, bibliotecas ou *frameworks* utilizar. Considerando a familiaridade que ambos os autores possuem com as linguagens JavaScript e TypeScript e sua popularidade atual, a escolha de um *framework* que as utilize simplificaria o desenvolvimento.

Com isso, decidimos utilizar Nuxt.js (Chopin et al., 2016b), um *framework* de desenvolvimento web construído sobre o Vue.js, destinado a simplificar a criação eficiente de aplicações. Vue.js (You, 2000), ou simplesmente Vue, por sua vez, é um framework de JavaScript progressivo para construção de interfaces de usuário interativas. Desenvolvido para facilitar o desenvolvimento web, o Vue.js adota uma abordagem de "componentização", permitindo a criação de interfaces modulares e reutilizáveis. Sua sintaxe é clara e simples e acompanhado por um sistema reativo eficiente, torna-o acessível a desenvolvedores de diferentes níveis de experiência – algo importante tanto para os autores de experiência de desenvolvimento *web* limitada quanto para futuros trabalhos que busquem aproveitar o sistema aqui desenvolto. O Vue.js também é conhecido por sua flexibilidade, integração gradual em projetos existentes, e por oferecer um ecossistema robusto com uma comunidade ativa, tornando-o uma escolha popular para a construção de aplicações web dinâmicas e eficientes.

Apesar de parecer estranho ou contraintuitivo, a prevalência de *frameworks* baseados em outros *frameworks* é uma prática cada vez mais comum em ferramentas de programação atuais, em grande parte impulsionada pela necessidade de eficiência no desenvolvimento e pela busca de uma automação de tarefas rotineiras. Essa abordagem é conhecida como "*batteries-included framework*"(*framework* de baterias inclusas), onde um *framework* (em nosso caso Nuxt.js) fornece uma estrutura abrangente que incorpora boas práticas, padrões de design e ferramentas comuns a um *framework* pré-existente (Vue.js). Uma das vantagens notáveis do Nuxt.js é a facilidade na configuração de rotas, utilizando uma estrutura de diretórios intuitiva, o que reduz a necessidade de configurações manuais. Além disso, foi preferido algo pensado em escalabilidade e que possa ser utilizado em trabalhos futuros, com um integração de diferentes módulos e sistemas de hospedagem de sites.

Nuxt.js também integra-se perfeitamente ao ecossistema Vue, aproveitando as vantagens dos componentes Vue. Sua abordagem modular e extensível permite a adição de funcionalidades por meio de módulos, facilitando a integração de recursos adicionais, como autenticação e análise.

Em suma, o Nuxt.js é uma ferramenta robusta e amplamente utilizada para o desenvolvimento ágil e escalável de aplicações web, reunindo características como renderização universal, configuração de rotas simplificada, integração com o ecossistema Vue, modularidade e simplicidade. Essas características fazem dele uma escolha popular entre desenvolvedores que buscam eficiência e desempenho em projetos Vue.js.

5.1.2 D3.js

Para definir qual biblioteca utilizar para criar animações buscamos primeiro saber quais tecnologias foram utilizadas para animar os sites do Data Structure Visualizations e VisuAlgo. O primeiro apresenta em seu próprio site uma seção que disponibiliza exemplos e código fonte de sua biblioteca de animações própria. Já, de acordo com o código presente no repositório de Yuwono (Yuwono, 2017), o VisuAlgo utiliza a biblioteca de animações D3.js (Bostock, 2013).

Além destas bibliotecas, outras já conhecidas para criação de animações como Anime.js, GreenSock Animation Platform e Three.js foram consideradas para adoção em nosso projeto. Foram então utilizados como critérios para escolha da biblioteca a ser utilizada o valor de licenças, qualidade de documentação, disponibilidade de material *online*, facilidade de uso e facilidade de criar e desenhar objetos customizados, o que culminou na escolha do D3.js.

O D3.js, ou simplesmente D3 (*Data-Driven Documents*), é uma biblioteca JavaScript para manipulação e visualização de dados interativos em páginas *web*. Desenvolvida por Mike Bostock, o D3 permite vincular dados a elementos do *Document Object Model* (DOM) de uma página HTML e aplicar transformações na representação visual desses dados.

A principal característica do D3 é sua abordagem "*data-driven*", onde os dados influenciam diretamente a criação e manipulação de elementos na página. Isto significa que manipular os dados utilizados para criar elementos visuais culmina diretamente em também manipular sua representação visual. Isso possibilita a criação de visualizações dinâmicas e interativas, que respondem automaticamente a mudanças nos dados.

D3.js também fornece uma ampla gama de funcionalidades para lidar com escalas, transições, eventos e manipulação de *Scalable Vector Graphics* (SVG), o que o torna mais fácil de criar visualizações de dados e objetos personalizados e complexos. Buscando compreender melhor seu funcionamento, encontramos uma playlist[1] de tutoriais que foi especialmente útil.

