CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Gabriel Alves de Miranda

OSLIVE: MÓDULO DE EXERCÍCIOS DO MECANISMO DE GERÊNCIA DE MEMÓRIA POR SEGMENTAÇÃO

Palmas – TO

2022

Gabriel Alves de Miranda

OSLIVE: MÓDULO DE EXERCÍCIOS DO MECANISMO DE GERÊNCIA DE MEMÓRIA POR SEGMENTAÇÃO

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Laboratório de Criação do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof.ª Madianita Bogo Marioti

Palmas – TO

2022

Gabriel Alves de Miranda

OSLIVE: MÓDULO DE EXERCÍCIOS DO MECANISMO DE GERÊNCIA DE MEMÓRIA POR SEGMENTAÇÃO

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Laboratório de Criação do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof.ª Madianita Bogo Marioti

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e. Madianita Bogo Marioti
Orientadora
Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e. Fabiano Fagundes
Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Fernanda Pereira Gomes
Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2022

RESUMO

Miranda, Gabriel Alves. **OSLIVE: MÓDULO DE EXERCÍCIOS DO MECANISMO DE GERÊNCIA DE MEMÓRIA POR SEGMENTAÇÃO**. 2022. Trabalho de Conclusão de Curso (Graduação) - Curso de Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2022.

Uma das principais características de um sistema operacional moderno está na capacidade de permitir que vários processos sejam executados e carregados simultaneamente na memória principal. Para satisfazer esse recurso, o Sistema Operacional (SO) faz uso de alguns mecanismo de gerência de memória sendo um deles a gerência de memória por segmentação. Essa é uma das técnicas que são ensinadas em praticamente todos os cursos da computação na disciplina de Sistemas Operacionais, que, tende a ser complexa por apresentar aspectos de baixo nível do computador. Pensando nisso, o OSLive foi desenvolvido a fim de auxiliar no processo de ensino e aprendizagem da disciplina por meio de simulações de exercícios que corroboram com os livros didáticos apresentados em sala de aula. Nesse contexto, o presente trabalho teve como objetivo criar um módulo de exercícios do mecanismo de segmentação do SO com funcionalidades onde o usuário possa interagir com o sistema e aprender a partir de simulações e resolução de exercícios. Desenvolvido em Vue Js (framework JavaScript de código-aberto), Bootstrap e jQuery, o presente trabalho permite ao usuário interagir com o sistema através da inserção de dados e resolução de exercícios, dessa forma, ampliando as possibilidades de simulação dos conceitos de segmentação do sistema operacional apresentados em sala de aula.

LISTA DE FIGURAS

Figura 1. Abstração de um sistema operacional 11
Figura 2. Exemplo Alocação de memória por segmentos 15
Figura 3. Janelas SOsim 17
Figura 4. Aba de Gerência de Memória 18
Figura 5. Metodologia para a realização do trabalho 22
Figura 6. Arquitetura do módulo de segmentação 24
Figura 7. Módulo de Gerência de Memória do OSLive 26
Figura 8. Criação de processos 27
Figura 9. Interface exercício Tabela de Segmentos 29
Figura 10. Interface exercício de Memória física 29
Figura 11. Execução de Exercício Tabela de Segmentos 31
Figura 12. Correção de resposta da Tabela de Segmentos e Processos 32
Figura 13. Visualizar resposta da Tabela de Segmentos 33
Figura 14. Execução de Exercício Tabela de Memória Física 34
Figura 15. Visualizar resposta Tabela de Memória Física 35

LISTA DE TABELAS

Tabela 1. Cronograma de etapas do projeto 25

LISTA DE ABREVIATURAS E SIGLAS

CEULP: Centro Universitário Luterano de Palmas

CPU: Central Process Unit

CSS: Cascading Style Sheets

E/S: Entrada e Saída

HTML: Hypertext Markup Language

MMU: Memory Management Unit

RAM: Random Access Memory

SO: Sistema Operacional

ULBRA: Universidade Luterana do Brasil

SUMÁRIO

1. Introdução 7
2. Referencial teórico 8
2.1. Ensino Didático Por Meio de Simulação Computacional 9
2.2. Sistemas Operacionais 10
2.3. Gerência de Memória 12
2.3.1. Gerência de Memória por segmentação 14
2.4. Trabalhos Correlatos 16
2.4.1. SOsim - Simulador para o Ensino de Sistemas Operacionais Versão 2.0 17
2.4.2. SWSO - Simulador Web de Sistemas Operacionais 18
3. Materiais e métodos 19
3.1. Materiais 19
3.1.1. Vue.js 19
3.1.2. Bootstrap 20
3.1.3. jQuery 21
3.2. Métodos 21
4. Resultados e Discussões 23
4.1. Arquitetura da aplicação 23
4.2. Módulo de simulação do mecanismo de segmentação 25
4.3. Módulo de Resolução de Exercícios 28
4.4. Execução dos Exercícios 30
5. Considerações Finais 36
Referências 37

1. Introdução

Um computador é formado basicamente por *hardware* e *software*, onde o primeiro é composto por circuitos eletrônicos (processador, memória, portas de entrada/saída etc.) e o segundo por programas destinados ao usuário do sistema como jogos, navegadores de internet e editores de texto (MAZIERO, 2020).

Ainda segundo Maziero, entre os aplicativos e *hardware* do computador existe uma camada de *software* denominada Sistema Operacional(SO), ele responsável, basicamente, por interligar aspectos de baixo nível, também conhecidos como linguagem de máquina, com os de alto nível, fornecendo uma aparência mais amigável e compreensível para a maioria dos usuários. Essa é uma ferramenta fundamental para o uso do computador, pois ela é responsável por expandir as funções do hardware no qual sem ela, o uso do mesmo seria praticamente inútil (QINGQIANG, 2009, tradução nossa).

De acordo com recomendações do MEC (2012), a disciplina de Sistemas Operacionais é obrigatória para os currículos de grande parte dos cursos da área de Computação do Brasil, pois oferece conceitos fundamentais sobre o gerenciamento dos recursos dos computadores. Além disso, o SO está presente em praticamente qualquer dispositivo inteligente como celulares, tablets, computadores de bordo e afins.

Dentre as diversas subáreas de Sistemas Operacionais, que vão desde o gerenciamento de processos à sistema de arquivos, existe o mecanismo de gerência de memória por segmentação que é uma das técnicas utilizadas para o gerenciamento de memória dos sistemas.

Para Deitel et. al 2005, Na segmentação: “Os dados e instruções de um programa são divididos em blocos chamados segmentos”. Esses segmentos possuem tamanhos distintos e são alocados na memória de forma contígua, vale ressaltar que a segmentação é um conceito de alocação lógica e não física como ocorre na paginação, estes conceitos serão explicados mais adiante.

No cenário educacional, a abordagem desse tipo de conteúdo pode se tornar um tanto quanto de difícil compreensão sem a ajuda de ferramentas complementares, no entanto, atualmente o panorama didático está cada vez mais dinâmico graças à inovação tecnológica e às sistematizações de metodologia de ensino. Com isso, muitas ferramentas didáticas foram criadas e são indicadas para o ensino de Sistema Operacional, dentre elas: SOsim - Simulador para o ensino de sistemas operacionais.

Segundo o site oficial do SOsim, o programa foi desenvolvido pelo prof. Luiz Paulo Maia, apresentado como parte de sua tese de mestrado no Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro (NCE/UFRJ) tendo como foco principal disponibilizar uma ferramenta gratuita que permitisse facilitar e melhorar as aulas de sistemas operacionais para alunos e professores.

Por se tratar de uma disciplina com diversos conceitos complicados que envolvem processos complexos de baixo nível e de difícil entendimento para o aluno, está sendo desenvolvido o ambiente web OSLive, que disponibiliza aplicações divididas em módulos que simulam resoluções de exercícios presentes nos livros e apresentados em sala de aula. O OSLive é um ambiente web que faz parte do Grupo de Estudos em Novas Tecnologias para processos de Ensino e Aprendizagem (GENTE) do Centro Universitário Luterano de Palmas (CEULP/ULBRA), que visa auxiliar no processo de ensino e aprendizagem da disciplina de Sistemas Operacionais fornecendo atividades de ensino e simulações dos processos realizados pelo SO.

Atualmente, já existe um módulo de simulação da gerência de memória por segmentação no OSLive, com interface gráfica que simula um cenário de compartilhamento de memória física de um computador. Tendo como base essa ideia, pensou-se em desenvolver um novo módulo de segmentação do OSLive a partir da primeira versão, de modo que o mesmo fique com a interface padronizada e mais próxima da didática dos livros recomendados na sala de aula.

Pensando nisso, foi implementado um módulo de resolução de exercícios do mecanismo de segmentação do OSLive, com interface padronizada e implementação de funcionalidades que permite ao usuário responder exercícios através da inserção de dados e visualização de respostas. Analisando a interface padrão do OSlive, examinando a versão de simulação de segmentação e elencando funcionalidades, foi possível criar uma aplicação de resolução de exercícios usando ferramentas de desenvolvimento web que oferecem suporte para criação do mesmo e apresentar os resultados dos testes funcionais no presente trabalho.

2. Referencial Teórico

Nesta seção são expostos os conceitos e processos relacionados ao funcionamento de um Sistema Operacional, com foco na gerência de memória por segmentação, bem como a

delineação das vantagens do ensino didático por meio de simulação computacional para o auxílio de aprendizagem e imersão de estudantes.

2.1. Ensino Didático por Meio de Simulação Computacional

Para Zem-Lopes et al 2014 “softwares educacionais web têm evoluído nos últimos anos, impulsionados pelos avanços das TIC(tecnologia da informação) e pela popularização das tecnologias da Web Semântica”. Para os autores, essa evolução tem promovido mudanças na forma com que produtos e serviços de diversos setores são disponibilizados, e com a educação não diferente, visto que, houve melhorias no processo de aprendizagem por meio de softwares educacionais.

