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
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Wllynilson Pereira Cardoso Carneiro

IMPLEMENTAÇÃO DO MÓDULO PARA SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA DO OSLIVE

Palmas – TO

2019

Wllynilson Pereira Cardoso Carneiro

IMPLEMENTAÇÃO DO MÓDULO PARA SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA DO OSLIVE

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Madianita Bogo Marioti

Palmas – TO

2019

Wllynilson Pereira Cardoso Carneiro

IMPLEMENTAÇÃO DO MÓDULO PARA SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA DO OSLIVE

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Madianita Bogo Marioti.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Madianita Bogo Marioti

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Heloise Acco Tives Leão

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2019

AGRADECIMENTOS

A Deus e a minha família que, durante toda essa caminhada, não mediram esforços para me apoiar. A minha querida esposa Erlângela que sempre entendeu minha ausência, em momentos importantes, para me dedicar aos estudos, ao meu filho Lorenzo Will, hoje com quatro anos de idade. Aos meus pais Nilson e Lucinea que me educaram e sempre estiveram presentes e a minha irmã Rhitta de Khássia.

A todos os meus professores da graduação: Cristina Filipakis, Fabiano Fagundes, Fernando Luiz, Parcilene Fernandes, Jackson Gomes, William Christhie, Heloise Acco, Fábio Castro, Diemy, Reginaldo, Celso, Cíntia, que além de proporcionar o conhecimento sempre que foi necessário estiveram à disposição.

A minha orientadora Madianita Bogo que esteve sempre disponível para tirar dúvidas, esclarecimentos e quaisquer outros assuntos referentes a esse trabalho. Agradeço imensamente todo o seu apoio, paciência e principalmente todos os ensinamentos.

Aos meus amigos Ewerton Santiago e Gustavo Siqueira.

RESUMO

CARNEIRO, Wllynilson Pereira Cardoso. **IMPLEMENTAÇÃO DO MÓDULO PARA SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA DO OSLIVE** 2019. Trabalho de Conclusão de Curso (Graduação) - Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2019.

A disciplina Sistemas Operacionais, obrigatória nas matrizes curriculares dos cursos superiores da área de Computação, apresenta um conjunto de conteúdos complexos e abstratos, o que dificulta o aprendizado. Inserido nesse contexto, está sendo desenvolvido o OSLive no CEULP/ULBRA, que é um ambiente web que oferece recursos para auxiliar os alunos a compreenderem os conceitos vistos em sala de aula, por meio de simulações, tutoriais e exercícios. Uma das áreas do Sistema Operacional é a gerência de memória que coordena o uso da memória física (RAM). Dentre os vários mecanismos existentes na RAM há o mecanismo de paginação por demanda. O objetivo deste trabalho é desenvolver um módulo da plataforma *online* OSLive que consiste numa aplicação que ofereça a simulação do mecanismo de paginação por demanda, de modo a auxiliar os alunos a compreenderem este conteúdo.

Palavras-chave: Sistemas Operacionais, Gerência de Memória, Paginação por Demanda, simulação.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Entrada e Saída ..... E/S

Gerência de Processos ..... GP

Sistemas Operacionais ..... SO

Paginação por Demanda ..... PD

Visual Studio Code ..... VSCODE

Unidade Central de Processamento ..... CPU

Memória de Acesso Aleatório (Randon Access Memory) ..... RAM

Unidade de Gerenciamento de Memória (Memory Management Unit) …….. MMU

Grupo de Estudos em Novas Tecnologias para processos de Ensino e Aprendizagem...GENTE

SUMÁRIO

1 INTRODUÇÃO

Um Sistema Operacional que, de acordo com Oliveira, Carissimi e Toscani (2001), "é uma camada de software colocada entre o *hardware* e os programas que executam tarefas para o usuário", é fundamental para o funcionamento dos Sistemas de Computação. Um SO possibilita que os usuários utilizem os recursos que o computador oferece e entender seu funcionamento é essencial para quem trabalha na área de Computação. Assim, a disciplina de SO deve estar presente na matriz curricular dos cursos superiores da área de computação e informática no Brasil, de acordo com o Ministério da Educação - MEC (BRASIL, 2012).

A disciplina de SO abrange as quatro áreas de atuação de um SO: gerência de arquivos (GA), gerência de entradas e saídas (E/S), gerência de memória (GM) e gerência de processos (GP).

A Gerência de Memória, de acordo com Oliveira, Carissimi e Toscani (2001) "tem função de prover os mecanismos necessários para que os diversos processos compartilhem a memória de forma segura e eficiente". Na Gerência de Memória, o SO, entre outras coisas, controla os endereços de memória RAM (*Random Access Memory*) que estão em uso, aloca memória quando solicitada por um processo e libera o acesso quando o processo finaliza a sua execução.

Existem vários mecanismos de gerência de memória, entre eles, a paginação por demanda (PD), sendo que segundo Oliveira, Carissimi e Toscani (2001), " na paginação um programa é carregado página a página". Cada página lógica de um processo ocupa uma página física na memória física. No entanto, esta área ocupada não precisa ser contígua.

A paginação por demanda, de acordo com DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R. Choffnes(1999), "garante que o sistema traga para a memória principal somente as páginas de que o processo realmente necessita, o que potencialmente permite que mais processos ocupem a memória principal". Ou seja, os conteúdos necessários aos processos são carregados para a memória física quando efetivamente utilizados, de acordo com a demanda.

A execução do mecanismo de Paginação por Demanda pelos Sistemas Operacionais é realizada sem a visualização e percepção do usuário, de forma que o aluno da disciplina de SO acaba tendo que entender o conteúdo mesmo sem a visualização do seu funcionamento. Uma maneira interessante de resumir e sintetizar um conceito é por meio da simulação.

Nesse contexto, o objetivo deste trabalho é a implementação de uma aplicação web responsiva, que realizará a simulação do funcionamento do mecanismo de PD. Essa aplicação será um módulo do OSLive, que é uma plataforma online projetada para apoio ao processo de ensino e aprendizagem da disciplina de Sistemas Operacionais, oferecendo recursos que visam auxiliar os alunos a compreenderem os conceitos vistos em sala de aula, por meio de simulações, tutoriais e exercícios. Este é um projeto 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.

Assim, para alcançar o objetivo deste trabalho, buscou-se responder ao seguinte problema de pesquisa: como simular a paginação por demanda de forma a auxiliar no aprendizado deste conteúdo na disciplina de SO?

A partir do problema apresentado, é possível considerar a seguinte hipótese de que, usando as tecnologias Angular, Bootstrap e TypeScript é possível criar um simulador que auxiliará no aprendizado da paginação por demanda. Dessa forma, esse trabalho propõe como objetivo geral, o desenvolvimento de um aplicativo web que seja capaz de realizar a simulação do mecanismo de paginação por demanda (PD), que é subdividido nos seguintes objetivos específicos:

- criar a representação gráfica do funcionamento do mecanismo de PD;
- projetar as telas com representação gráfica do mecanismo de PD;
- implementar a interface gráfica para a simulação da PD, em uma ambiente web responsivo;
- realizar e apresentar os testes funcionais.

Esse trabalho apresenta, na seção 2, o Referencial Teórico, que é dividido entre Sistemas Operacionais, Gerência de Memória, Paginação Simples, Memória Virtual e Paginação por Demanda. Na seção 3 é apresentada a Metodologia que é dividida em Desenho de Estudo e Materiais. Na seção 4 é apresentado os Resultados e Discussão.