5.1.3 Tailwind e DaisyUI

Como última decisão do ferramental necessário para desenvolvimento, foi necessário escolher uma identidade visual para ser utilizada. Para isso, foi decidido utilizar uma biblioteca de interface de usuário (ou *UI library*, em inglês) para utilizarmos em nosso site. Sua utilização traz uma série de vantagens para o processo de desenvolvimento de aplicações, especialmente em contextos que envolvem *frameworks* como o Vue.js. Um dos principais benefícios é a significativa economia de tempo e esforço: ao fornecerem um conjunto de componentes pré-construídos e estilizados, essas bibliotecas permitem que os desenvolvedores evitem a necessidade de criar cada elemento da interface a partir do zero, acelerando o desenvolvimento. Além disso, elas também proporcionam consistência e identidade visual, seguindo diretrizes de design feitas por sua equipe de desenvolvedores, o que contribui para uma experiência do usuário mais concisa, uma vez que a aplicação adere a um estilo visual consistente em todas as suas partes.

Outra vantagem do uso destas bibliotecas é a capacidade de criar interfaces responsivas e compatíveis com diferentes dispositivos e tamanhos de tela. Muitas bibliotecas de UI são desenvolvidas com foco na responsividade, o que facilita a adaptação eficiente da interface a diferentes contextos, proporcionando uma experiência de usuário consistente independentemente do dispositivo utilizado.

Dentre as bibliotecas que foram consideradas para uso se encontram Vuetify (Leider e Leider, 2016), feita justamente para usar em conjunto com Vue, e Bootstrap (Otto, 2011), uma das bibliotecas mais conhecidas e desenvolvida desde o ano de 2011. Apesar disso, foi usada a

[1] https://youtube.com/playlist?list=PL6il2r9i3BqH9PmbOf5wA5E1wOG3FT22p&si=L-bBXt1kdkIeVZQP

DaisyUI (Saadeghi, 2020), uma "biblioteca de componentes"construída em volta do Tailwind (Tailwindcss, 2023), um dos *framework* CSS mais utilizados atualmente, com quase 400 milhões de downloads de acordo com o gerenciador de pacotes npm (npm, 2019). A escolha se deve muito pela popularidade do Tailwind, o estilo visual, uma ampla escolha de componentes para uso, o uso de temas e suas proporcionado pela DaisyUI e o fato de um dos autores ter experiência no uso da DaisyUI.

5.2 ALGORITMOS IMPLEMENTADOS

Este trabalho destaca a importância da representação visual e interação na educação, com ênfase em representações visuais de sistemas operacionais. Para dar vida ao projeto, foi criado um site para auxiliar na apresentação dos resultados, e abaixo, forneceremos explicações mais detalhadas sobre os algoritmos.

5.2.1 Produtor/consumidor

O problema do produtor/consumidor envolve a coordenação de processos (produtores e consumidores) que compartilham o acesso a um *buffer* de capacidade limitada. No desenvolvimento os produtores representam P(N) e os consumidores C(N). Sua representação pode ser observada na figura 5.2. Um resumo sobre como cada um deles funciona:

- **Participantes:**
  - **Produtor:** Produz e deposita itens no *buffer*.
  - **Consumidor:** Consome itens do *buffer*.
- ***Buffer*:**
  - Capacidade limitada (N itens).

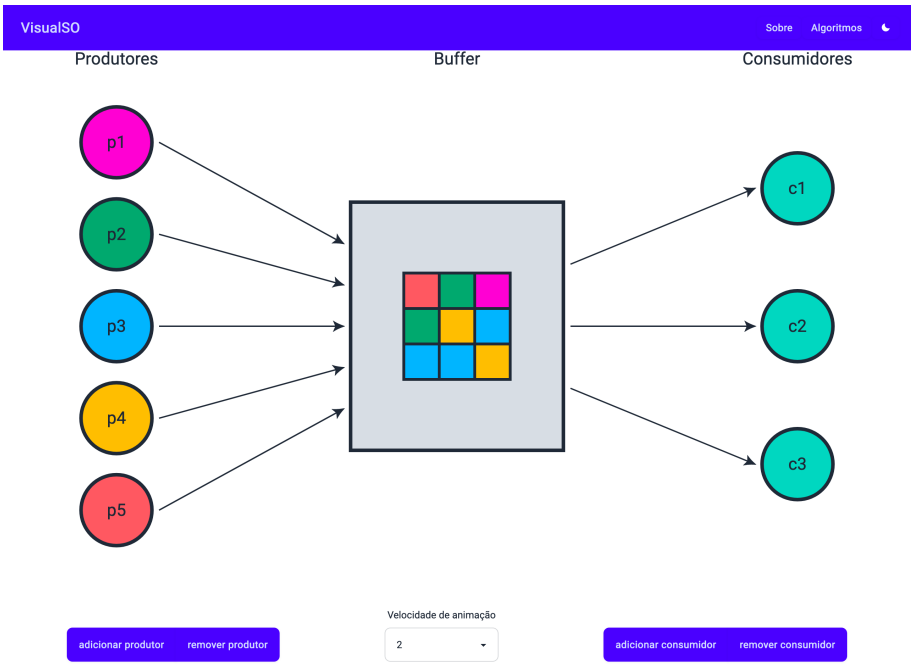


Figura 5.2: Visualização do algoritmo produtor/consumidor.