Para Giraffa e Viccari (1998), qualquer programa pode ser considerado *software* educacional, desde que ele sirva para auxiliar alunos e professores no processo de ensino-aprendizagem dentro de um contexto escolar. Esses *softwares* buscam trazer aspectos do mundo real para dentro de simulações computacionais, seja através de representação gráfica ou interatividade com o usuário.

A formação de competências para os estudantes pode ser adquirida por meio de Softwares Educacionais, como simulações, por exemplo, pois além de proporcionar e instigar aspectos cognitivos do usuário, esse tipo de metodologia de ensino favorece para melhor abstração de conteúdo e ajuda no processo de aprendizado.

Em essência, “a simulação é uma técnica de ensino que se fundamenta em princípios do ensino baseado em tarefas e se utiliza da reprodução parcial ou total destas tarefas em um modelo artificial”. Essa definição, dada por Pazin Filho e Scarpelini (2007), é muito ampla e pode incluir diversas áreas do conhecimento humano. Segundo Smith (1998), “a simulação é o processo de projetar um modelo de um sistema real ou imaginário e conduzir experimentos com esse modelo”.

Por exemplo, um professor de física está interessado em construir um protótipo de motor elétrico a fim de demonstrar para os alunos o funcionamento de uma máquina capaz de converter energia em trabalho mecânico, mas não tem os recursos necessários para isso. Neste caso, seria interessante a utilização de um simulador capaz de apresentar visualmente a interação entre campos eletromagnéticos que faz o motor girar. Esse modelo descreve por exemplo os tipos de peças necessárias (eletroímãs, bobinas de cobre, imãs..), peso, medidas, *design* e assim por diante.

Por permitir a realização de testes em ambientes simulados e verificar resultados antes

de iniciar alguma ação no ambiente real, evitando gastos de recursos uma vez que se tem conhecimento prévio dos resultados, a simulação tem sido bastante utilizada nas mais diversas áreas do conhecimento. Além de possibilitar a visualização e o funcionamento de sistemas que não seriam visíveis no mundo real, esse recurso contribui para a compreensão de fenômenos para fins didáticos, científicos ou profissionais.

Para o ensino de Sistemas Operacionais não é diferente, também existem diversas ferramentas intuitivas(essas ferramentas serão apresentadas na seção 2.5) que ajudam tanto o aluno quanto o professor no processo de ensino/aprendizagem. Além de facilitar na compreensão do conceito e apresentar processos de baixo nível que um sistema operacional realiza, estas ferramentas ajudam também a despertar mais interesse por parte do aluno fazendo com ele direcione mais atenção ao conteúdo apresentado em aula que até então seria difícil visualizar claramente sem a ajuda de uma interface gráfica.

Com isso, a simulação computacional utilizada como apoio ao processo de ensino e aprendizagem possibilita entrever como o SO trabalha, isso pode auxiliar na compreensão dos conceitos abordados em aula. Parte destes conceitos serão apresentados nos próximos tópicos.

2.2. Sistemas operacionais

Tanenbaum (2010) diz que “Os sistemas operacionais realizam basicamente duas funções não relacionadas: Fornecer aos programadores de aplicativos um conjunto de recursos abstratos claro em vez de recursos confusos de hardware e gerenciar esses recursos de hardware”. Para eles o SO é basicamente um *software* intermediário que fica entre a linguagem de baixo nível e alto nível - “o sistema operacional procura tornar a utilização do computador mais eficiente e conveniente”.

A utilização mais eficiente busca um maior retorno no investimento feito no hardware, em outras palavras, melhor aproveitamento dos recursos de hardware. Uma utilização mais conveniente, tem relação com o fato de camuflar do usuário o acesso aos recursos de baixo nível, apresentando-as de maneira mais plena e de fácil compreensão. A figura 1 sintetiza essa ideia.

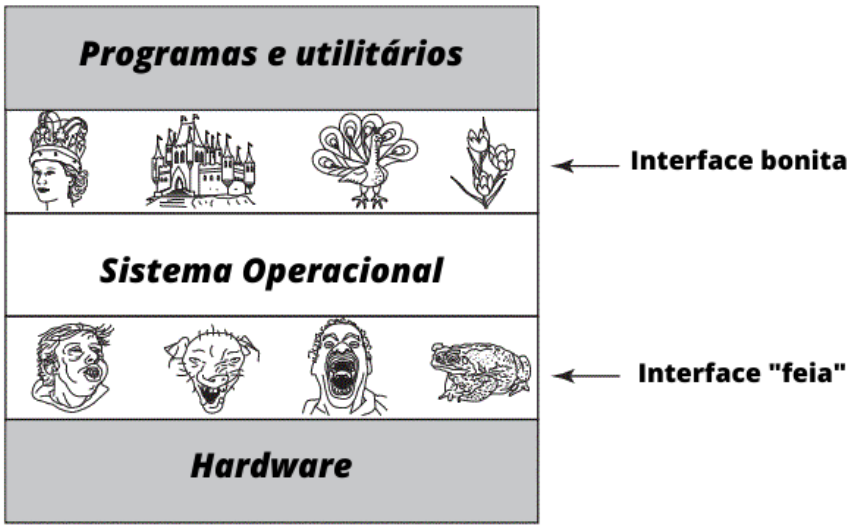
Para Tanenbaum (2010) existem duas visões para sistemas operacionais quando apresentados pela perspectiva do usuário *Top-down*: é uma abstração do hardware, em que o SO faz o intermédio entre os aplicativos (*software*) e os componentes físicos do computador *hardware*; *Bottom-up*: este é responsável por gerenciar os recursos inerentes às aplicações, memórias, discos e periféricos de entrada e saída.

Vale salientar que os usuários que lidam diretamente com o sistema operacional e suas abstrações são os programadores, que devem entender os processos inerentes a computação de baixo nível (linguagem de máquina), *Bottom-up*. Enquanto que o *Top-down*, diz respeito aos usuários que lidam com as abstrações de alto nível fornecidas através de linhas de comando, aplicativos ou GUIs (*Graphical User Interface*).

Para Martinez (2011), uma GUI fornece uma maneira fácil de interação entre homem e máquina pois tira proveito dos recursos gráficos dos computadores para tornar essa comunicação mais fácil, ocultando os detalhes da programação do usuário. Este tipo de interface usa janelas, ícones, menus, botões, listas suspensas, caixas de diálogo etc..

Sendo assim, uma das principais tarefas do sistema operacional é ocultar o hardware e apresentar programas com abstrações mais agradáveis e limpas de se interagir.. Os sistemas operacionais transformam o que é “feio” em belo, como mostra a Figura 1.

Figura 1. Abstração de um sistema operacional



Fonte: Adaptado de Tanenbaum (2006)

Observa-se na figura 1 que o sistema operacional funciona como intermediário entre linguagem de máquina, baixo nível, e programas utilitários, alto nível. Ele é responsável por unir dois contextos diferentes dentro do sistema, apresentando uma “interface bonita” e de fácil compreensão para o usuário final, programas utilitários, e fazer o gerenciamento de recursos visando a otimização do computador como um todo, manipulando o uso da memória e processos realizados em linguagem de máquina.

Para Machado e Maia (2005), através dos processos é possível alocar recursos, compartilhar dados, trocar informações e sincronizar sua execução. Nos sistemas multiprogramáveis os processos são executados concorrentemente, compartilhando o uso do processador, memória principal, dispositivos de E/S (entrada e saída) dentre outros recursos. Visto que o objetivo deste trabalho diz respeito aos princípios do Gerenciamento de Memória em Sistema Operacional, a próxima seção abordará detalhes sobre esse mecanismo.

2.3. Gerência de Memória

Cada tipo de memória é recomendada para determinada necessidade e arquitetura computacional, as mais velozes por exemplo como à memória RAM são acessadas diretamente pelo processador pois fornece acesso direto ao endereço solicitado. Também conhecida como memória principal, nela reside todos os programas e dados que serão executados ou referenciados pelo processador, um programa residente no HD por exemplo, para ser executado, deve ser de alguma forma carregado para a memória principal (MAIA, 2001).

De acordo com Senger e Guardia (2013), “para que os programas sejam executados, é preciso que tanto seus códigos quanto os dados que eles manipulam sejam carregados na memória principal. Uma vez presentes na memória principal, os programas podem fazer referências a endereços de trechos de código, como funções chamadas durante a execução, e também a endereços que correspondem a posições de memória que armazenam os conteúdos das variáveis”

O sistema operacional faz uso de algumas estratégias a fim de otimizar a memória memória do computador, elas servem para ditar como a memória será ocupada pelos processos, o local de carregamento dos processos e por quanto tempo eles deverão permanecer carregados na memória principal.

Por ser tratada como fundamental a um sistema operacional, a utilização de técnicas de gerenciamento de memória torna-se necessária. Para entender o gerenciamento de memória, dois conceitos devem ser abordados: a memória física e a memória lógica. Para Oliveira (2001), a memória lógica é aquela que o processo consegue enxergar, eles são gerenciados pelo SO e manipulados pelo processo. Já a memória física, como o próprio nome sugere, são os circuitos integrados de memória, a parte de hardware da memória do computador.

Para Silberschatz et al. (2015), o mapeamento de endereços virtuais para endereços físicos é feito por um dispositivo de hardware chamado de unidade de gerenciamento da memória (MMU — *Memory Management Unit*). Geralmente ele é implementado como parte

da CPU e é um dispositivo de hardware capaz de traduzir endereços virtuais em endereços físicos.

Nos conceitos abordados nos livros didáticos são apresentados diversos mecanismos de Gerenciamento de Memória com o fim de definir como será o uso da memória principal, como partições fixas, partições variáveis, paginação e segmentação. Esses mecanismos conceituais servem de base para a definição dos mecanismos de gerência de memória dos SOs atuais.