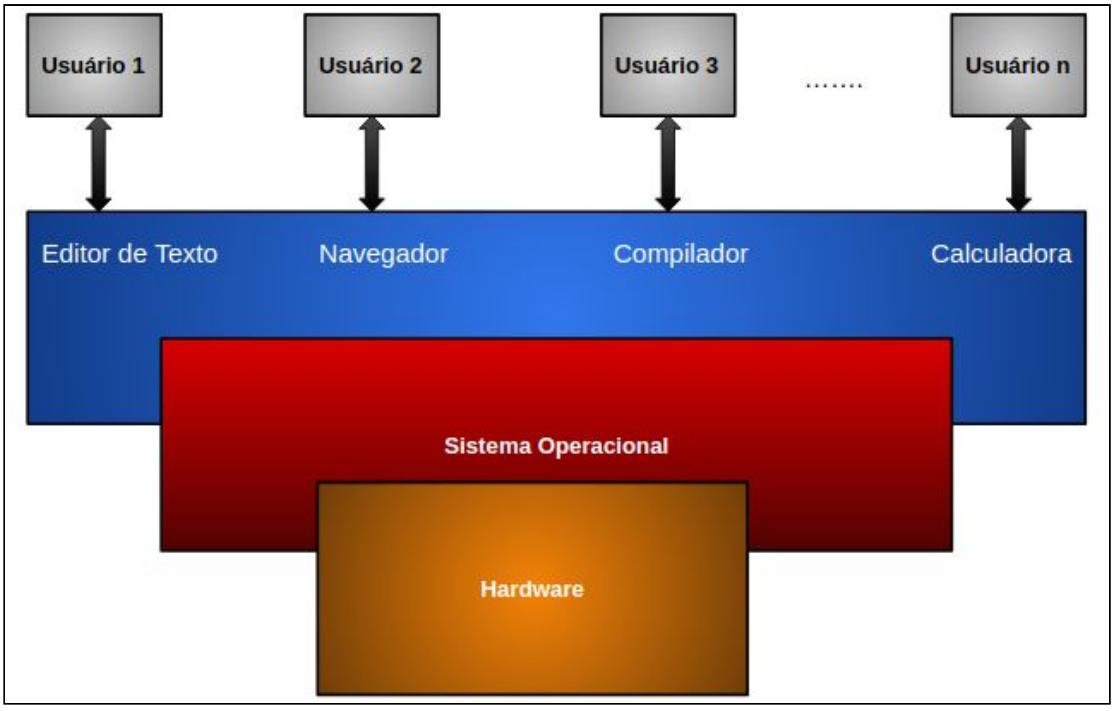
2 REFERENCIAL TEÓRICO

Esta seção apresenta uma visão geral do Sistema Operacional e suas quatro áreas: gerência de arquivos, gerência de entrada e saída, gerência de memória e gerência de processos. Em seguida, apresenta mais detalhadamente o mecanismo de paginação por demanda, que é o foco deste trabalho.

2.1 Sistema Operacional

O Sistema Operacional (SO) fornece a interação entre homem e máquina, atuando como um gerenciador que fica em uma camada entre os aplicativos de usuário e o *hardware* do computador. Um Sistema Operacional, segundo Silberschatz, Galvin e Gagne. (2013, p. 3), "fornece uma base para os programas aplicativos e atua como intermediário entre o usuário e o *hardware* do computador". A figura 1 demonstra o funcionamento de um sistema da computação clássica.

Figura 1. Representação do sistema operacional (adaptada)



Fonte: Oliveira, Carissimi e Toscani, 2010

A figura 1 mostra a interação entre o SO e os usuários. Enquanto diversos usuários fazem uso dos *softwares* instalados, o sistema operacional atua como intermediário, ocultando os acessos ao *hardware*.

Os Sistemas Operacionais atuais são multitarefas, ou seja, conseguem realizar a execução simultânea de vários programas, entre aplicativos de usuários e programas de sistema. A execução simultânea aumenta a eficiência do SO, pois dá a ideia de paralelismo.

Em sistemas multiprogramados, um usuário, por exemplo, pode abrir um editor de texto e ao mesmo tempo usar o navegador para acessar o portal da faculdade, enquanto outro usuário, pode realizar a edição de vídeo ao mesmo tempo em que o player de música é executado.

Para realizar o gerenciamento dos recursos, o SO divide as suas atividades em quatro áreas:

- **Gerência de arquivos:** a gerência de arquivos é a camada mais próxima do usuário, principalmente, porque o usuário realiza operações de escrita e leitura com interface gráfica mais amigável através de diretórios e coleção de arquivos. A gerência de arquivos é a parte mais visível do sistema operacional para o usuário pois ele está sempre manipulando arquivos, seja para criar ou editar seus documentos, ou seja executando programas, que são arquivos, no computador (SIQUEIRA, 2018).
- **Gerência de Processos:** um processo é uma entidade abstrata, formada pelos recursos de hardware, pela execução das instruções referentes a um algoritmo (RIBEIRO, 2005). O SO trabalha com vários processos simultâneos e cabe ao SO o gerenciamento de todos. De acordo com Ribeiro (2005) os processos são programas carregados na memória do computador e podem ser classificados como leves e pesados. Como vários processos são carregados simultaneamente no sistema, cabe ao SO o gerenciamento de todos, coordenando o acesso à CPU e sincronizando as ações dos processos.
- **Gerência de Entrada / Saída (E/S):** a gerência de entrada e saída fornece uma interface para que qualquer dispositivo, independente de sua tecnologia, consiga acesso uniforme (Rômulo, 2002). O SO é responsável por realizar o controle dos dispositivos de E/S. Segundo Rômulo “o objetivo da gerência de E/S é criar uma

camada de software que esconda através de uma interface comum os detalhes específicos de cada dispositivo".

- **Gerência de Memória:** é função da gerência de memória prover mecanismos necessários para que os diversos processos compartilhem a memória RAM de forma segura e eficiente;

Como o objetivo deste trabalho é implementar a simulação da paginação por demanda, que está dentro da gerência de memória, a próxima seção descreve esta área mais detalhadamente.

2.2 Gerência de Memória

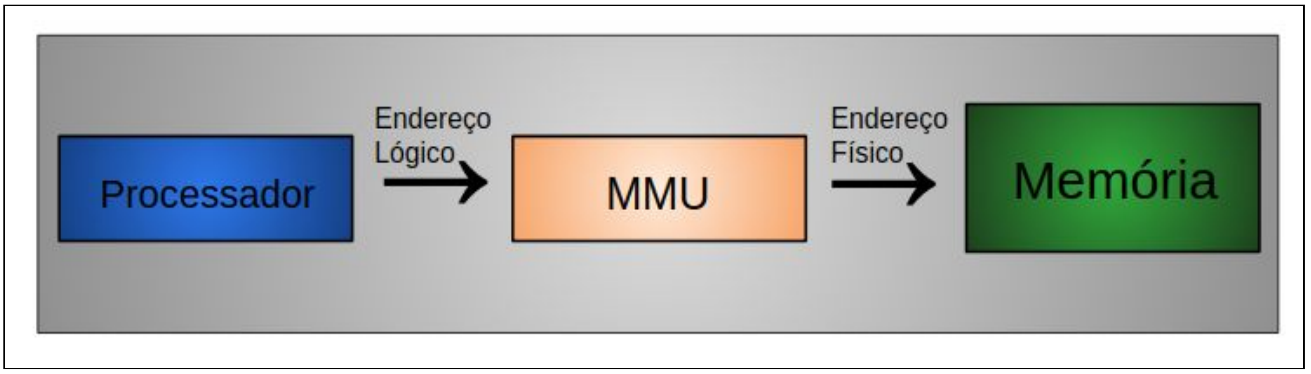
Através da divisão do tempo do processador diversos processos são executados de maneira simultânea concorrendo aos recursos como, por exemplo, a CPU e a memória RAM (memória física). Para que um processo utilize a CPU é necessário que outro processo deixe de utilizá-la. Assim, é necessário realizar o chaveamento que, segundo Toscani (2017), "é a troca do processo em execução o contexto do processo que estava em execução deve ser guardado e o contexto do próximo a executar deve ser restaurado". Como o acesso a memória é mais rápido que o acesso ao disco os processos devem estar carregados na memória RAM, prontos para execução, tornando o chaveamento e o uso dos dados mais eficientes.

No que se refere ao gerenciamento de memória, existem dois espaços de memória, a memória lógica e a memória física. A memória lógica de um processo é aquela que o processo vê, ou seja, é aquela que o processo é capaz de endereçar e acessar usando as suas instruções (OLIVEIRA, 2002). A memória física é a memória principal, que representa a quantidade de memória RAM instalada no computador. A memória física funciona em nível de máquina referenciando componentes eletrônicos e circuitos, desta forma, os endereços físicos correspondem a uma posição real na memória.