  - Compartilhado entre produtores e consumidores.

- **Comportamento:**
  - Produtores só podem depositar se houver espaço; caso contrário, eles esperam.
  - Produtores só irão produzir no momento em que for clicado no botão referente ao produtor escolhido.
  - Consumidores só podem consumir se houver itens; caso contrário, eles esperam.
  - Consumidores só poderão consumir no momento em que for clicado no botão referente ao consumidor escolhido.
  - É possível adicionar novos produtores, clicando no botão "adicionar produtor", e o mesmo comportamento para consumidores.
  - É possível remover os produtores atuais, clicando no botão "remover produtor", e o mesmo comportamento para consumidores.
  - Também é possível alterar a velocidade da animação, para melhor compreensão do algoritmo.
  - Apesar de não representar visualmente, produtores e consumidores que são clicados quando o *buffer* estiver cheio ficam em uma fila de espera, até que possam produzir ou consumir.

- **Bloqueio:**
  - O acesso ao *buffer* é bloqueante; os processos são bloqueados até que possam acessar o *buffer*.

- **Simetria:**
  - Produtores e consumidores têm comportamentos cíclicos e simétricos.

- **Efeito:**
  - O depósito bem-sucedido por um produtor "consome"um espaço vazio.
  - O consumo bem-sucedido por um consumidor "produz"um espaço vazio.

- **Ilustração:**
  - O problema frequentemente envolve múltiplos produtores e consumidores compartilhando um *buffer*.

5.2.2 Escalonador de tarefas

Outro problema escolhido para ser representado foi o escalonador de tarefas, que decidimos adicionar múltiplas opções de escolha de qual algoritmo de escalonamento o usuário poderá selecionar para comparação. O usuário poderá escolher qual algoritmo deseja executar selecionando uma das opções no menu *dropdown* à direita da tela e poder alterar informações como tempo de ingresso, duração e prioridade das 5 tarefas disponíveis na tabela à esquerda, como pode ser observado na figura 5.3.

A seguir, descrevemos as opções de algoritmos que podem ser selecionados pelo usuário.

Figura 5.3: Visualização do algoritmo escalonador de tarefas.

5.2.2.1 *First-Come, First Served*

O FCFS opera com base no princípio de atender as tarefas na ordem em que chegam à fila de prontos. As tarefas são colocadas em uma fila de prontos, e a tarefa que chega primeiro é a primeira a ser executada.

Pontos Chave:

1. **Ingresso:**
   - As tarefas são adicionadas à fila de prontos na ordem em que chegam ao sistema.
   - O tempo de chegada das tarefas determina sua posição na fila.
2. **Duração:**
   - A duração de uma tarefa é o tempo que leva para concluir sua execução.
   - As tarefas são executadas na ordem em que chegam até a conclusão.
3. **Prioridade:**
   - O FCFS não considera prioridades de tarefa. Ele segue uma abordagem não preemptiva, o que significa que uma vez que uma tarefa começa a ser executada, ela continua até a conclusão.
   - As tarefas são agendadas exclusivamente com base em seu tempo de chegada.

5.2.2.2 *Shortest Job First*

O algoritmo *Shortest Job First* (SJF) é uma técnica de escalonamento para otimizar a execução de processos. Ele prioriza a execução da tarefa mais curta da fila, visando minimizar o tempo total de processamento.

Pontos Chave:

1. **Ingresso:**
   - No SJF, os processos são adicionados à fila de prontos com base no momento em que chegam ao sistema.

- O tempo de ingresso dos processos determina a ordem em que estão na fila. Este ponto é serve para determinar a ordem de execução, já que o SJF prioriza processos com tempos de *burst* mais curtos. O conhecimento preciso do momento de ingresso é fundamental, especialmente em SJF não preemptivo, onde os processos são executados até a conclusão sem interrupções.

2. **Duração:**

   - A duração está relacionada ao tempo de *burst* de um processo, ou seja, quanto tempo leva para ser executado
   - Os processos são executados na ordem em que chegam até a conclusão. No SJF, a escolha do próximo processo é baseada na premissa de que o processo com a menor duração deve ser executado primeiro. Informações precisas sobre a duração de cada processo são essenciais para escolhas otimizadas.