"A partição fixa é a forma mais simples de alocação de memória, que consiste em dividir a memória destinada aos processos em N partições fixas, de tamanhos iguais ou distintos". Essa estratégia abordada por Maziero (2011), se torna mais flexível quando surge a necessidade de ajustar o tamanho da partição de acordo com o tamanho do processo. Isso ocorre com as partições variáveis, elas permitem que os tamanhos das regiões possam variar em função das necessidades das tarefas, portanto, as partições são criadas de acordo com o tamanho de cada processo, assim então evitando o desperdício de memória, visto que ela é alocada de forma contígua.

As partições fixas e variáveis utilizam estratégia de alocação contígua, onde os processos são dispostos de forma sequencial sobre um conjunto de blocos consecutivos na memória, sem "buracos" entre os blocos, dessa forma, a localização do conteúdo do processo na memória é definida pelo endereço de seu primeiro bloco e isso fornece acesso sequencial rápido aos dados (MAZIERO, 2019).

Além da alocação contígua, outros mecanismos utilizam estratégia de alocação não contígua, como a paginação e segmentação. Para Oliveira 2001 "Quando um programa é carregado em uma área de memória maior que o necessário, isso resulta em um desperdício de memória, isto é, memória perdida dentro da área alocada para um processo", isso ocorre porque os programas não preenchem totalmente o espaço disponível da partição, pois ela não é alocada de forma contígua.

Esses mecanismos são utilizados principalmente como apoio à memória virtual. A ideia básica por trás da memória virtual é que o tamanho combinado do programa, dos dados e da pilha pode exceder a quantidade de memória física disponível para eles, ou seja, o sistema operacional mantém na memória principal as partes do programa correntemente em uso e o restante no disco rígido (TANENBAUM, WOODHULL, 2008).

A paginação e segmentação surgem com a necessidade de mais espaço na memória principal. Tanebaum e Woodhull, 2008 diz que "A paginação utiliza unidades lógicas chamadas páginas. Essas páginas fazem referência a espaços na memória física chamados de molduras de

página (ou quadros). As páginas e as molduras (quadros) têm sempre exatamente o mesmo tamanho". Já a segmentação, particiona a memória em blocos ou quadros de tamanho fixo, como a proposta deste trabalho está relacionado com a segmentação, este será explicado melhor na próxima seção.

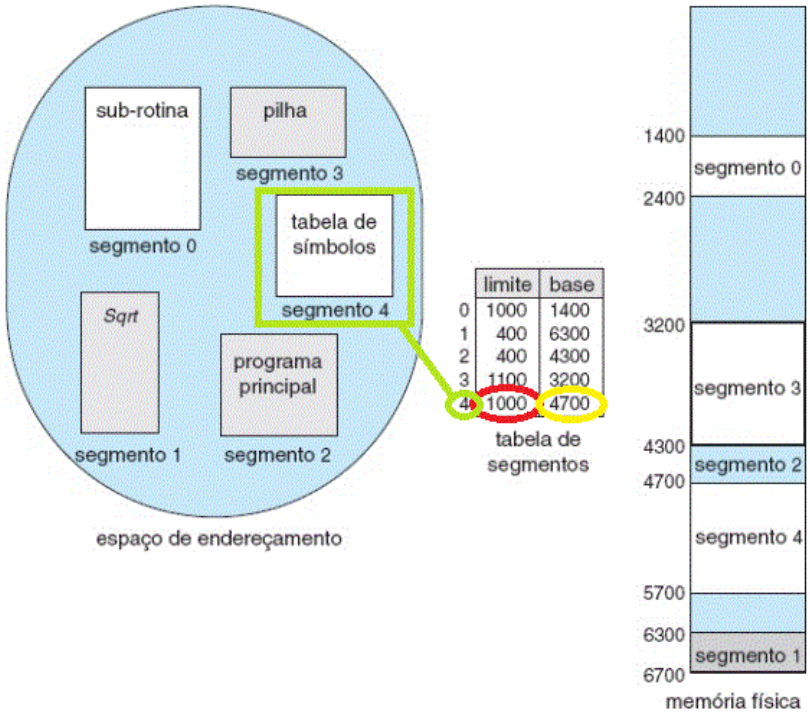
2.3.1. Gerência de Memória por segmentação

Na alocação por segmentos, ou alocação segmentada, o espaço de memória lógica de um processo é fracionado em áreas, ou segmentos, que podem ser alocados separadamente na memória física, ou seja, em áreas não contíguas. Podem ser definidos segmentos para itens específicos, como bibliotecas compartilhadas, vetores, matrizes, pilhas de threads, buffers de entrada/saída, etc (MAZIERO, 2011).

Para Gonçalves et al., 2010, cada segmento é constituído de uma sequência linear de endereços de 0 a algum número máximo, que se refere ao tamanho de cada segmento. Segmentos diferentes podem ter diferentes tamanhos, o que geralmente ocorre e pode variar durante a execução. Por exemplo, a divisão típica dos processos se dá em código, dado e pilha (inclusive na simulação de segmentação do OSLive), onde cada uma delas possui um tamanho diferente ou igual.

Como foi supracitado, os segmentos não precisam necessariamente possuir a mesma estrutura de dados, dessa forma, elas funcionam independentemente uma das outras e podem crescer ou reduzir de tamanho sem que afetem as demais. A figura 2 ilustra como geralmente é feita a divisão.

Figura 2. Exemplo Alocação de memória por segmentos



Fonte: Adaptado Silberschatz, Galvin e Gagne (2011)

A Figura 4 apresenta como é feita a segmentação de um programa, neste exemplo o processo foi dividido logicamente em 5 segmentos que refletem a sua estrutura funcional: rotinas, módulos, código, dados, pilha, com seus respectivos índices. Eles são responsáveis pela identificação do segmento na tabela de segmentos.

Segundo Silberschatz et al (2011) “a tabela de segmentos possui uma entrada separada para cada segmento, dando o endereço de início do segmento na memória física (a base) e o tamanho desse segmento (o limite)”. Por exemplo, o segmento 4, possui 1000 bytes de comprimento e começa na posição 4700, logo, uma referência ao byte 100 do segmento 4 é mapeado na posição 4700 + 100 = 4800.

Para encontrar o endereço físico, o sistema operacional realiza comparações entre o deslocamento fornecido e o limite do segmento. Por exemplo, um processo deseja manipular o segmento 4 que possui 1000 bytes, ele começa no endereço físico 4700 e termina no 5700, caso este processo tente realizar a leitura ou escrita ao byte 500 (deslocamento) desse mesmo segmento, o SO verificará se o deslocamento é maior ou menor que o limite do segmento (círculo vermelho da figura 2).

Caso o deslocamento seja maior, uma interrupção é gerada indicando que o processo está tentando endereçar fora da área de memória lógica, se o valor for menor, será somado o

deslocamento ao valor base (círculo amarelo da figura 2), 500 + 4700 = , encontrando assim o endereço lógico. Esse mecanismo de gerência pode gerar desperdício de espaço disponível em memória, existem dois tipos, a fragmentação interna e a fragmentação externa.

Para Oliveira (et al., 2015 p. 412) ”O mecanismo de segmentação pode gerar desperdícios de memória chamados de fragmentação externa. A fragmentação externa ocorre quando há espaço total na memória suficiente para atender a uma solicitação, mas os espaços disponíveis não são contíguos’. Por exemplo, um segmento de 1000 bytes e precisa ser alocado na memória física que possui 1500 bytes disponíveis, no entanto, este espaço está fragmentado em três brechas de 500 bytes cada, dessa forma, para que o segmento fosse alocado, seria necessário um espaço contíguo de no mínimo 1000 bytes de comprimento.

Além da fragmentação externa, existe também a fragmentação interna conforme apresentada por Oliveira et al (2015): “particionar a memória física em blocos de tamanho fixo e alocar a memória em unidades com base no tamanho do bloco". Nessa abordagem, a memória alocada a um processo pode ser um pouco maior do que a memória solicitada. A diferença entre esses dois números é a fragmentação interna — memória não utilizada que pertence a uma partição.”

A fragmentação interna e a fragmentação externa são fenômenos em que a memória é desperdiçada na memória contígua e memória não contígua, o primeiro ocorre na alocação de memória de tamanho fixo e o segundo na alocação de memória dinâmica, sendo que, tanto o problema da fragmentação interna e externa não podem ser totalmente eliminados, apenas reduzidos. Os conceitos de gerência de memória por segmentação e fragmentação externa estão diretamente relacionados com este trabalho, a próxima seção exemplifica a versão atual do OSLive e suas principais características no que diz respeito à segmentação de memória.

2.4. Trabalhos correlatos

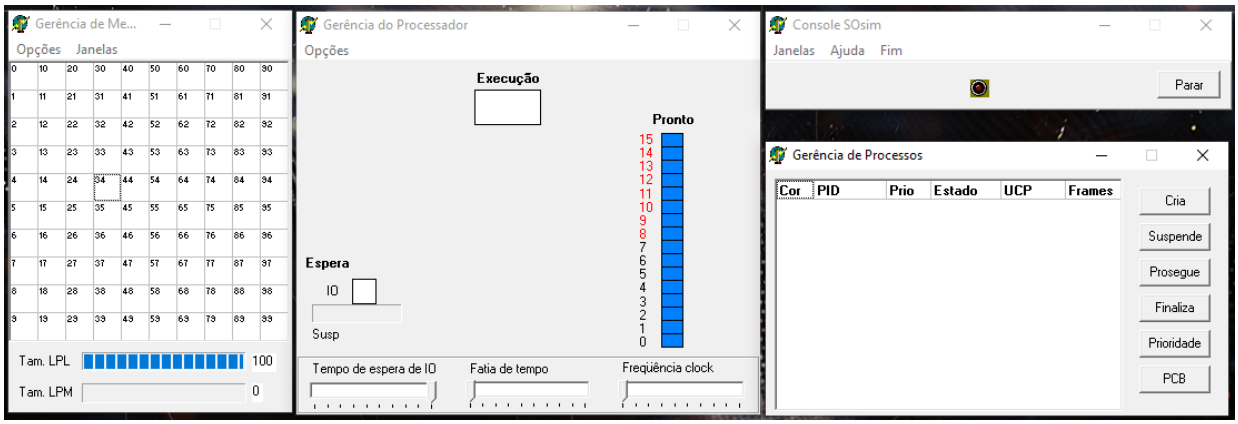
Esta seção tem como objetivo apresentar os trabalhos relacionados à proposta do presente trabalho. Tais trabalhos estão relacionados com o ensino de Sistema Operacional por meio de simulação, estes demonstram as principais características relevantes em um SO, cujo objetivo é prover o auxílio no aprendizado sobre gerência de memória no âmbito educacional e ajudar educadores no processo de ensino e alunos no de aprendizado.