Quando um processo deseja realizar uma leitura ou escrita na memória, ele tem apenas o endereço lógico da informação. O processador passa o endereço lógico do dado que o processo deseja ler ou escrever e este deve ser mapeado para o endereço físico que indica onde a informação está armazenada na memória RAM.

A MMU (unidade de gerência de memória - *Memory Management Unit*) é o componente de *hardware*, que normalmente está no processador, responsável por retornar ao processador o endereço físico que indica o local em que o dado está armazenado na memória principal. Conforme Oliveira et al. (2010, p 157), "a MMU é que vai mapear os endereços lógicos gerados pelos processos nos correspondentes endereços físicos que serão enviados para a memória". A figura 2, apresenta o diagrama do funcionamento da MMU entre o processador e a memória.

Figura 2. Diagrama MMU - adaptada



Fonte: Oliveira, Carissimi e Toscani, 2010

Enquanto o processador possui os endereços lógicos, a MMU é o mecanismo responsável por estabelecer a relação dos endereços lógicos com os endereços físicos da memória. Quando o processador solicita a memória, a MMU recebe o endereço lógico e faz a tradução para o endereço físico da memória.

Nos livros didáticos de SO são apresentados vários mecanismos de gerência de memória, que determinam como fica a relação entre a memória lógica e a memória física. Os SOs implementam suas políticas de gerência de memória baseando-se no funcionamento de um destes mecanismos, de acordo com as suas necessidades. Dois desses mecanismos são a paginação e a paginação por demanda, que são descritos a seguir.

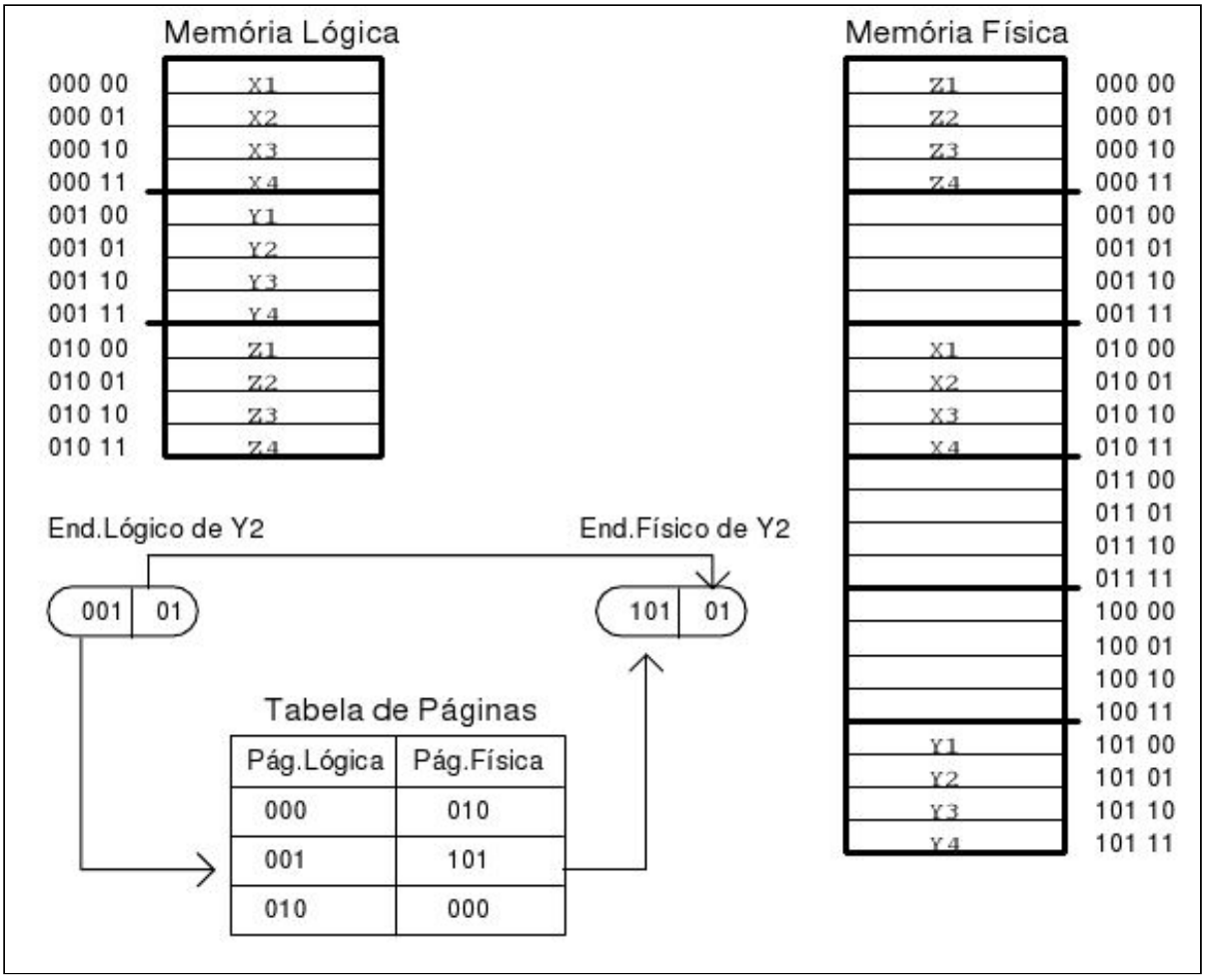
2.2.1 Paginação Simples

Na paginação simples cada processo possui uma memória lógica contígua, dividida em páginas lógicas de mesmo tamanho. A memória física é dividida em blocos de tamanho fixo, chamados quadros, em que o tamanho dos quadros é definido pelo MMU. As páginas lógicas têm o mesmo tamanho dos quadros da memória física. Uma tabela de páginas associa as páginas lógicas com as páginas físicas (OLIVEIRA, 2001).

Na paginação simples todas as páginas são sempre carregadas para a memória física, e existe uma entrada válida na tabela de páginas para cada página lógica. Os bits válidos e inválidos indicam se a página está dentro do espaço de endereçamento lógico. Quando o bit é válido ele é representado pela letra ‘V’, quando inválido é representado pela letra ‘I’.

A figura 3 apresenta um exemplo do mecanismo de paginação, em que um processo P1, com 12 bytes, está em execução.

Figura 3. Representação da paginação



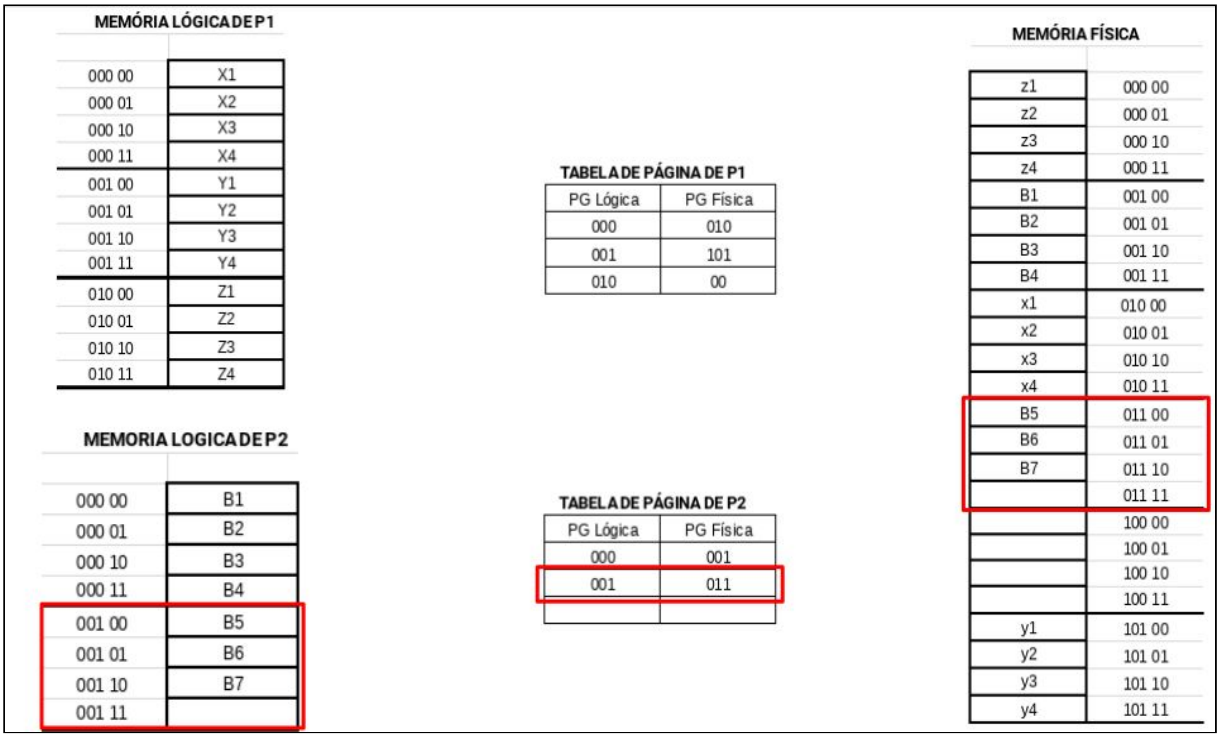
Fonte: Oliveira, Carissimi e Toscani, 2001