3. **Prioridade:**

   - O SJF segue uma abordagem de escalonamento baseada na prioridade, onde a prioridade é determinada pelo tempo de *burst*. Menor tempo significa maior prioridade.
   - O SJF prioriza processos com tempos de *burst* menores, sendo uma abordagem otimizada para minimizar o tempo total de processamento.
   - Este é um modelo não preemptivo, ou seja, ele espera a conclusão de um processo antes de iniciar o próximo.

5.2.2.3 Round-Robin

O algoritmo Round-Robin opera distribuindo o tempo de CPU de maneira equitativa entre todos os processos na fila de prontos. Cada processo é executado por um curto período de tempo chamado *quantum*. Após o *quantum* expirar, o processo é movido para o final da fila, permitindo que o próximo processo na fila tenha a oportunidade de ser executado. Esse ciclo continua até que todos os processos sejam concluídos. O Round-Robin é conhecido por sua simplicidade e justiça, garantindo que todos os processos recebam uma fatia igual de tempo de CPU.

Pontos Chave:

1. **Ingresso:**

   - No algoritmo Round-Robin (RR), os processos são adicionados à fila de prontos na ordem em que chegam ao sistema.
   - O tempo de ingresso dos processos determina sua posição na fila e a ordem inicial de execução.

2. **Duração:**

   - A duração de um processo no Round-Robin é determinada pelo *quantum* de tempo, que é o intervalo máximo que um processo pode ser executado antes de ser preemptado.
   - Os processos são executados em pequenos fragmentos de tempo (*quantum*) de forma circular, retornando ao início da fila após cada execução.

3. **Prioridade:**

    - O Round-Robin é um algoritmo de escalonamento não preemptivo, o que significa que uma vez que um processo começa a ser executado, ele não é interrompido até que o *quantum* expire.
    - Não considera explicitamente prioridades de tarefa. Todos os processos têm igualdade de oportunidade, seguindo uma abordagem justa e balanceada.

5.2.2.4 Prioridade cooperativa

O Escalonamento Cooperativo por Prioridade prioriza processos mais importantes com base em suas prioridades mais altas. Garante que os processos críticos sejam atendidos primeiro, dependendo da correta atribuição de prioridades.

Pontos Chave:

1. **Ingresso:**

    - Os processos são adicionados à fila de prontos na ordem em que chegam ao sistema.
    - O tempo de ingresso dos processos determina sua posição na fila e a ordem inicial de execução.

2. **Duração:**

    - A duração de um processo é determinada pelo tempo necessário para concluir sua execução.

3. **Prioridade:**

    - No escalonamento cooperativo, a prioridade é um fator crítico. Cada processo possui uma prioridade associada, indicando sua importância relativa em relação aos outros processos.
    - Os processos são agendados com base em suas prioridades, garantindo que os processos mais prioritários sejam atendidos primeiro.

5.2.2.5 Prioridade preemptivo

No escalonamento por prioridade preemptivo, quando uma tarefa de maior prioridade está pronta para execução, o escalonador interrompe a tarefa atualmente em execução e concede o processador à tarefa mais prioritária. A tarefa interrompida retorna à fila de prontas. Esse processo, conhecido como "preempção", assegura que tarefas de alta prioridade sejam atendidas prontamente, garantindo uma execução eficiente e rápida.

Pontos Chave:

1. **Ingresso:**

    - No algoritmo de Escalonamento por Prioridade Preemptivo, os processos são adicionados à fila de prontos na ordem em que chegam ao sistema.
    - O tempo de ingresso dos processos determina sua posição na fila e a ordem inicial de execução.

2. **Duração:**

- A duração de um processo no Preemptivo é determinada pelo tempo necessário para concluir sua execução.
- Os processos são executados de acordo com suas prioridades, sendo interrompidos se uma tarefa de prioridade mais alta estiver pronta para execução.

3. **Prioridade:**

- No escalonamento por prioridade preemptivo, cada processo possui uma prioridade associada, indicando sua importância relativa em relação aos outros processos.
- Quando uma tarefa de maior prioridade está pronta para execução, ela preempta a tarefa em execução, recebendo o processador. A tarefa anterior retorna à fila de prontas.

5.2.2.6 Prioridades dinâmicas

O Escalonamento por Prioridades Dinâmicas é uma abordagem em que as prioridades das tarefas podem ser ajustadas dinamicamente durante a execução do sistema. Na implementação deste trabalho, o valor da prioridade dinâmica é atualizado a cada intervalo de tempo.

Pontos Chave:

1. **Ingresso:**

- No algoritmo de Escalonamento por Prioridade Preemptivo, os processos são adicionados à fila de prontos na ordem em que chegam ao sistema.
- O tempo de ingresso dos processos determina sua posição na fila e a ordem inicial de execução.