2.4.1. SOsim - Simulador para o Ensino de Sistemas Operacionais Versão 2.0

Para Maia (2001) a o SOsim: “tem como objetivo auxiliar o ensino e aprendizado de sistemas operacionais. O SOsim servirá como ferramenta de suporte visual para aulas de sistemas operacionais”. Ele é bastante popular no meio acadêmico por praticamente ter sido feito baseado no livro "Arquitetura de Sistemas Operacionais", livro este que é bastante mencionado nas aulas de SO da CEULP-ULBRA.

O site do SOsim menciona que o sistema permite ao professor apresentar os conceitos e mecanismos de um sistema operacional multiprogramável e/ou multitarefa, como Unix, OpenVMS e Windows, de forma simples e animada. O simulador também permite visualizar os conceitos de multiprogramação, processos e suas mudanças de estado, gerência do processador (escalonamento) e a gerência memória. A partir da área de configuração do SOsim, é possível selecionar, por exemplo, diferentes políticas de gerência de memória e alterar o funcionamento do simulador. Desta forma, o aluno tem a oportunidade de visualizar os conceitos teóricos apresentados em aula de forma simples e animada. A visão das Janelas relacionadas às funcionalidade do Sosim é exibida na Figura 4.

Figura 3. Janelas SOsim



O SOsim é dividido em quatro janelas, cada uma responsável por representar as atividades de gerenciamento realizadas pelo SO, como gerência de processos, processador e memória. Apesar de ter o objetivo de apresentar os mecanismos de forma simples, o SOsim possui uma interface relativamente confusa, pois às vezes se faz necessário ter diversas janelas

abertas para compreender o funcionamento dos processos e as ações do sistema operacional. Outro problema é a incompatibilidade com outros sistemas, já que o mesmo foi desenvolvido para funcionar exclusivamente em sistemas operacionais Windows, portanto, ele não é compatível com outros sistemas.

2.4.2. SWSO - Simulador Web de Sistemas Operacionais

Souza e Oliveira (sem data) desenvolveram o SWSO (Simulador Web de Sistemas Operacionais), um simulador Web de Sistemas Operacionais que auxilia no processo de ensino-aprendizagem da disciplina de SO. A ferramenta tem como objetivo simular o funcionamento da mesma, explorando os conceitos de gerência de disco, memória e processos de forma integrada e interativa. Além disso, o software foi desenvolvido para permitir que qualquer pessoa com um mínimo de conhecimento na linguagem Java possa baixar o código da aplicação e implementar novos algoritmos de forma prática.

O objetivo principal do SWSO é fornecer um software confiável para a simulação do funcionamento de um sistema operacional. O simulador tende a oferecer ao professor uma ferramenta online de apoio às aulas, com o foco nos seguintes conceitos: Gerência de Disco, Gerência de Memória (memória virtual) e Gerência de Processos e para os alunos uma plataforma dinâmica que ajuda no processo de aprendizado que possibilite o aumento de interação entre professor e aluno durante as aulas.

Figura 4. Aba de Gerência de Memória

A gerência de memória do SWSO como mostra a figura 5, é dividida em Tabela de Processos e Memória Principal, nelas é possível observar o comportamento da memória ao

longo do tempo. Nesta tela é apresentado informações relevantes relacionadas aos processos ativos na memória principal, a quantidade de páginas que um processo tem na memória principal, a quantidade total e também a tabela de página dos processos. Cada processo contém o seu id (PID), a quantidade de páginas e a quantidade de *frames*/quadros ocupados. Nesta parte do programa é possível configurar a política de busca, a política de alocação, o algoritmo de substituição de páginas e a quantidade de frames que um processo pode ter na memória principal, vale notar que o usuário consegue associar o processo e a posição da memória que ele ocupa através da cor.

Ao contrário do SOsim, o SWSO é responsivo e pode ser executado em praticamente qualquer navegador, entretanto, apesar de sua proposta ser de apresentar um simulador que mostra o funcionamento de SO e explore os conceitos de gerência de disco, memória e processos, ficou ausente o conceito do mecanismo de gerência de memória por segmentação, já que só é possível realizar simulação e apresentar conceitos a respeito de gerência de memória por substituição de páginas.

3. Materiais e Métodos

Esta seção apresenta os recursos e os procedimentos que foram utilizados para o desenvolvimento da atualização do módulo de gerência de memória por segmentação do OSLive.

3.1. Materiais

Foi utilizado para o desenvolvimento do módulo de segmentação as seguintes ferramentas: Vue.js; Bootstrap e jQuery. O Vue JS define a lógica de programação, o Bootstrap cuida da interface responsiva e outros recursos gráficos, e o jQuery fica com a parte mais interativa da aplicação, como animações e efeitos.

3.1.1. Vue.js

Segundo o site oficial do Vue.js, ele é “um *framework* progressivo para a construção de interfaces de usuário” (VUE JS, 2018). A biblioteca central concentra-se apenas na camada de visualização e é fácil de selecionar e integrar com outras bibliotecas ou projetos existentes. Por outro lado, o Vue.js também é capaz de fornecer aplicativos sofisticados de página única quando usado em combinação com ferramentas modernas e bibliotecas de suporte (VUE JS, 2018).

Basicamente, o Vue.js sincroniza uma página da web com JavaScript. Para pessoas já familiarizadas com HTML e CSS, programar com Vue não será um desafio, conhecendo apenas alguns conceitos básicos já é possível codificar um aplicativo que atenda à maioria das necessidades e casos de uso comuns.

Sendo um dos projetos de JavaScript mais alardeados nos últimos anos devido à sua flexibilidade e adoção por comunidades como o Laravel. Sua visão se concentra em primeiro lugar no ideal de adotabilidade incremental, em que Vue.js pode ser usado como uma biblioteca e apenas um decorador de interfaces de usuário ou como uma estrutura completa para arquiteturas altamente opinativas (PRESTON, 2018).

Ou seja, por ser versátil, o Vue.js se adapta bem às necessidades que vão desde pequenos componentes interativos que funcionam bem com o resto do site à um aplicativo web gigantesco com diversos recursos e funcionalidades.

3.1.2. Bootstrap

Bootstrap é um *framework* web com código-fonte aberto para desenvolvimento de componentes de interface e *front-end* para sites e aplicações web usando HTML, CSS e JavaScript, baseado em modelos de design para a tipografia, melhorando a experiência do usuário em um site amigável e responsivo (CORREIA e LEDEL, s.d.).

Seu design responsivo possibilita que uma página web ou aplicativo detecte o tamanho e a orientação da tela do visitante e se adapte automaticamente a exibição de acordo com o dispositivo, como: smartphones, tablets e aplicativos móveis. Além do mais, o Bootstrap inclui componentes de interface de usuário, *layouts* e ferramentas JS junto com sua estrutura para implementação e disponibiliza códigos pré definidos em seu site com diversas funções prontas para serem usadas.

Originalmente criado por um designer e desenvolvedor no Twitter, o Bootstrap se tornou uma das estruturas de *front-end* e projetos de código aberto mais populares do mundo.

Simplificando, Bootstrap é uma coleção enorme de partes reutilizáveis e versáteis de código que são escritas em CSS, HTML e JavaScript. Uma vez que também é um *framework*,

todas as bases já estão estabelecidas para um desenvolvimento web responsivo, e tudo que os desenvolvedores precisam fazer é inserir o código no seu sistema e adaptá-los às suas necessidades.

3.1.3. jQuery

jQuery é uma biblioteca JavaScript rápida, pequena e rica em recursos. Isso torna as coisas como a passagem e manipulação de documentos HTML, manipulação de eventos, animação e Ajax mais simples com uma API fácil de usar e que funciona em vários navegadores. Com uma combinação de versatilidade e extensibilidade, o jQuery muda a maneira de como as pessoas codificam em JavaScript (JQUERY, s.d.).

O design da biblioteca se presta a um início rápido para designers com pouca experiência em programação, uma vez que sua estrutura é de fácil entendimento e foi feita para ser compreendida por programadores com pouca experiência.

3.2. Métodos

O desenvolvimento deste trabalho foi realizado e dividido em etapas que se iniciou com a análise de requisitos juntamente com o especialista de domínio, seguido pelo estudo de conceitos e dos módulos da primeira versão do OSLive, planejamento da aplicação levando em consideração a interface padrão atual do OSLive, desenvolvimento do software e testes funcionais. A figura 7 demonstra as etapas supracitadas.

Figura 5. Metodologia para a realização do trabalho

Na primeira etapa (Figura 7 - 1), intitulada Reuniões com Especialista, foi delineado juntamente com o especialista de domínio os principais aspectos funcionais do trabalho, os objetivos gerais e específicos e as medidas a serem adotadas para o desenvolvimento e atualização do módulo de segmentação do OSLive. Além disso, as especificações e mudanças a serem executadas no novo módulo de exercícios do OSLive como: implementação de uma interface padrão, correção de bugs, correção da coluna “Limite” da tabela de segmentos e inserção de novos botões e novas funcionalidades.