É apresentado na figura 3 o funcionamento da técnica de paginação. A memória lógica é composta por 12 bytes e foi dividida em 3 páginas lógicas cada uma. Os endereços lógicos também são divididos em duas partes: um número de página lógica e um deslocamento dentro dessa página. A memória física também é dividida em páginas físicas com tamanho fixo igual ao endereço da página lógica.

Durante a carga é montada uma tabela de páginas para o processo. Essa tabela é responsável por endereçar e mapear para cada página lógica a sua respectiva página física (OLIVEIRA, 2001). No exemplo, a memória física é composta por 24 bytes. A página física tem o mesmo tamanho que a página lógica.

Considerando a situação da memória física apresentada na figura 3, caso fosse inserido um processo B com 7 bytes, a situação do sistema, adicionando a memória lógica do novo processo, ficaria como apresentado na figura 4.

Figura 4. Nova tabela de páginas



Como pode ser verificado nos trechos destacados na figura 4, a página lógica 001 foi carregada na página física 011, o que é indicado na tabela de página de P2.

Embora evite a fragmentação externa, esse mecanismo permite a fragmentação interna. Entende-se por fragmentação interna quando, por exemplo, após a inserção do processo B, existe uma sobra no quadro reservado para este processo.

Como B tem 7 bytes e o quadro da memória lógica contém 8 bytes reservados, a área não usada, após a alocação é um exemplo de fragmentação interna. Essa área gera espaços muito pequenos. Esses espaços são tão pequenos que não conseguem alocar um processo inteiro.

Quando se usa paginação simples, todo o programa deve está alocado na memória física, mesmo as partes que não estão sendo executadas. Caso, por exemplo, um usuário

abrisse um editor de textos como o Microsoft Word e todos os seus recursos fossem carregados para a memória RAM, provavelmente não haveria memória suficiente capaz de executar.

Assim, sempre o espaço de memória física terá que ser suficiente para armazenar tudo que é implementado no programa que é executado. Uma maneira de resolver isso, é através da memória virtual que será detalhada na próxima seção.

2.2.2 Memória Virtual

A memória virtual de acordo com Oliveira (2001), "permite a execução de programas que não são carregados para a memória física". A virtualização da memória possibilita que programas, que não necessitam estar na memória o tempo todo, possam executar apenas a funcionalidade específica.

Quando não é possível manter todos os processos simultaneamente na memória, o mecanismo de gerência de memória reserva uma área no disco. Essa área é usada para que a memória faça cópias e trocas de processos com o disco, essa ação é conhecida como swapp.

Através do *swapping* o SO consegue executar mais processos do que os que caberiam em um mesmo instante na memória. Quando o processo é suspenso e copiado da memória para o disco é conhecido como swap-out. A ação inversa é conhecida como swap-in, quando o processo é copiado do disco para a memória e seu descritor volta para a fila do processador.

As políticas de escalonamento dos SO´s mais usados, como windows e linux, implementam o mecanismo de paginação por demanda, que é baseado no mecanismo de paginação simples, apresentado na seção 2.2.1, associado o conceito de memória virtual. A paginação por demanda é apresentada a seguir.

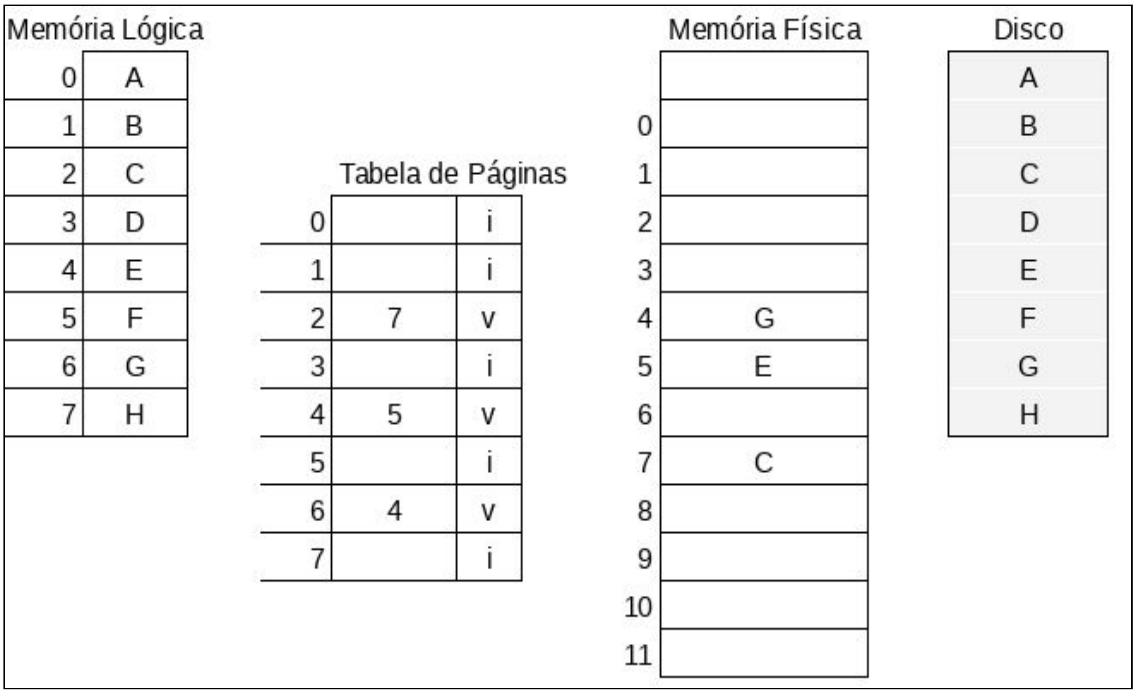
2.2.3 Paginação por demanda

Na prática é difícil usar paginação, pois todo o programa deve estar carregado na memória física, mesmo que parte dele não seja utilizada. Na paginação por demanda apenas as páginas acessadas pelo processo serão carregadas para memória física. A PD é baseada em

paginação com swapping, pois no momento em que os processos solicitam um conteúdo, para leitura ou escrita, este está no disco.

Os processos indicam o endereço lógico para serem carregados para a memória RAM, os conteúdos que ainda não foram acessados estão apenas no disco e devem ser copiados para a memória RAM. Desta forma, só vão para a memória física os processos que serão utilizados.