2. **Duração:**

- A duração de um processo é determinada pelo valor da prioridade, que é atualizado a cada intervalo de tempo.

3. **Prioridade:**

- É ajustada a cada turno com base no processo de envelhecimento, ou seja, o tempo de execução do algoritmo.

Quando selecionado o algoritmo RR, o usuário também poderá escolher o valor do *quantum* a ser usado em uma caixa de texto que deverá aparecer abaixo do menu de seleção de algoritmos. Por fim, também optamos em adicionar um modo escuro, proporcionando uma experiência de visualização mais confortável em ambientes com baixa luminosidade como pode ser observado na figura 5.4. Esta opção que pode ligada ou desligada clicando no ícone de sol ou lua topo direito superior da página.

5.3 ARQUITETURA DO VISO

Pela escolha de usar o Nuxt.js, a arquitetura de nosso projeto passa a utilizar a estrutura de diretórios já padronizada para aplicações Nuxt, que é possível visualizar em sua documentação (Chopin et al., 2016a). Disso, podemos separar algumas das funcionalidades que tiramos proveito durante o desenvolvimento:

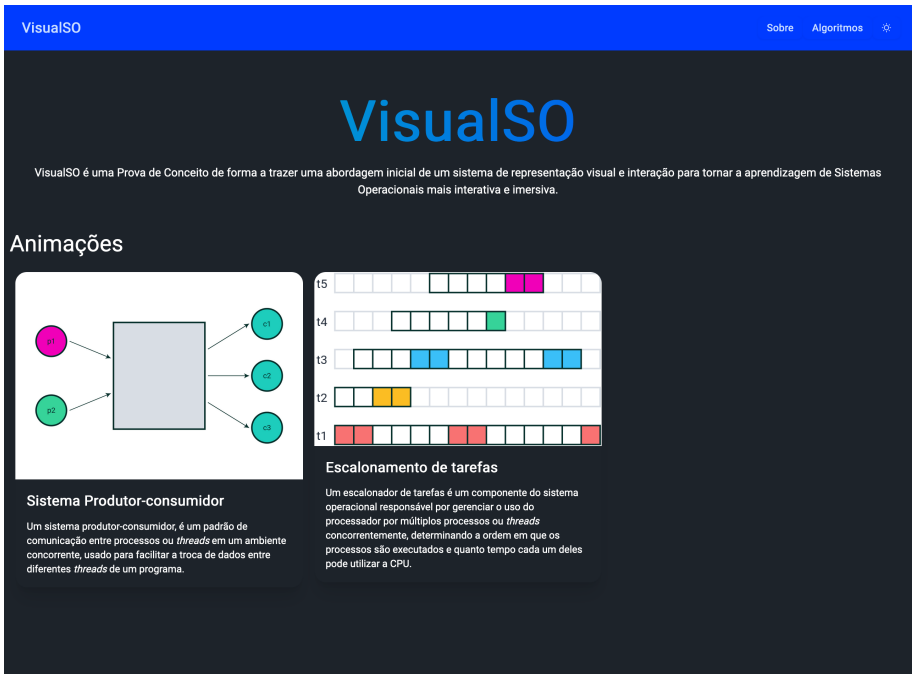


Figura 5.4: Página inicial do site em modo escuro.

- **Roteamento**

  Uma das maiores facilidade providas pelo uso do Nuxt, o roteamento é feito através dos arquivos presentes no diretório 'pages'. Cada arquivo de extensão .vue corresponde a uma rota na aplicação. Por exemplo, um arquivo chamado 'index.vue' em 'pages' representa a rota raiz, enquanto um arquivo sobre.vue representa a rota /sobre. De forma semelhante, temos as rotas aninhadas, feitas através da organização de arquivos em subdiretórios dentro de pages. Se tivermos um diretório 'blog' e um arquivo 'index.vue' dentro desse diretório, ele representará a rota /blog, enquanto um arquivo 'sobre.vue' dentro do diretório 'blog' representa a rota /blog/sobre.

  O Nuxt também apresenta outras possíveis configurações e ferramentas de rotas, como rotas dinâmicas e redirecionamentos, que não foram utilizados na aplicação desenvolvida.

- ***Layout* de páginas**

  O sistema de layout de página no Nuxt.js oferece uma maneira flexível de personalizar o *layout* das páginas em uma aplicação e, semelhante com o sistema de roteamento, são configuráveis a partir de arquivos presentes no diretório 'layouts'. Criamos em nossa aplicação um *layout* padrão para todas as páginas presentes ao adicionar um arquivo 'default.vue' em 'layouts', com a configuração de nossa barra de navegação utilizada. Também é possível definir um padrão de *layout* ao adicionar novos arquivos de extensão .vue no diretório, mas não houve necessidade para tal durante o desenvolvimento.

