|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | |  | |
|  | escola  superior  de tecnologia  e gestão  POLITÉCNICO  DO PORTO | | | | |  | |
|  | | | LEI |  |  | | |
| lICENCIATURA Engenharia INFORMÁTICA  SISTEMAS OPERATIVOS | |  |
|  | | | Trabalho Prático  Grupo A  José Magalhães – 8210125  Miguel Carvalho – 8210130  01/2024 | | |  |
|
|
|  | | | | | | | |

Índice

[Índice 3](#_Toc155049228)

[Índice de Figuras 5](#_Toc155049229)

[1 Introdução 6](#_Toc155049230)

[1.1 Âmbito / Contextualização 6](#_Toc155049231)

[1.2 Objetivos 6](#_Toc155049232)

[1.3 Estrutura do documento 7](#_Toc155049233)

[2 Manual de compilação, configuração e utilização da aplicação 8](#_Toc155049234)

[3 Descrição das funcionalidades implementadas 9](#_Toc155049235)

[3.1 Módulo *Kernel* 9](#_Toc155049236)

[3.2 Módulo MEM 9](#_Toc155049237)

[3.3 Módulo CPU 10](#_Toc155049238)

[3.4 Módulo Middleware 10](#_Toc155049239)

[4 Descrição e justificação da utilização de mecanismos de sincronização e comunicação entre módulos 11](#_Toc155049240)

[4.1 Módulo Main 11](#_Toc155049241)

[5 Funcionalidades adicionais 12](#_Toc155049242)

[5.1 Módulo Login 12](#_Toc155049243)

[5.1.1 Ficheiro JSON 12](#_Toc155049244)

[5.2 Módulo Gráfico Informativo 12](#_Toc155049245)

[5.3 Módulo 12](#_Toc155049246)

Índice de Figuras

[Figura 1 - Logo 8](#_Toc156241769)

[Figura 2 - Gráfico 13](#_Toc156241770)

1. Introdução

O desenvolvimento de sistemas operativos eficientes e robustos é essencial para uma gestão eficaz dos recursos computacionais em ambientes críticos, como é o caso de satélites. Este relatório documenta o trabalho prático realizado na unidade curricular de Sistemas Operativos, cujo objetivo é simular um sistema operativo para um satélite, incorporando conceitos de multiprocessamento, comunicação e sincronização.

* 1. Âmbito / Contextualização

No âmbito desta disciplina, o trabalho proposto abrange a simulação de um sistema operativo de um satélite que gere duas unidades de computação, o CPU e a MEM, com ênfase na sincronização e comunicação entre os diferentes componentes. O propósito desta simulação está na necessidade de assegurar que os sistemas computacionais em satélites satisfaçam requisitos críticos, tais como segurança, robustez, fiabilidade e a capacidade de comunicação em tempo real. Ao enfrentar estes desafios, tivemos a oportunidade de aplicar conceitos teóricos adquiridos durante aulas práticas.

* 1. Objetivos

O principal objetivo deste projeto foi desenvolver um simulador de sistema operativo para um satélite, utilizando linguagem Java e aplicando conceitos já mencionados. O trabalho visa proporcionar uma compreensão prática da matéria abordada na unidade curricular, focando-se em situações reais de gestão de recursos computacionais em tempo real.

* 1. Estrutura do documento

Este relatório é composto por vários capítulos e subcapítulos de modo a facilitar a leitura e a interpretação dos conteúdos. A estrutura compreende os seguintes capítulos:

* **Introdução** – Este capítulo apresenta uma visão geral do trabalho.
* **Manual de compilação, configuração e utilização da aplicação** – Neste capítulo serão fornecidas orientações detalhadas sobre como compilar, configurar e utilizar a aplicação desenvolvida.
* **Descrição das funcionalidades implementadas** – Aqui, será realizada uma análise detalhada das funcionalidades desenvolvidas no sistema operativo. Cada componente, destacando as suas responsabilidades e contribuições para o bom funcionamento da aplicação.
* **Descrição e justificação da utilização de mecanismos de sincronização e comunicação entre módulos** – Este capítulo explora os mecanismos de sincronização e comunicação implementados no projeto. Serão identificados e explicados os métodos escolhidos para garantir a coordenação eficiente entre as diferentes partes do sistema.
* **Funcionalidades adicionais** – Neste último capítulo, serão apresentadas as funcionalidades adicionais propostas e implementadas para enriquecer o sistema operativo.

1. Manual de compilação, configuração e utilização da aplicação

Este trabalho foi desenvolvido na linguagem *JAVA* e utilizado o *Visual Studio Code*.

Recomendamos a instalação e configuração prévia dessas ferramentas para facilitar a utilização da aplicação. O código-fonte completo está disponível no repositório do GitHub “<https://github.com/TPsLEI/TP_SO>”. Qualquer documentação adicional ou informações relevantes também podem ser encontradas no mesmo repositório.