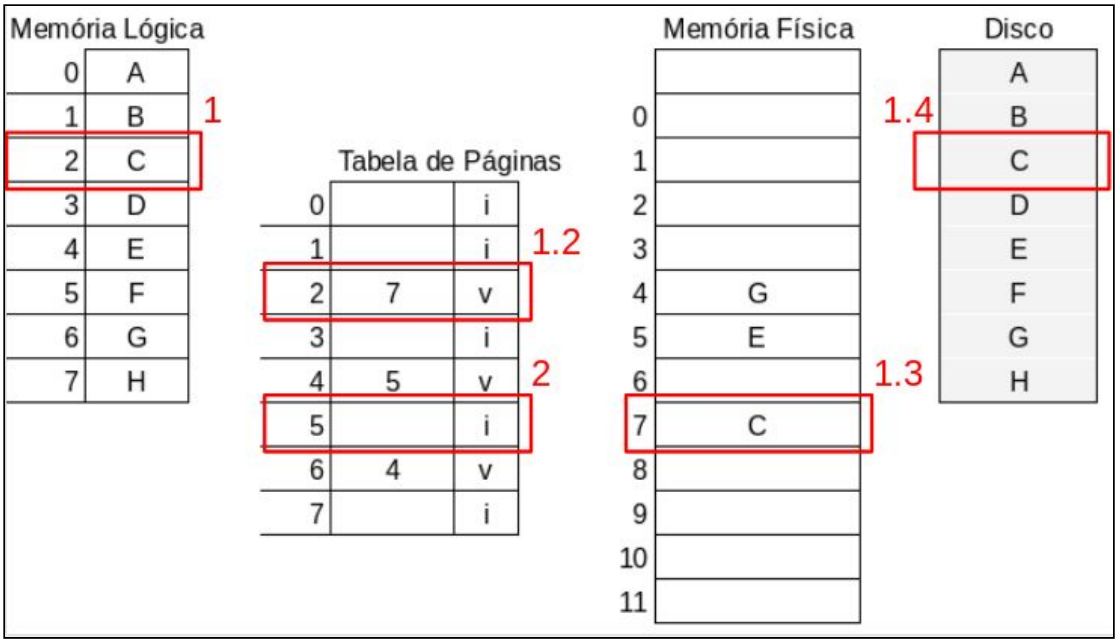
Figura 5. Representação do Mecanismo básico



Como pode ser visualizado na tabela de páginas, além da relação dos endereços lógicos e físicos, existe uma coluna com um bit válido ou inválido, o bit válido “v” representa que a página lógica foi carregada na memória física. O bit inválido “i” retrata que a página lógica não foi carregada na memória física e os dados estão apenas no disco. Ambos representam quais páginas lógicas foram carregadas na memória física.

A figura 6 representa a relação da memória lógica com a memória física através da tabela de páginas.

Figura 6. Representação do Mecanismo básico da PD



Como pode ser observado na figura 6, a página lógica 2, representada pela marcação vermelha número “1”, está carregada na página física 7, representada pela marcação vermelha “1.3”. Como pode ser observado na marcação vermelha “1.2”, na tabela de páginas os endereços lógico “2” e físico “7” e o bit é marcado com “v”, indicando que o endereço é válido, ou seja, está carregado na memória RAM.

Já na marcação vermelha “2”, mostra a situação em que a página lógica “5” não está carregada na memória RAM. A linha correspondente, na tabela de páginas não possui o endereço da página física e o bit está marcado com “i”, que representa que aquele conteúdo está apenas no disco.

Na PD, quando o processador solicita um endereço lógico para o mapeamento de endereço físicos são realizadas as ações seguindo a sequência do pseudocódigo, apresentado na figura 7. A sequência foi criada a partir de ações do mecanismo de PD apresentada por Oliveira (2001).

Figura 7. Pseudocódigo do funcionamento da PD

```
1  var fpepl = fila de processos esperando página lógica;
2
3  processador solicita o endereço da página lógica X;
4
5  se (na tabela de página X esta com bit v){
6      MMU retorna endereço físico associado a X;
7  }
8  senão se(na tabela de página X está com bit i){
9      MMU gera Interrupção de proteção e aciona o SO;
10     se (página X está fora do espaço de endereçamento do processo){
11         o processo é abortado;
12     }
13     senão{
14         ocorre Interrupção por falta de página e o SO é acionado;
15         o processo é suspenso;
16         descritor do processo é inserido na fpepl;
17         uma página física livre deve ser alocada;
18         o conteúdo da pg lóg. X é localizado no disco e transferido para a memória;
19         ao final da transferida é realizada um interrupção;
20         a tabela de páginas é atualizada;
21         o descritor do processo é colocado na fila de aptos;
22     }
23 }
```

Quando o processo chama uma instrução de leitura ou escrita, para executar o que foi solicitado o processador passa o endereço lógico. Como pode ser observado na figura 7, se o conteúdo não está na memória física (bit ‘i’) e está dentro do espaço de endereçamento (linhas 8 a 14) é gerada uma interrupção por falta de páginas. Nesse caso, são realizadas as ações das linhas 15 até a 21, que descrevem o tratamento da interrupção por falta de páginas, que é realizado pelo SO.

Quando está sendo feito o tratamento da falta de páginas e o processo está bloqueado na fila de processos esperando página lógica (fpepl), enquanto é realizada a transferência dos dados, a CPU continua trabalhando normalmente com outros processos.

Após o tratamento de falta de páginas, o processo volta para a fila de aptos e espera a CPU para executar. Quando o processo executa novamente, ele repete a chamada à instrução que gerou a falta de página, neste caso, será retornado o endereço físico associado, já que o conteúdo estará carregado na memória física, como foi explicado anteriormente.

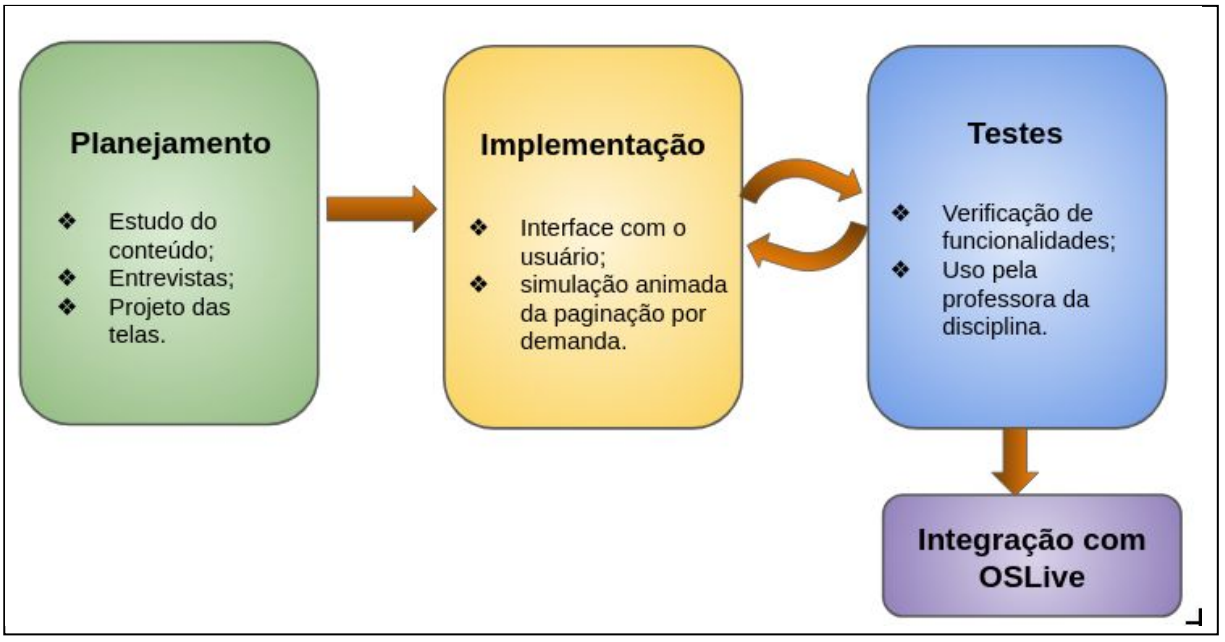
3 METODOLOGIA

Nesta seção encontra-se detalhada a metodologia de desenvolvimento deste trabalho, que consiste nos desenho de estudo definido e *softwares* que foram utilizados para o desenvolvimento do ambiente.

3.1 Desenho de Estudo

O desenvolvimento deste trabalho foi dividido da seguinte forma, conforme apresentado na figura 8.