- **Padrão de arquivo**

  A estrutura de um arquivo Vue, geralmente definido com a extensão .vue, é composta por três seções principais: template, script e style. Essa lógica de estrutura se dá por conta da *Single File Component* (SFC), uma abordagem do Vue.js que permite encapsular a lógica, o *template* e estilos de um componente em um único arquivo.

A seção template contém o código muito semelhante ao HTML, que representa a estrutura da interface do usuário. É nesta parte em que o componente que será renderizado é definido. O Vue.js utiliza uma sintaxe especial chamada Vue Template Syntax para facilitar a vinculação de dados e a manipulação do DOM.

A seção script contém o código JavaScript que define o comportamento e a lógica do componente. Aqui é possível definir propriedades de dados, métodos, hooks do ciclo de vida e importa dependências necessárias. A seção script é escrita em JavaScript (ou TypeScript, se configurado).

A seção style contém o código CSS que estiliza o componente. Estilos em formato CSS ou de pré-processadores de estilos como SASS ou LESS podem também ser utilizados. Os estilos aqui definidos podem ser opcionalmente definidos somente dentro do escopo do componente adicionando a opção de 'scoped', garantindo encapsulamento.

Podemos observar um exemplo a seguir de como se é construído um arquivo Vue:

Listing 5.1: Exemplo de código em Vue.js.

```
<template>
  <div>
    <h1>{{ mensagem }}</h1>
    <button @click="exibirMensagem">Clique-me</button>
  </div>
</template>


<script setup>
let mensagem = 'Olá, Vue!';

function exibirMensagem() {
  alert(mensagem);
}
</script>

<style scoped>
div {
  background-color: lightgray;
  padding: 20px;
}

h1 {
  color: blue;
}

button {
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  cursor: pointer;
}
</style>
```

5.4 DIFICULDADES E LIMITAÇÕES

Criar um site dedicado a animações de algoritmos pode apresentar desafios significativos ao longo do processo. Uma dessas dificuldades reside na complexidade intrínseca dos algoritmos. Algoritmos frequentemente envolvem uma lógica complexa e com certos detalhes técnicos que podem dificultar a representação visual fiel e educativa por meio de animações. Traduzir conceitos abstratos de maneira clara, compreensível e mantendo a precisão das informações transmitidas é um grande desafio.

5.4.1 Dificuldades

Uma das dificuldades está na visualização de processos que podem operar em conjuntos de dados extensos ou estruturas de dados complexas. Tornar esses processos acessíveis e informativos para o usuário final pode ser uma tarefa desafiadora, especialmente quando se lida com algoritmos mais avançados ou abstratos, que exigem uma representação visual clara. Um bom exemplo disso é criar uma representação de alocações de arquivos que possa, ao mesmo tempo, representar como são criados os i-nodes e tornar nossa representação adaptável a possíveis interações com usuários.

A questão da usabilidade e experiência do usuário também é um ponto importante. Equilibrar a riqueza de informações nas animações com uma interface de usuário amigável é crucial. É necessário garantir que as animações sejam educativas sem sobrecarregar o usuário com detalhes excessivos, proporcionando uma experiência intuitiva que incentive a exploração e a compreensão. Apesar dos algoritmos escolhidos serem relativamente simples e baseados em representações previamente feitas por Maziero (2020), saber como melhor elaborar animações para que estes algoritmos "ganhem vida"foi algo que levou mais tempo que o imaginado.

Por fim, a maior dificuldade da dupla foi um dos mais comuns: a inexperiência. Apesar de ambos terem uma certa familiaridade com JavaScript, ambos possuíam um conhecimento limitado sobre o desenvolvimento de uma aplicação desde o seu começo, além de pouca a nenhuma experiência com algumas das tecnologias que decidimos utilizar. Aliado a isso, o tempo limitado para o desenvolvimento da aplicação teve um impacto direto para que pudéssemos aprender sobre como utilizar estas tecnologias e aplicá-las.

5.4.2 Limitações

Com isso, alguns comprometimentos tiveram de ser tomados durante o desenvolvimento. O primeiro deles foi abrir mão da compatibilidade de nossas animações em dispositivos móveis. Apesar de conseguirmos criar animações que funcionem de forma responsiva em telas maiores, a compatibilidade com telas menores de celulares e dispositivos semelhantes se tornou um desafio próprio por conta de suas resoluções. Outra limitação de nosso trabalho foi a falta de testes em diferentes navegadores e diferentes versões.

E, finalmente, gostaríamos de ter realizado testes práticos com alunos que estivessem atualmente cursando a disciplina de sistemas operacionais. Nosso objetivo era avaliar se o que desenvolvemos teria o potencial de auxiliar os estudantes na compreensão dos temas abordados. No entanto, lamentavelmente, isso não foi possível devido à indisponibilidade da disciplina de sistemas operacionais em nosso currículo durante a realização deste trabalho.