A segunda etapa (2), Estudo dos Conceitos, foi realizada através de pesquisas bibliográficas, estudo de livros e periódicos encontrados em bibliotecas virtuais, revistas científicas e teses de doutorado. As pesquisas tiveram enfoque, principalmente, nos conceitos gerais de SO e de como ele gerencia a memória por meio da segmentação.

A etapa seguinte (3), Estudo da Primeira Versão do OSlive, constitui em estudar o código e a lógica de programação utilizada na primeira versão do módulo do OSlive, esse estudo serviu para manter a estrutura principal do mesmo no que diz respeito à funções e variáveis, de modo que, mesmo após o incremento de novas funcionalidades, a estrutura continue a mesma facilitando assim a manutenção e possíveis ajustes futuros. Além disso, o estudo da primeira versão não só do módulo de segmentação, como também do de paginação, foi determinante para programar a interface padrão do presente projeto.

O (4) Planejamento da Aplicação, por sua vez, consistiu no planejamento de como seria desenvolvido o módulo de resolução de exercícios; seu *layout*, lógica de programação, interface, interação com o usuário e partes mais técnicas do processo de desenvolvimento. Por fim, na fase final dos métodos (5), Implementação, foi realizada de fato a programação do módulo, com codificações gráficas utilizando *Bootstrap* e de funcionalidades que possibilitam a interação do usuário com a aplicação, seja por meio de *inputs* ou a forma com que o usuário visualiza o sistema.

A medida que o sistema foi sendo desenvolvido, Testes (6) funcionais foram executados a fim de verificar o funcionamento da aplicação, seguindo uma metodologia de desenvolvimento iterativa e incremental onde a medida que novos recursos foram adicionados, testes e correções foram aplicadas para garantir a consistência e funcionamento da aplicação como um todo.

4. Resultados e discussões

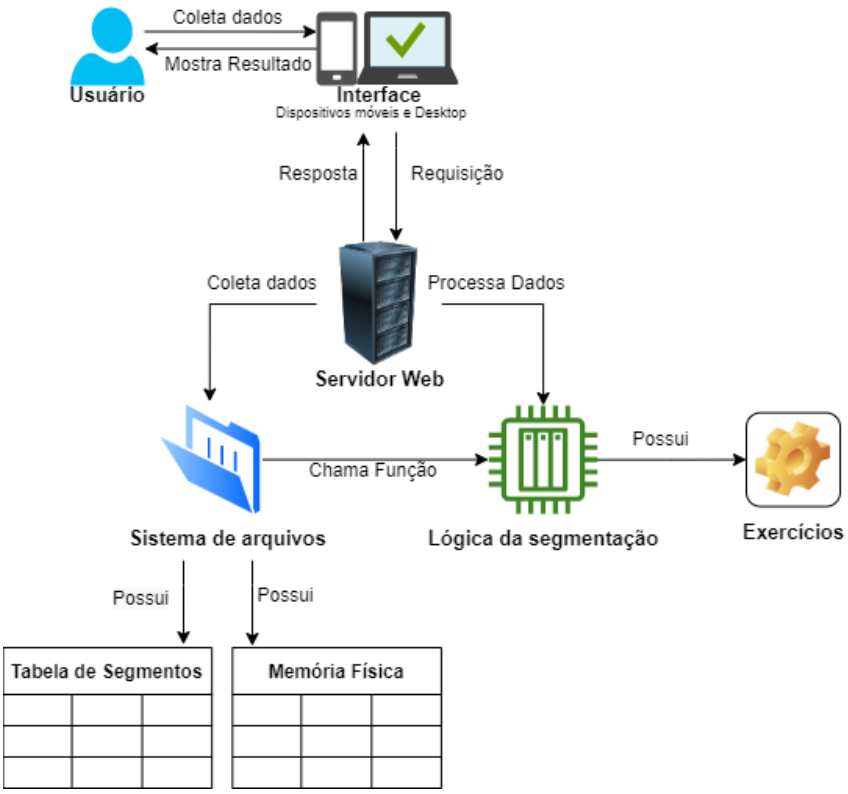
O trabalho consistiu no desenvolvimento de um novo módulo para o OSLive, que é a resolução de exercícios sobre o mecanismo de gerência de memória por segmentação, criado a partir do módulo de simulação de gerência de memória por segmentação já presente no OSLive. Com o intuito de otimizar o aprendizado dos alunos, foram criados exercícios que possibilitam a entrada de dados do usuário, a fim de compará-las com dados (respostas) já pré definidas, já que, atualmente, na versão de simulação, só é possível simular a gerência de memória, não sendo possível interagir com o sistema.

Esta seção irá apresentar como foi feito o desenvolvimento das funcionalidades, as camadas, arquitetura e interface da aplicação. Além disso, delinear como foi feita a aplicação, bem como os exemplos dos exercícios supracitados.

4.1. Arquitetura da aplicação

A figura 6 exemplifica a arquitetura de funcionamento do novo módulo de gerência de memória por segmentação. Para desenvolvê-lo foi necessário realizar alterações no Sistema de Arquivos e Lógica da Segmentação da primeira versão, que inclui desde a interface à lógica de programação do mesmo.

Figura 6. Arquitetura do módulo de segmentação



De acordo com a figura acima, a aplicação foi desenvolvida seguindo padrões de desenvolvimento *web* onde são suportados diferentes tipos de clientes como *desktop* e *mobile*, sendo que o cliente faz requisição diretamente com o servidor web. Esse tipo de arquitetura torna os testes funcionais simples já que toda aplicação está integrada, facilitando assim testes de integração e *end-to-end*.

Na figura supracitada, o usuário coleta dados da interface que por sua vez faz requisições diretamente com o servidor web. Dentro do servidor encontra-se o novo módulo de segmentação que é dividido em duas partes; Sistema de Arquivos e Lógica da Segmentação. O primeiro diz respeito aos pacotes Bootstrap, arquivos CSS, HTML e afins e é a parte gráfica da aplicação que representa a gerência de memória do SO. Já na Lógica da Segmentação, ficam os *scripts* que realizam a lógica por trás da segmentação de memória do SO como, Typescript e Jquery.

4.2. Módulo de simulação do mecanismo de segmentação

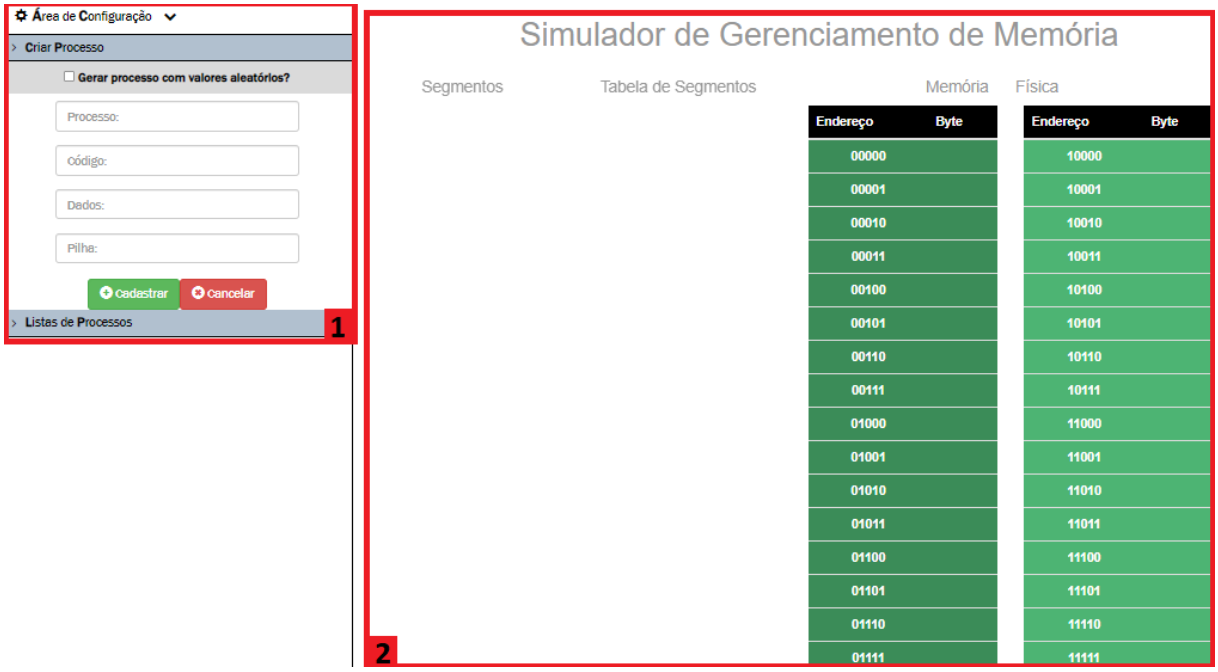
O módulo de gerência de memória por segmentação oferece um ambiente gráfico responsivo que se adapta ao tamanho de tela de qualquer computador ou *smartphone* e foi

projetado visando evitar poluição visual, apresentando apenas os elementos fundamentais com clareza e objetividade.

A tela inicial do ambiente da simulação do mecanismo de segmentação é apresentada na figura 7. Na área de configuração (retângulo nº 1) é possível criar um ou mais processos que serão segmentados em código, dado, e pilha. A representação dos segmentos foi feita desta forma porque geralmente o sistema operacional separa os programas de acordo com a sua estrutura funcional que geralmente são; rotinas, módulos, códigos, dados e pilhas.

A criação de processos pode ser feita de modo aleatório ou inserindo manualmente o tamanho do *byte* de cada segmento. Na área de processos (retângulo nº 2), serão apresentados os segmentos, a tabela de segmentos e a tabela de memória física. Os Segmentos representam o Código, Dado e Pilha criados na área de configuração, cada um com o seu respectivo deslocamento; a Tabela de Segmentos serve para traduzir os endereços lógicos em físicos de acordo com a base e limite; já a Memória Física mostra onde cada byte dos segmentos está armazenado dentro dela.