Figura 8. Metodologia utilizada



As etapas para o desenvolvimento do trabalho, figura 8, são descritas a seguir:

- **Planejamento**: essa etapa divide-se em três fases:
    - Estudo do conteúdo: aquisição de conhecimento por meio de livros e artigos relacionados ao tema;

    - Entrevistas: nessa etapa foram realizadas reuniões com a especialista do domínio, professora da disciplina de Sistemas Operacionais, para definir quais funcionalidades a aplicação deveria apresentar;
    - Projeto da interface: com o apoio da especialista do domínio, foi projetada a interface com o usuário.

- **Implementação**: essa etapa divide-se em duas fases:
    - interface com o usuário: implementação de telas responsivas;
    - simulação: implementação da simulação que consiste na codificação do algoritmo de funcionamento do mecanismo de paginação por demanda, de acordo com os conceitos dos livros didático, mostrando a relação entre a memória lógica, SO e memória física.

- **Testes:** essa etapa divide-se em duas fases:
    - verificação de funcionalidades: foram realizados testes de que consistiram em utilizar a aplicação ao final de cada fase de implementação para verificar se a execução estava acontecendo de acordo com o que foi planejado para a funcionalidade oferecida;
    - uso pela professora da disciplina: através do uso pela professora da disciplina, que verificou a execução da simulação em vários momentos, foram feitos ajustes na simulação até que estivesse retornando os resultados adequados à utilização na disciplina.

- **Integração com OSLive:** essa etapa consiste na integração da aplicação com o ambiente de simulação OSLive. Esta etapa não foi feita neste trabalho, pois o ambiente OSLive ainda está em desenvolvimento**.

Vale salientar que a etapa de testes consistiu em validar a implementação e teve o objetivo de procurar problemas na implementação e quando os encontrou, os erros foram solucionados, mantendo-se esse ciclo constante até a finalização.

3.2 SOFTWARES

Para o desenvolvimento deste trabalho foram utilizadas as seguintes ferramentas: Visual Studio Code, Angular, Bootstrap e TypeScript, que são detalhadas a seguir:

- VSCODE: é um editor de texto leve e de código fonte aberto, que pode ser executado em qualquer ambiente de trabalho. No trabalho foi utilizado como IDE, para a edição e compilação dos códigos;
- Angular: é uma plataforma que facilita a criação de aplicativos para a Web, implementados em JavaScript. Angular combina modelos declarativos, injeção de dependência, ferramentas de ponta a ponta e práticas recomendadas integradas para resolver desafios de desenvolvimento (GOOGLE, 2018?). No trabalho foi usado para realizar a interação entre as classes e o html;
- Bootstrap JS/CSS: é um kit de ferramentas de código aberto para desenvolvimento com HTML, CSS e JS (BOOTSTRAP, 2018). No trabalho foi usado para criar a interface responsiva;
- Typescript: é uma extensão do JavaScript destinada a facilitar o desenvolvimento de aplicativos JavaScript de larga escala (Bierman, 2014). No trabalho foi usado para a adição de orientação a objetos (OO) ao Angular, tendo em vista que o JavaScript não é orientado a objetos.

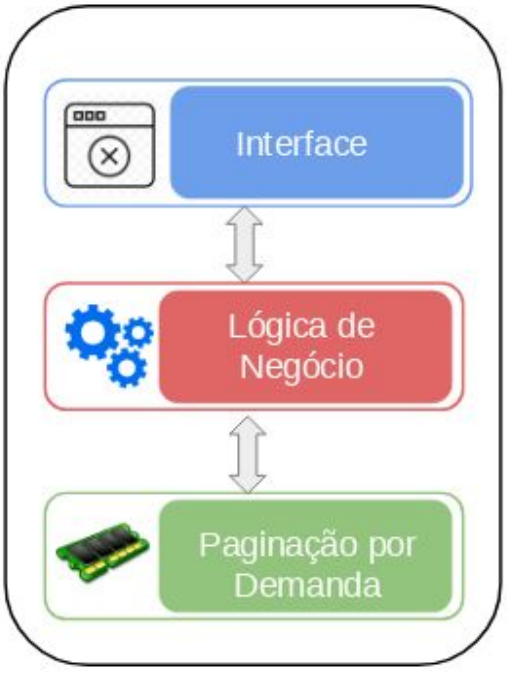
4 RESULTADOS E DISCUSSÃO

Esse trabalho consistiu na implementação de uma aplicação que simula o funcionamento do mecanismo de gerência de memória Paginação por Demanda, apresentando o que acontece quando os processos são criados. A seção Arquitetura do *Software* apresenta o modelo em camadas utilizado na implementação; a seção Interface da Aplicação mostra a tela inicial da aplicação e o que é apresentado para o usuário; e, por fim, é apresentado o exemplo da execução de uma simulação usando a aplicação.

4.1 Arquitetura do Software

Para melhor compreender o funcionamento da aplicação implementada, foi desenvolvida a arquitetura do *software* em camadas, no que se refere a organização e comunicação entre os elementos, como mostra a figura 9 que apresenta a arquitetura do software desenvolvido neste trabalho.

Figura 9. Arquitetura Proposta



Como pode ser verificado na figura 9, a arquitetura foi dividida em três camadas: Interface, Lógica de Negócios e Paginação por Demanda. A camada de Interface representa a interação com o usuário, na qual é realizada a entrada de dados e é mostrada a simulação do mecanismo de PD. A camada Lógica de Negócio é responsável por toda parte lógica da aplicação, que consiste em controlar o fluxo da informação, sendo assim, intermediária entre a interface e a parte responsável pela implementação da simulação do mecanismo de paginação por demanda. Na camada Paginação por demanda foi implementado o algoritmo da paginação por demanda, que é onde a simulação é realizada.

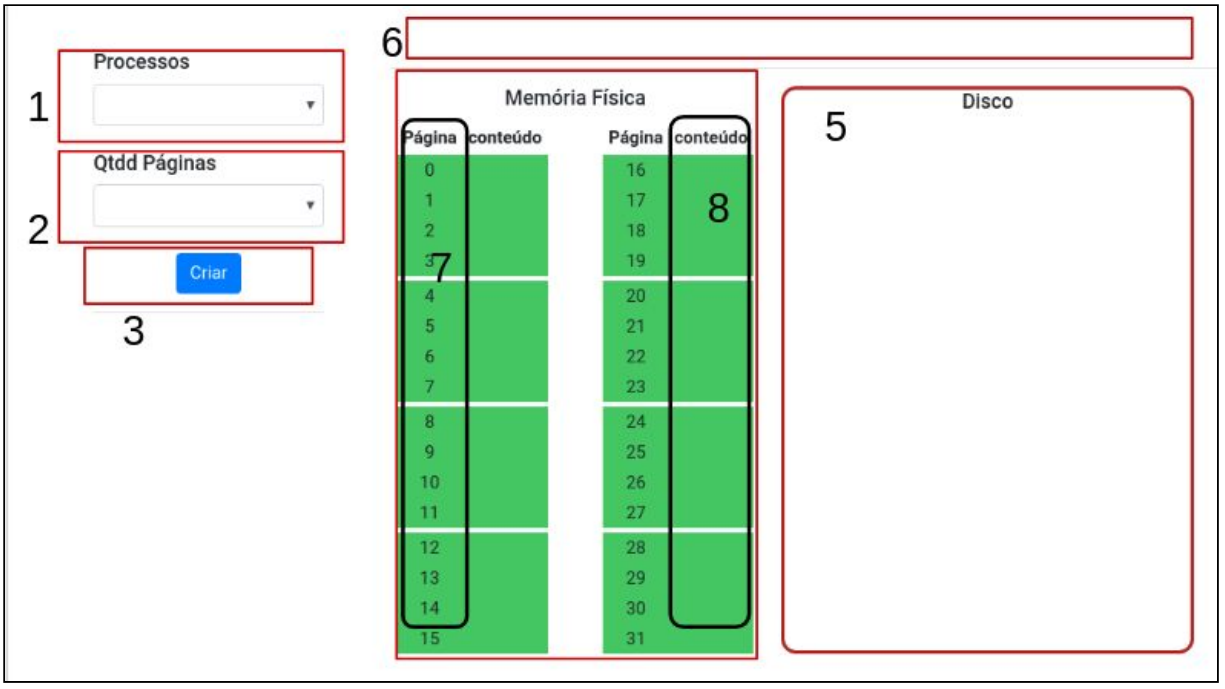
Inicialmente, foi implementada a interface e, para o desenvolvimento dessa camada, foi utilizado o HTML e Bootstrap. Depois, foi implementada a lógica de negócio e, por fim, o mecanismo de paginação por demanda. A camada de lógica de negócio e de paginação por demanda foram implementadas utilizando o angular e typescript.