6 CONCLUSÃO

Este trabalho abordou a importância da representação visual e interação na educação, com um foco específico no uso de representações para sistemas operacionais com fins educacionais, destacando como a tecnologia pode auxiliar no processo de aprendizagem. Através de uma revisão da literatura e análise comparativa de ferramentas de simulação e visualização, demonstramos o potencial das representações visuais de sistemas operacionais para aprimorar a experiência de aprendizado e promover uma melhor compreensão de conceitos complexos.

Nossa abordagem proposta de criar um site, que colabora ao ensinar mecanismos de sistemas operacionais, oferece um recurso que pode ser valioso para educadores e estudantes interessados em explorar os benefícios dessa abordagem. Ao aproveitar recursos digitais e ferramentas interativas, podemos criar um ambiente de aprendizado mais imersivo e envolvente que possa estimular alunos a terem uma melhor compreensão e retenção de conceitos-chave.

Em conclusão, este trabalho busca contribuir para o campo da tecnologia educacional, destacando o potencial das representações visuais e interação para aprimorar a experiência de aprendizado. Esperamos que este trabalho sirva como um recurso útil para educadores e estudantes, inspirando futuras pesquisas e inovações neste campo.

O resultado deste trabalho pode ser encontrado em formato de código[1] e também pode ser encontrada sua versão implementada no site[2].

6.1 TRABALHOS FUTUROS

Diante da restrição mencionada anteriormente, uma abordagem subsequente seria conduzir testes práticos com alunos atualmente matriculados em disciplinas relacionadas a sistemas operacionais. Essa iniciativa visa avaliar a eficácia do projeto, considerando a indisponibilidade da disciplina específica durante a execução deste trabalho.

Olhando para o futuro, ainda há muito a ser explorado neste campo. Pesquisas futuras poderiam se concentrar no desenvolvimento de ferramentas de visualização e simulação mais avançadas, bem como explorar o potencial de outras tecnologias emergentes, como realidade virtual e aumentada. Além disso, estudos adicionais poderiam investigar a eficácia das representações visuais em outras áreas da educação além de sistemas operacionais.

Um aspecto adicional que merece destaque é a utilização das tecnologias abordadas neste trabalho como base para o desenvolvimento de novos algoritmos mais complexos, promovendo avanços significativos na aplicação prática das tecnologias apresentadas.

Este trabalho não apenas delineia possíveis direções para a pesquisa futura, mas também destaca a importância de ampliar o escopo de aplicação das tecnologias emergentes na educação. Ao fazê-lo, busca-se não apenas aprimorar o entendimento dos conceitos existentes, mas também fomentar o surgimento de novas abordagens e soluções inovadoras.

[1] https://github.com/ViniTVS/visualso

[2] https://visualso.vercel.app/

REFERÊNCIAS

Al-Oudat, N. (2017). A framework for visual modular design of educational operating system. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2-8):95–98.

B. Andrião, A., Darambaris, J. e V. Ferreira, R. (2015). Hashi - mastering the japanese dinner. `http://lasdpc.icmc.usp.br/~ssc640/grad//ec2015/hashi/`.

Bostock, M. (2013). D3.js - data-driven documents. `https://d3js.org`.

Brame, C. (2016). Active learning. *Vanderbilt University Center for Teaching*.

Brown, J. S. e VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science*, 4(4):379–426.

Chopin, A., Chopin, S. e Parsa, P. (2016a). app.vue · nuxt directory structure.

Chopin, A., Chopin, S. e Parsa, P. (2016b). Nuxt: The intuitive web framework.

de França Cabrini, M., Raad, R. e de Andrade Santos Weigert, R. (2015). Sws - sleep/wakeup simulator. `http://lasdpc.icmc.usp.br/~ssc640/grad/bcc2015/grupoa12`.

Flanagan, C., Freund, S. N. e Qadeer, S. (2005). Scaling static race detection for large java programs. *ACM SIGPLAN Notices*, 40(10):308–319.

Galles, D. (2011). Data structure visualization. `https://www.cs.usfca.edu/~galles/visualization/`.

Grissom, S., McNally, M. F. e Naps, T. (2003). Algorithm visualization in cs education: comparing levels of student engagement. Em *Proceedings of the 2003 ACM symposium on Software visualization*, páginas 87–94.

Halim, S., Halim, F., Zi Chun, K. e Loh, V. (2011). visualising data structures and algorithms through animation - visualgo.

Hill, J. H. e Gokhale, A. S. (2005). Visual os: Design and implementation of a visual framework for learning operating system concepts. Em *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 1*, ACM-SE 43, página 355–358, New York, NY, USA. Association for Computing Machinery.

Hundhausen, C. e Douglas, S. (2000). Using visualizations to learn algorithms: should students construct their own, or view an expert's? Em *Proceeding 2000 IEEE International Symposium on Visual Languages*, páginas 21–28.