Figura 7. Módulo de Gerência de Memória do OSLive



Para criar um novo processo, é preciso informar a identificação do processo (uma letra), o código, dado e pilha que representam partes de um programa padrão a ser segmentado. Ao inserir os dados são criados segmentos, cada um com seu respectivo endereçamento, tanto da memória física, quanto da lógica e, em *Kbytes*, propostos na inserção, como mostra a figura 8.

Figura 8. Criação de processos

A figura 8 mostra o ambiente após a criação do processo “A”. Ele foi dividido em três segmentos com seus respectivos deslocamentos e *bytes*; tabela de segmentos que indica a base e limite de cada segmento e o endereçamento da memória física. Cada quadro numerado representa um conceito, sendo eles:

1. Na área 1 é possível visualizar a ordem de criação dos processos, cada processo recebe uma letra de A a F, esse é o limite de processos que podem ser criados;
2. A área 2 mostra o tamanho de cada segmento, o deslocamento e os bytes. Além disso, indica se o segmento é código, dado ou pilha;
3. A área 3 apresenta o número do segmento, a base e o limite. Com essas informações é possível localizar cada segmento do processo que é apresentado na memória física (quadro 4);
4. A área 4 é a representação da memória física, que por sua vez armazena o processo e à medida que os segmentos vão sendo carregados, as cores de seus respectivos segmentos são apresentados.

Seguindo a ideia de desenvolver uma aplicação voltada para o ensino por meio de simulações didáticas, o presente trabalho levou em consideração aspectos já desenvolvidos anteriormente no módulo de simulação, estes supracitados, como; lógica de implementação,

código fonte e arquitetura. Na verdade, não teria sido possível desenvolvê-lo sem a ajuda das funcionalidades já criadas anteriormente, com isso, a seção a seguir abordará o que de fato foi construído ao longo do semestre

4.3. Módulo de Resolução de Exercícios

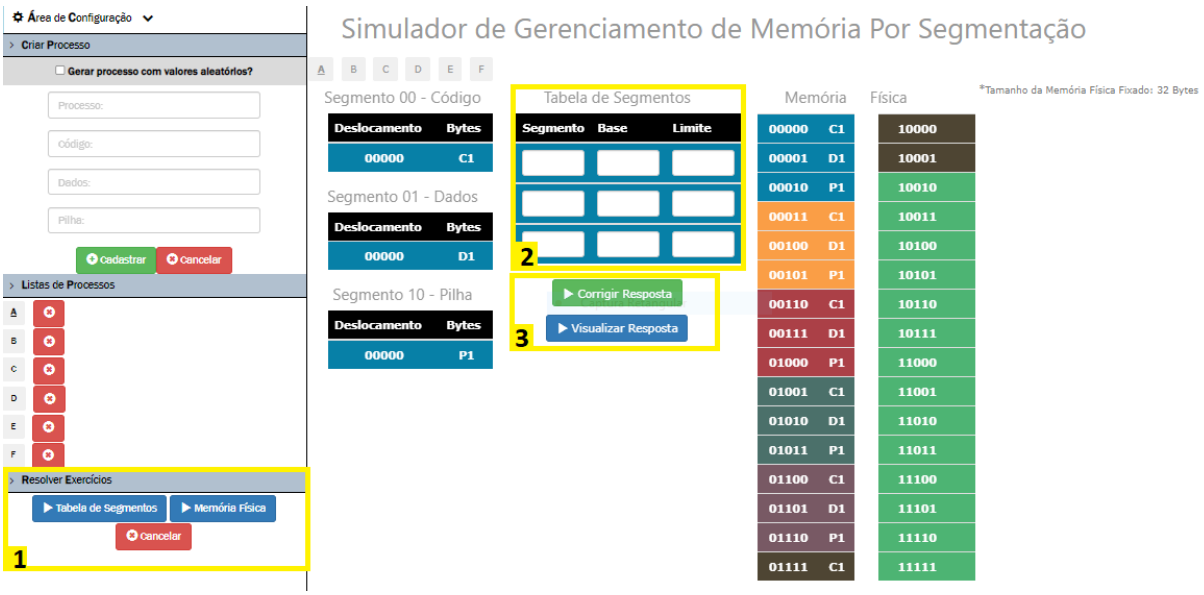
A interface de exercício de simulação do processo de gerência de memória por segmentação é apresentada na figura 9. Ela é praticamente a mesma apresentada na seção 4.1, as principais diferenças estão relacionadas às cores das tabelas, o número de processos que podem ser criados e a forma com que os dados são apresentados.

Cada cor apresentada na tabela está relacionada com um processo específico, por exemplo, o processo A sempre será representado nas tabelas com a cor azul, assim como o processo F na cor marrom, essa ideia de utilizar cores fixas para cada processo foi escolhida a fim de facilitar o correlacionamento dos dados da tabela, logo então, ajudando o usuário a responder os exercícios visto que será mais fácil localizar os processos através das cores

Vale notar que existe um número máximo de processos que podem ser criados, eles podem ir de A a F, ao atingir a última letra, a sequência contínua para a próxima letra disponível depois de A e antes de F, isso levando em consideração se o usuário opte por gerar os processos automaticamente.

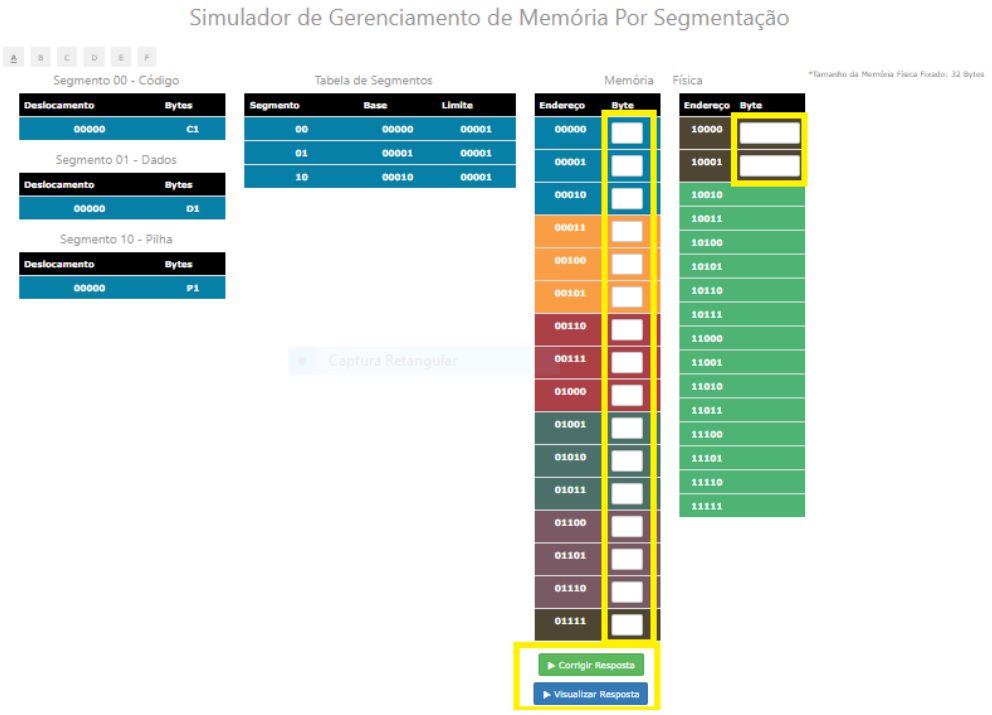
Outra diferença entre o primeiro módulo e este, diz respeito a forma com que os dados são apresentados, na área de configuração foi acrescentado uma nova aba chamada Resolver Exercícios, nela o usuário poderá escolher entre responder exercícios na Tabela de Segmentos ou na Tabela de Memória Física, ao selecionar um dos dois, aparecerá *inputs* para o preenchimento de dados dentro da tabela onde os botões Corrigir Resposta e Visualizar Resposta aparecerão logo abaixo da tabela selecionada. Como mostra a figura 9.

Figura 9. Interface exercício Tabela de Segmentos



Na figura 9 é apresentada a tela em que o usuário selecionou preencher a “Tabela de segmentos”, ação feita no retângulo 1. Logo, são apresentados espaços em branco dentro da tabela para que o usuário possa inserir sua resposta, como mostra o retângulo de número 2, além disso, os botões Corrigir resposta e Visualizar resposta, só poderão ser exibidos na área do exercício selecionado, neste caso, as ações de corrigir resposta e visualizar resposta só terão efeito na tabela de segmentos, como mostra o retângulo 3. A figura 10 mostra como seria caso o usuário selecionasse Memória física na área de configuração ao invés de Tabela de segmentos.