As subseções seguintes apresentam a aplicação, descrevendo a interface da aplicação que apresenta o que é oferecido ao usuário e a execução de uma simulação que a partir de entradas do usuário é realizado um funcionamento diferente.

4.2 Interface da Aplicação

Essa seção apresenta a interface da aplicação, sendo que a tela mostrada na figura 10 é aberta quando o usuário acessa a aplicação, antes de executar qualquer entrada de dados ou simulação de execução de processo.

Figura 10. Interface da aplicação



Conforme mostra a figura 10, as áreas da interface foram destacadas com retângulos numerados. As áreas de 1 a 3 são as entradas, onde o usuário insere as informações do processo que será criado, e as áreas de 4 a 6 são apresentadas as saídas, onde a aplicação retorna as informações após executar a simulação.

A interface oferecida para a entrada de dados é a seguinte: na área 1, o usuário escolhe o nome do processo, uma letra entre A e H, que é único e identifica o processo ao longo da simulação; na área 2 o usuário informa a quantidade de páginas que o processo ocupará com o seu conteúdo, de forma que foi definido que cada processo pode ocupar no máximo 8 páginas; e na área 3 é oferecida ao usuário a opção de criar processo a partir das informações indicadas nas áreas 1 e 2.

Cabe destacar que a limitação de 8 páginas por processo foi uma questão estratégica, essa limitação não influencia na execução da simulação, no que se refere a representação do mecanismo, já que o funcionamento é o mesmo, independentemente da quantidade de páginas.

A interface oferecida para a saída de dados é a seguinte: na área 6 os processos criados são representados por seu nome, que foi escolhido pelo usuário na entrada de dados; na área 4

é apresentada a memória física (RAM), que é dividida por 32 páginas (0 a 31), já que no mecanismo de PD a memória é dividida em páginas físicas que serão associadas às páginas lógicas do processo; na área 7 é apresentado o endereço da página física; na área 8 é apresentado o conteúdo que está armazenado na página; na área 5 é apresentado o HD que é indispensável na simulação da PD, pois os conteúdos dos processos são armazenados no disco, sendo carregados para a memória física sob demanda, ou seja, quando utilizados.

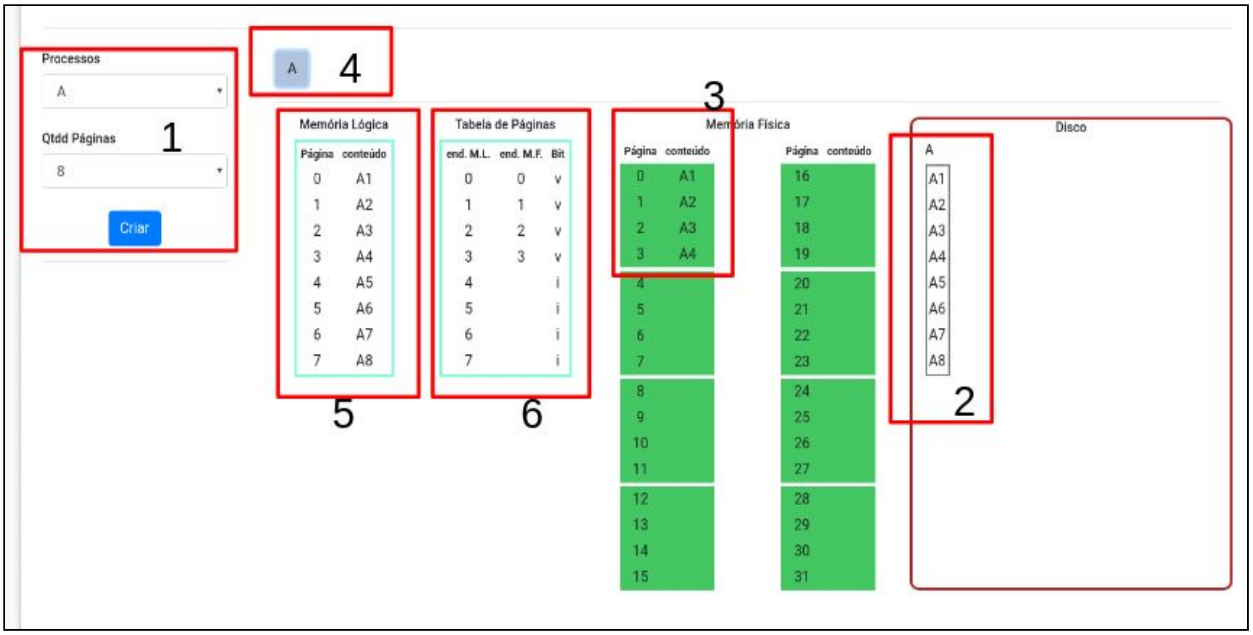
A seção 4.3 apresenta um exemplo de uma execução de simulação da PD, de forma que é possível verificar o funcionamento da aplicação.

4.3 Execução de uma Simulação

Esta seção apresenta um exemplo de simulação da execução da paginação por demanda, a partir das entradas de dados realizadas pelo usuário. Para isso, é mostrada a situação da memória lógica, da memória física e do disco após o início de cada processo.

As seguintes características da PD são apresentadas na simulação: todo o conteúdo da memória lógica do processo é armazenado no disco; os processos compartilham a memória física e tem sua memória lógica individual, sendo que a aplicação mostra a memória lógica do processo ativo. A figura 11 apresenta o resultado da simulação da PD após o usuário criar um processo **A** com **8** páginas.

Figura 11. Criar processo A

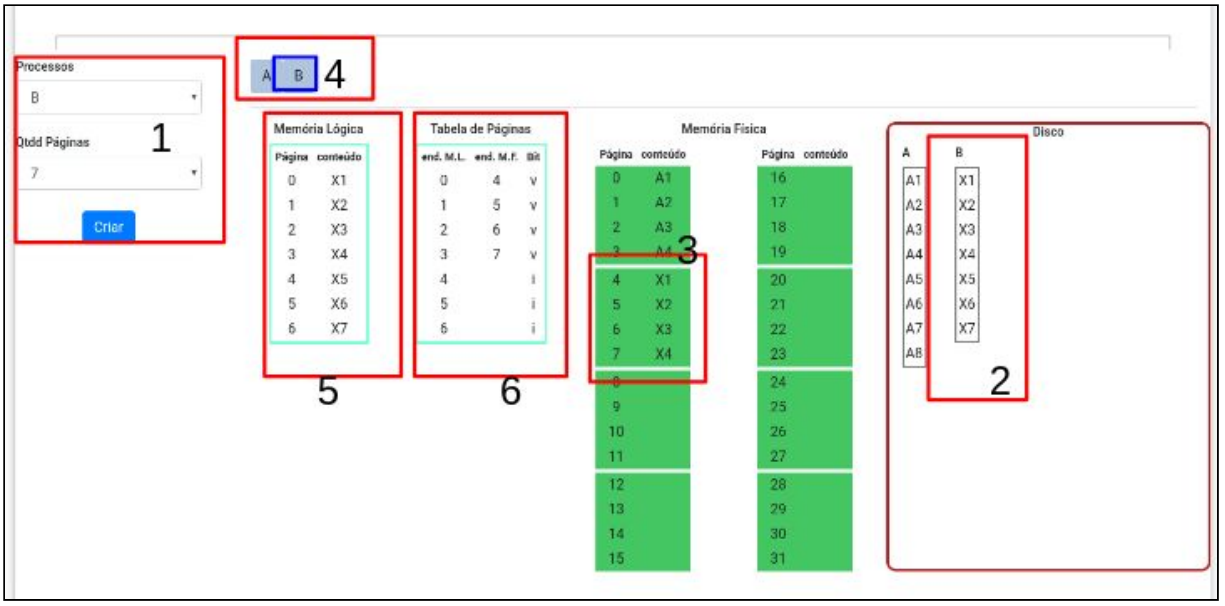


Conforme apresentado na figura 11, na área 1 o usuário preencheu os campos de entrada, selecionando o processo **A** com 8 páginas. Quando o usuário escolheu a opção "criar", o conteúdo das 8 páginas lógicas do processo foi todo carregado para o disco, área 2, o que é uma característica da PD.