Karavirta, V., Korhonen, A. e Seppälä, O. (2013). Misconceptions in visual algorithm simulation revisited: On ui's effect on student performance, attitudes and misconceptions. Em *2013 Learning and Teaching in Computing and Engineering*, páginas 62–69.

Korhonen, A., Malmi, L., Silvasti, P., Nikander, J., Tenhunen, P., Mård, P., Salonen, H. e Karavirta, V. (2009). Trakla2. Em *Proceedings of the 9th Koli Calling International Conference on Computing Education Research*, páginas 43–46.

Leider, J. e Leider, H. (2016). Vuetify — a material design framework for vue.js. https://vuetifyjs.com/.

Maziero, C. (2020). *Sistemas Operacionais: Conceitos e Mecanismos*. DINF - UFPR.

Moreno, R., Ozogul, G. e Reisslein, M. (2011). Teaching with concrete and abstract visual representations: Effects on students' problem solving, problem representations, and learning perceptions. *Journal of Educational Psychology*, 103(1):32–47.

Nariman, D. (2021). Impact of the interactive e-learning instructions on effectiveness of a programming course. Em Barolli, L., Poniszewska-Maranda, A. e Enokido, T., editores, *Complex, Intelligent and Software Intensive Systems*, páginas 588–597, Cham. Springer International Publishing.

Netzer, R. H. B. e Miller, B. P. (1992). What are race conditions? some issues and formalizations. *ACM Lett. Program. Lang. Syst.*, 1(1):74–88.

npm, I. (2019). Tailwindcss. https://www.npmjs.com/package/tailwindcss.

Otto, M. (2011). Bootstrap. https://getbootstrap.com/.

Pacheco, L. d. O. e Costa, R. d. O. (2011). Simularso - simulador de recursos de sistemas operacionais. Trabalho de Conclusão de Curso. Universidade Católica de Santos.

Pham, H. (2022). A web-based interactive and visualized approach to simulations of operating systems. Em *2022 5th International Conference on Information and Computer Technologies (ICICT)*, páginas 148–152.

RA3Player (2013). Animation about tcp/ip protocol. https://www.youtube.com/watch?v=7Zf203Vmbig.

Raiyn, J. (2016). The role of visual learning in improving students' high-order thinking skills. *Journal of Education and Practice*, 7(24):115–121.

Robertson, I. (2000). Imitative problem solving: Why transfer of learning often fails to occur. *Instructional Science*, 28(4):263–289.

Robertson, I. e Kahney, H. (1996). The use of examples in expository texts: Outline of an interpretation theory for text analysis. *Instructional Science*, 24(2):93–123.

Saadeghi, P. (2020). Daisyui — tailwind css components. https://daisyui.com/.

Scalet Bicalho, G., Antognoni de Castro, L. e Carvalho França, D. (2015). Syscalls game. http://lasdpc.icmc.usp.br/~ssc640/grad/bcc2015/grupoa11/.

Scratch (2018). Scratch - imagine, program, share. https://scratch.mit.edu/.

Seppälä, O., Malmi, L. e Korhonen, A. (2006). Observations on student misconceptions—a case study of the build – heap algorithm. *Computer Science Education*, 16(3):241–255.

Shabiralyani, G., Hasan, K. S., Hamad, N. e Iqbal, N. (2015). Impact of visual aids in enhancing the learning process case research: District dera ghazi khan. *Journal of education and practice*, 6(19):226–233.

Stokes, S. (2002). Visual literacy in teaching and learning: A literature perspective. *Electronic Journal for the integration of Technology in Education*, 1(1):10–19.

Strömbäck, F., Mannila, L. e Kamkar, M. (2018). Exploring students' understanding of concurrency. Em *Proceedings of the 23rd Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '18)*, páginas 50–55.

Sweller, J., Van Merrienboer, J. J. G. e Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, 31:261–292.

Tailwindcss (2023). Tailwind css - rapidly build modern websites without ever leaving your html. `https://tailwindcss.com/`.

Tanenbaum, A. S. e Bos, H. (2014). *Modern Operating Systems*. Pearson, Boston, MA, 4 edition.

Tree, S. (2020). How dijkstra's algorithm works. `https://www.youtube.com/watch?v=EFg3u_E6eHU`.

Wikipedia, the free encyclopedia (2023). inode pointer structure. `https://en.wikipedia.org/wiki/Inode_pointer_structure#/media/File:Ext2-inode.svg`. [Online; accessed July 6, 2023].

Yi-Ming Kao, G. e Ruan, C.-A. (2022). Designing and evaluating a high interactive augmented reality system for programming learning. *Computers in Human Behavior*, 132:107245.

You, E. (2000). Vue.js. `https://vuejs.org/`.

Yuwono, S. K. (2017). Visualgo-list. `https://github.com/yulonglong/VisuAlgo-List`.