Figura 10. Interface exercício de Memória física



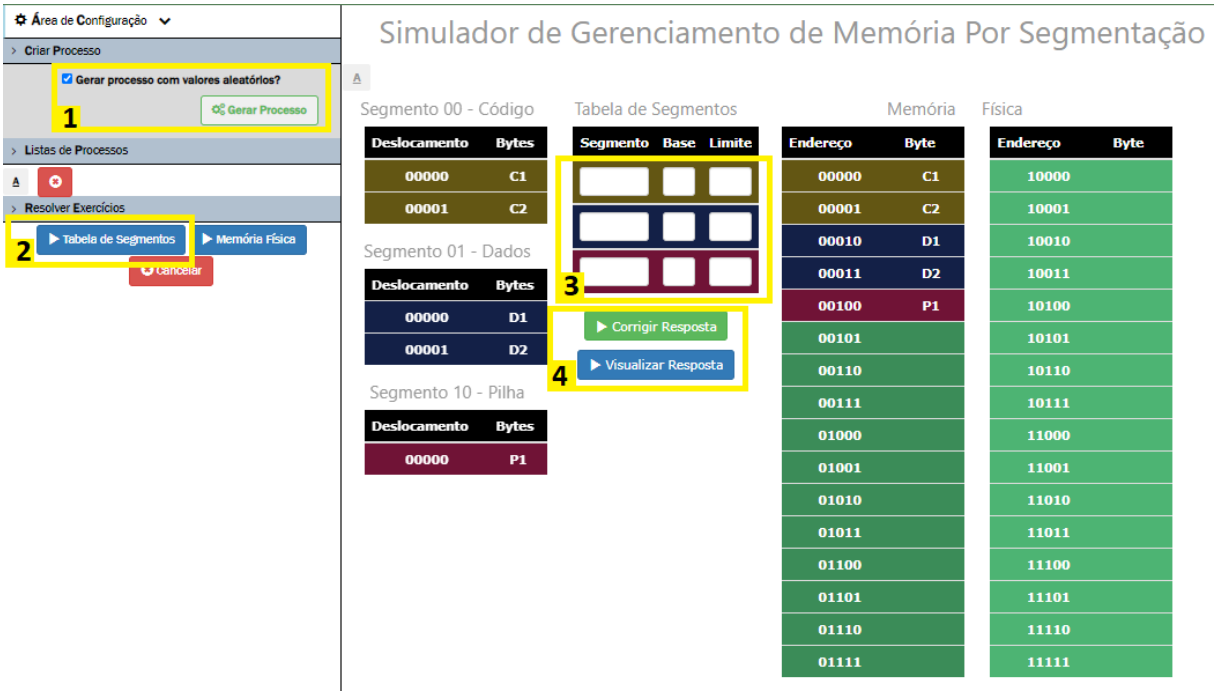
Como foi explicado anteriormente, os botões referentes ao exercício agora só podem ser executados na tabela de memória física, visto que o botão “Memória Física” foi selecionado na área de configuração. Desta forma, facilita para o entendimento e usabilidade do usuário no que diz respeito ao preenchimento de respostas dos exercícios, pois fornece mais clareza e agilidade para tal.

4.4. Execução dos exercícios

Esta seção apresentará exemplos de execução de exercícios no módulo de gerência de memória por segmentação. Para a demonstração, os processos foram criados aleatoriamente no campo de configuração. Após a criação dos processos, é preciso selecionar qual dos exercícios será respondido na área de “Resolver Exercícios”, retângulo verde, sendo eles: Tabela de Segmentos e Memória Física. Será explicado a resolução de cada um respectivamente.

A figura 11 mostra como é realizado o exercício de Tabela de segmentos.

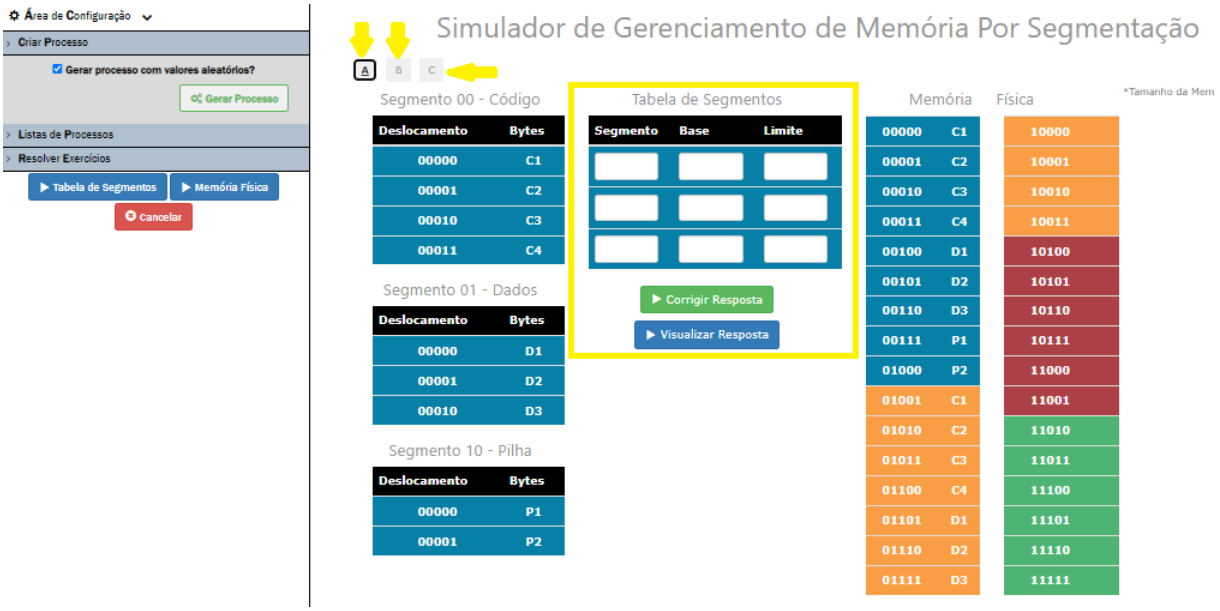
Figura 11. Execução de Exercício Tabela de Segmentos



A Figura 11 apresenta a tela de simulação de exercício do mecanismo de segmentação para o preenchimento da tabela de segmentos, conforme descrição a seguir.

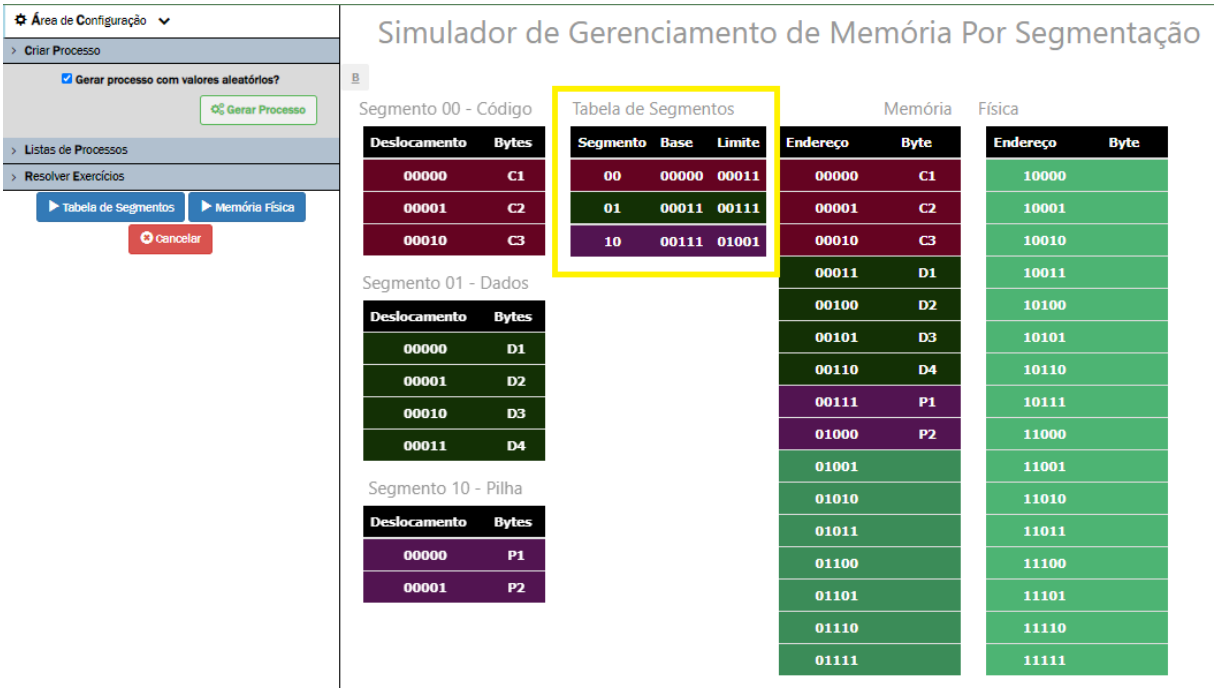
- A área de número 1 possibilita ao usuário criar processos de forma aleatória, caso ele não esteja interessado em processos de nome e tamanhos específicos;
- A área 2 permite ao usuário selecionar qual exercício ele pretende responder como também cancelar a simulação, sendo que, ao ser cancelada, a página é atualizada;
- A área 3 encontra-se o formulário para a resolução de exercícios; ela será preenchida de acordo com o limite da memória lógica;
- No número 4 é possível selecionar Corrigir Resposta e Visualizar Resposta; a figura 12 mostra o que acontece quando a primeira opção é selecionada.
- Vale notar que é possível responder mais de um processo ao mesmo tempo, por exemplo, caso o usuário crie 2 ou mais processos ele poderá selecionar em qual será feito a resolução de exercícios. As setas da figura 12 demonstra essa funcionalidade.

Figura 12. Correção de resposta da Tabela de Segmentos e Processos



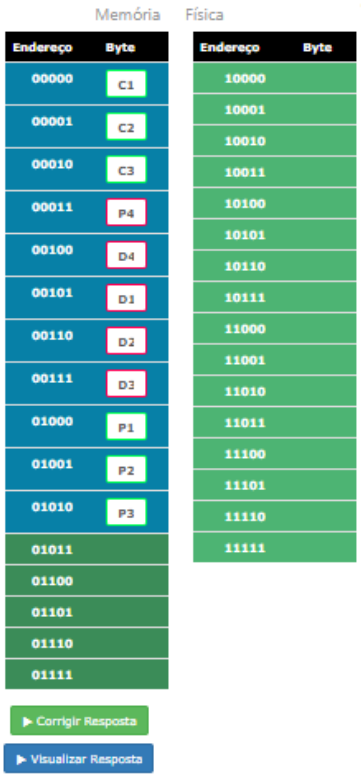
Como mostra o retângulo da figura 12, ao preencher a tabela de segmentos e escolher corrigir a resposta, os campos de entrada mudam de cor de acordo com a resposta, os campos delineados em vermelho representam as respostas erradas enquanto as verdes, respostas corretas. Também é possível visualizar as respostas, clicando no botão azul.