Na área 2 é apresentado o disco, que armazena todo o conteúdo do processo; a área 3 mostra que parte do conteúdo do processo **A** foi carregado para a memória física, ou seja, apenas a parte que será executada no início do processo; a área 4 indica que o processo **A** foi criado; nas marcações 5 e 6 são apresentadas a memória lógica e tabela de páginas do respectivo processo, na tabela de páginas os endereços da memória lógica e memória física são relacionados e, quando existe essa relação, o Bit na tabela de páginas é alterado para 'v', caso não exista a relação, o Bit é marcado com 'i'.

A partir da situação apresentada, foi dada continuidade a simulação considerando que, após a criação do processo **A**, o usuário escolheu criar o processo **B** com 7 páginas**.** A figura 12 apresenta o resultado da simulação após o processo **B** ser criado, de forma que as áreas da aplicação foram atualizadas para a apresentar as informações do novo processo.

Figura 12. Criar processo B



Conforme apresentado na figura 12, a área 4 apresenta todos os processos criados. O processo **B** é o processo ativo, desta forma, a memória lógica e a tabela de páginas apresentadas são desse processo. Caso o processo **A** fosse selecionado seria apresentado a memória lógica e a tabela de páginas dele.

A memória lógica do processo **B** representa o conteúdo das 7 páginas que foram criadas, que estão armazenadas no disco, área 2. Essa memória agora mostra informações do processo selecionado onde cada página da memória física que possui o conteúdo de **B** está relacionada com as páginas da memória lógica. Apenas os endereços onde existe a relação memória lógica e memória física recebem o bit ‘v’ de válido, os demais bits permanecem com ‘i’ de inválido.

Na memória física houve o acréscimo de mais páginas, essas páginas são referentes ao processo **B** que acabou de ser criado. Esse processo tem 7 páginas e apenas 4 foram carregadas para a memória física. Como na memória física já existe um processo carregado, os dados do processo **B** são inseridos a partir da próxima página disponível.

Caso um novo processo fosse iniciado, seria indicado na área de processos criados, uma nova área de memória lógica seria criada assim como uma nova tabela de páginas. No disco seria demonstrado cada um dos novos processos.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo o desenvolvimento uma aplicação de simulação do mecanismo de paginação por demanda, tendo como objetivos específicos, a criação da representação gráfica do funcionamento do mecanismo de PD, projetar as telas com representação gráfica do mecanismo de PD, implementar a interface gráfica para a simulação da PD, em um ambiente web responsivo e, por fim, realizar e apresentar os testes funcionais. Todos os objetivos foram realizados, o que tornou válida a hipótese afirmada no início do trabalho, que foi "Usando as tecnologias Html5, Angular, Bootstrap e TypeScript é possível criar um simulador que auxiliará no aprendizado da paginação por demanda".

Acredita-se que a experiência que o usuário terá na utilização da ferramenta durante a simulação possa ajudar a entender o funcionamento da paginação por demanda, facilitando a associação da prática aos conceitos vistos em sala. Além disso, espera-se que que com a simulação do mecanismo haja mais interesse por parte dos alunos em se aprofundar no conceito disponível nos livros didáticos.

Buscou-se desenvolver uma ferramenta que atenda aos conceitos dos livros didáticos no que se refere a representação do mecanismo de paginação por demanda. Foi seguido o padrão estabelecido nos livros utilizados como referência na disciplina de SO de várias instituições nacionais.

Foi implementado o funcionamento do algoritmo do mecanismo de paginação por demanda quando os processos são criados. O funcionamento completo do mecanismo envolve a solicitação de uma página pela CPU, quando o processo está em execução, que pode estar carregada ou não, sendo que o funcionamento é diferente nos dois casos.

Além disso, não foi feito controle de usuários, o que poderia ser interessante para a ferramenta. Quando existe o controle de usuários existe a possibilidade de realizar uma simulação específica ou salvar-la para simular depois.

Assim, sugere-se como trabalhos futuros: implementar o que falta na simulação do funcionamento completo da PD, implementar o controle de usuários e criar atrativos para a utilização por parte dos alunos como, por exemplo, usar técnicas de gamificação.

REFERÊNCIAS

BARBIERE, Lu. O Que é Bootstrap e Para Que Serve?. 2018. Disponível em: <https://goo.gl/yNLFhK>. Acesso em: 01 out. 2018.

BIERMAN G., Abadi M., Torgersen M. (2014) Understanding TypeScript. In: Jones R. (eds) ECOOP 2014 – Object-Oriented Programming. ECOOP 2014. Lecture Notes in Computer Science, vol 8586. Springer, Berlin, Heidelberg

BOOTSTRAP, Bootstrap. Bootstrap. 2018. Disponível em: <http://getbootstrap.com/>. Acesso em: 01 out. 2018.

BRASIL. Ministério da Educação. Parecer CNE/CES Nº 136/2012, aprovado em 8 de março de 2012. Diretrizes Curriculares Nacionais para os cursos de graduação em Computação. Disponível em:<https://goo.gl/cNQr1X>. Acesso em: 11 setembro 2018.

DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R. Choffnes. Sistemas Operacionais. 3° Edição. ed. São Paulo - Brasil: Pearson, 1999. 760 p. Disponível em: <https://goo.gl/7Me4my>. Acesso em: 08 nov. 2018.

DE SIQUEIRA, Fernando. Gerência de Sistema de Arquivos. Disponível em: <https://goo.gl/VqjHPN>. Acesso em: 07 nov. 2018.

FOUNDATION, The JQuery. JQuery. 2018. Disponível em: <http://jquery.com/>. Acesso em: 01 out. 2018.

GOOGLE. What is Angular?. [*S. l.*], 2018?. Disponível em: https://angular.io/docs. Acesso em: 15 abr. 2019

OFICINA, Redação. MySQL - o que é?. 2018. Disponível em: <https://goo.gl/FjyagG>. Acesso em: 01 out. 2018.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. Sistemas Operacionais. 2. ed. Porto Alegre: Instituto de Informática da UFRGS : Sagra Luzzatto, 2001. 247 p.

PAULO MAIA, Luiz. SOsim: SIMULADOR PARA O ENSINO DE SISTEMAS OPERACIONAIS. RIO DE JANEIRO, RJ – BRASIL: [s.n.], 2001. 97 p. Disponível em: <https://goo.gl/sqbqM3>. Acesso em: 03 nov. 2018.

PRADO, Darci dos Santos, Teoria das Filas e da Simulação. 2° Edição. ed. Belo Horizonte (MG): Editora de Gerenciamento Gerencial, 2004. 125 p. v. Volume 2.

RIBEIRO, Uirá Endy. Sistemas Distribuídos: Desenvolvendo Aplicações de Alta Performance no Linux. [S.L.]: Axcel Books do Brasil Editora, 2005. 384 p.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. Fundamentos de Sistemas Operacionais. 8. ed. Rio de Janeiro: Ltc, 2013. 515 p.

S. TANENBAUM, Andrew. Sistemas Operacionais Modernos. 2° Edição. ed. São Paulo - Brasil: Pearson, 2003. 685 p. Disponível em: <https://goo.gl/JByp4p>. Acesso em: 06 nov. 2018.

TOSCANI, Simão Sirineo. Gerência de Processador. Disponível em: <https://goo.gl/N4shVy>. Acesso em: 16 jun. 2017.