Figura 13. Visualizar resposta da Tabela de Segmentos



Ao selecionar Corrigir Resposta, a tabela de segmentos é preenchida por completo, sendo possível visualizar o segmento, a base e o limite do mesmo. Para resolver exercícios com a tabela de Memória Física basta seguir os mesmos passos anteriores, criação de processos, seleção de exercícios e correção ou visualização de resposta. A figura 14 mostra como é feita a resolução com a tabela de memória física.

Figura 14. Execução de Exercício Tabela de Memória Física



Assim como no primeiro exercício, ao selecionar Corrigir Resposta, é demarcado onde o usuário acertou e onde errou. Vale notar que aqui a tabela não possui cores para que não haja referência com as cores da tabela de segmentos. Para visualizar a resposta também não é diferente; a imagem abaixo ilustra quando o botão azul de visualizar resposta é clicado.

Figura 15. Visualizar resposta Tabela de Memória Física

Ao selecionar Visualizar Resposta já é possível visualizar a cor de cada segmento, o endereço e o byte, além de possibilitar a visualização de respostas tanto da tabela de segmentos quanto de memória física, essa funcionalidade também fornece maior entendimento para que o usuário compare as respostas e entenda onde errou.

De acordo com os exemplos supracitados, acrescentar um novo módulo de segmentação no OSLive possibilita maior entendimento e aprendizado para o usuário através da resolução de exercícios que até então não era possível, deste modo, por meio da simulação e prática o aluno poderá reforçar o que foi apresentado em sala de aula.

5. Considerações Finais

O trabalho apresentado teve como objetivo buscar desenvolver um novo módulo de simulação do mecanismo de segmentação de memória com uma nova interface e funcionalidades como a inserção da opção de resolver exercícios na tabela de segmentos e memória física, além de possibilitar a correção e visualização de resposta dos exercícios. Todos os objetivos foram concluídos e tornou-se válida a hipótese levantada que teve como princípio desenvolver um novo módulo web para o OSLive com tecnologias web HTML, CSS, Javascript e afins.

A ferramenta foi desenvolvida levando em consideração a primeira versão de segmentação e de outros módulos do OSLive, espera-se que através desta nova versão o usuária possa interagir mais com o sistema de forma que facilite o entendimento do conteúdo apresentado em aula. A partir da entrada de dados do usuário foi efetivado o funcionamento da aplicação e padronização das cores das tabelas, visto que anteriormente isso não estava presente.

Além disso, algumas alterações foram necessárias para melhor funcionamento do sistema, dentre eles: mudança no padrão de cores das tabelas; correção de bug na área de configuração onde não era possível alterar os elementos colapsáveis para mostrá-los ou ocultá-los; mudança do título do módulo; definição do limite máximo de processos que podem ser criados, sendo de A a F, cada um com sua respectiva cor; correção do valor do limite na tabela de segmentos; e apresentação do tamanho da memória física em bytes.

Dito isto, para que o projeto se torne ainda melhor, sugere-se em trabalhos futuros a inserção de dicas que explicam o funcionamento do sistema e de como é feito o cálculo das informações apresentadas dentro da tabela, por exemplo, um botão “?” que forneça uma explicação clara e objetiva de como o sistema operacional realiza comparações entre o deslocamento fornecido e o limite do segmento para encontrar o endereço físico como também explicações dos conceitos básicos da segmentação do SO.

Por fim, espera-se que o presente trabalho tenha contribuído para a melhoria do OSLive como também para o auxílio e aprendizagem dos conceitos de gerência de memória por segmentação do sistema operacional.

Referências

BOOTSTRAP. **Introdução Bootstrap**. Disponível em: https://getbootstrap.com/docs/4.0/getting-started/introduction/. Acesso em: 01 dez. 2021

CORREIA, Samuel Tolentino; LEDEL, Leandro Camara. **Sistema de Software para Rastreabilidade de Medicamentos**. 2019. 20 f. TCC (Graduação) - Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Ifsp, São Paulo, 2019.

CHAFFER, Jonathan; SWEDBERG, Karl. **Learning Jquery**. 4. ed. S.I: Packt Publishing, 2013.

DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R.. **Sistemas Operacionais**. 3. ed. S.I: Pearson Prentice Hall, 2005..

GIRAFFA, Lucia Maria Martins; VICCARI, Rosa Maria. Estratégias de Ensino em Sistemas Tutores Inteligentes Modelados através da Tecnologia de Agentes. **Revista Brasileira de Informática na Educação (Brazilian Journal Of Computers In Education)**, S.I, v. 5, n. 3, p. 3-6, maio 1999. Disponível em: https://br-ie.org/pub/index.php/rbie/article/view/2276. Acesso em: 01 dez. 2021.

Martinez, W. L. (2011). **Graphical user interfaces**. Wiley Interdisciplinary Reviews: Computational Statistics, 2(3), 119-133. doi:10.1002/wics.150

MAIA, Luiz Paulo. **SOsim:** SIMULADOR PARA O ENSINO DE SISTEMAS OPERACIONAIS. 2001. 81 f. Tese (Doutorado) - Curso de Ciências em Informática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2001.

MARTINS, Tallyson Ruiter Andrade. **IMPLEMENTAÇÃO DO MÓDULO DE SIMULAÇÃO DO MECANISMO DE SEGMENTAÇÃO DO OSLIVE**. 2019. 28 f. TCC (Graduação) - Curso de Sistemas de Informação, Ceulp-Ulbra, Palmas, 2019.

MAZIERO, **Carlos Alberto. Sistemas Operacionais: Conceitos e Mecanismos**. Curitiba: Dinf, 2019. 456 p. Disponível em: https://www.researchgate.net/profile/Carlos-

Maziero/publication/343921399_Sistemas_Operacionais_Conceitos_e_Mecanismos/links/5f4817b2a6fdcc14c5d3b433/Sistemas-Operacionais-Conceitos-e-Mecanismos.pdf. Acesso em: 01 dez. 2021.

MAZIERO, Carlos Alberto. **Sistemas Operacionais V** - Gerência de Memória. 2011. 54 f. Dissertação (Mestrado) - Curso de Ciências da Computação, Utfpr, Curitiba, 2011. Disponível em: http://files.francielamorim.webnode.com.br/200000054-2961c2a5bf/SisOper%2005%20-%20Gerencia%20Memoria.pdf. Acesso em: 01 dez. 2021.

M. ZEM-LOPES, A.; Z. PEDRO, L.; ISOTANI, S. Qualidade de Softwares Educacionais Baseados na Web (Semântica): Um Mapeamento Sistemático. **RENOTE**, Porto Alegre, v. 12, n. 1, 2014. DOI: 10.22456/1679-1916.50335. Disponível em: https://www.seer.ufrgs.br/index.php/renote/article/view/50335. Acesso em: 29 jun. 2022.

JQUERY. **What is jQuery?** Disponível em: https://jquery.com/. Acesso em: 01 dez. 2021.

OLIVEIRA, Ramon Almeida; SOUZA, Antonio Carlos dos Santos. **SWSO** - Simulador Web de Sistemas Operacionais. 2014. 18 f. TCC (Graduação) - Curso de Sistemas de Informação, Instituto Federal da Bahia, S.I, 2014.

OLIVEIRA, Rômulo Silva; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. **Sistemas Operacionais**, Porto Alegre, 2001. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/19242/000102159.pdf?sequence=1&isAllowed=y. Acesso em: 28 jul. 2021.

OLIVEIRA, Rômulo Silva de, et al. Sistemas Operacionais, S.I, v. 3, n. 1, p. 9-9, nov. 2001. Disponível em: http://www.romulosilvadeoliveira.eng.br/artigos/Romulo-Carissimi-Simao-Rita2001.pdf. Acesso em: 21 fev. 2022.

PAZIN FILHO, Antonio; SCARPELINI, Sandro. SIMULAÇÃO: DEFINIÇÃO. **Portal de Revistas da Usp**, Ribeirão Preto, v. 52, n. 1, p. 5-6, fev. 207. Disponível em: https://www.revistas.usp.br/rmrp/article/view/312/313. Acesso em: 01 dez. 2021.

QINGQIANG, Wu; LANGCAI, Cao. **Teaching Mode of Operating System Course for Undergraduates Majoring in Computer Sciences**. 2019. 4 f. Dissertação (Mestrado) - Curso de Information Science, Xiamen University, Xiamen, 2009. Disponível em: https://sci-hub.mksa.top/10.1109/iccse.2009.5228196. Acesso em: 01 dez. 2021.

SMITH, Roger D.. **Simulation Article**. 4. ed. New York: Model Benders, 2000. Disponível em: http://www.modelbenders.com/encyclopedia/encyclopedia.html. Acesso em: 01 dez. 2021.

SENGER, Hermes; GUARDIA, Hélio Crestana. **Gerência de Memória**. 2013. 14 f. TCC (Graduação) - Curso de Sistemas de Informação, Ufsc, Sergipe, 2013. Disponível em: http://livresaber.sead.ufscar.br:8080/jspui/bitstream/123456789/2462/1/SO-AT4.1-Gerencia_de_memoria.pdf. Acesso em: 01 dez. 2021.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Fundamentos de Sistemas Operacioanis**. 9. ed. Rio de Janeiro: Abdr, 2015.

TANENBAUM, Andrew S.; BOS, Herbert. **MODERN OPERATING SYSTEMS**. 4. ed. New Jersey: Pearson, 2015. Disponível em: http://lib.bvu.edu.vn/bitstream/TVDHBRVT/19439/1/Modern-Operatin-systems.pdf. Acesso em: 01 dez. 2021.

TANENBAUM, Andrews. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson Education do Brasil, 2010.

TANENBAUM, A. S.; WOODHULL, A. S. **Sistemas Operacionais:projeto e implementação**. Porto Alegre: Bookman, 2008.

VUE.JS. **Introdução Vue.js**. Disponível em: https://br.vuejs.org/v2/guide/index.html. Acesso em: 01 dez. 2021.