Para compilar e executar o projeto, é necessário seguir os seguintes passos:

1. Abrir o terminal na pasta do “projeto”.
2. Executar os seguintes comandos, um de cada vez:
   * “javac -cp ./lib/miglayout.jar;./lib/miglayoutcore.jar;./lib/flatlaf.jar;./lib/json.jar;./lib/jfreechart.jar;./lib/jcommon.jar -d ./classes -encoding UTF-8 \*.java”
   * “java -cp .;./lib/miglayout.jar;./lib/miglayoutcore.jar;./lib/flatlaf.jar;./lib/json.jar;./lib/jfreechart.jar;./lib/jcommon.jar;./classes Kernel”

**NO FIM, COLOCAR EM PDF**

Estes passos garantem a compilação e execução adequadas do projeto.

Outra opção, em vez de executar os comandos manualmente, é simplesmente abrir o arquivo de configuração “*start.bat*”. Este arquivo já contém os comandos necessários para compilar e executar o projeto de forma automatizada. **\*\*\*\*\***

**MULTIPROCESSAMENTO, MULTIPLAS THREADS**

**Meter repositório público**

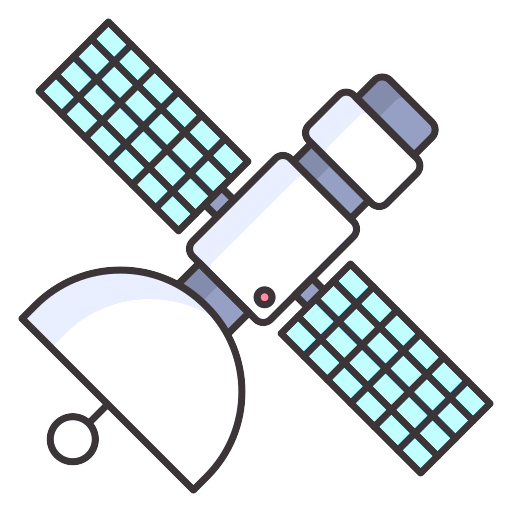


Figura 1 - Logo

1. Descrição das funcionalidades implementadas
   1. Módulo *Kernel*

O núcleo essencial do sistema, desempenha um papel fundamental ao processar, controlar e validar as etapas fundamentais do sistema operativo de um satélite. Este módulo fica encarregue das verificações e controlo, desempenhando um papel crucial ao tomar as decisões essenciais para garantir o bom funcionamento do sistema.

A função principal do “*kernel*” é iniciar a aplicação e exibir a página de Login. Utilizando a classe *“SwingUtilities”*, garantimos que a criação da interface gráfica ocorre de forma segura, proporcionando assim uma boa experiência ao utilizador.

A função *“handleLogin”* é essencial para autenticar utilizadores. Ao receber informações de nome de utilizador e password, o “*kernel*” lê um ficheiro JSON que contém dados de utilizador. Se o nome de utilizador corresponder a um registo existente e a palavra-passe estiver correta, o utilizador é autenticado com sucesso, resultando na abertura de uma nova página do menu. Caso contrário, é fornecida uma mensagem de erro.

Também são utilizadas duas funções, *“exportMessages”* e *“exportLogs”* para exportar mensagens e logs para arquivos CSV, respetivamente. Este processo é facilitado pelo uso da biblioteca *Swing*.

Foi implementada um método que fica responsável por atualizar periodicamente a “*textBox*” do satélite, garantindo assim que as informações exibidas são sempre as mais recentes.

Por último, foi adicionada uma função que responde às mensagens recebidas.

* 1. Módulo MEM

O módulo MEM desempenha um papel fundamental no armazenamento e manipulação de dados, contribuindo para a comunicação eficiente entre o satélite e a estação.

Na classe MEM, o método “*writeMessage*” foi desenvolvido para armazenar mensagens enviadas pelos utilizadores. A utilização do CompletableFuture possibilita a execução assíncrona dessas operações. Além disso, o método incorpora um log interno para rastrear as ações realizadas, proporcionando uma visão detalhada de todas as atividades do sistema.

O método “*log*” é responsável por registar eventos significativos num arquivo de *log* compartilhado. A sincronização desse método garante que a escrita no arquivo seja thread-safe, prevenindo possíveis situações de competição (*race conditions*). Um semáforo foi utilizado para a exclusão mútua, garantindo a consistência dos registos.

A implementação também inclui a consideração da criação de diretórios necessários para os arquivos de *log*.

* 1. Módulo CPU

O módulo CPU desempenha um papel crucial na gestão, escalonamento e execução das tarefas do sistema. Este módulo é responsável por coordenar a comunicação entre o satélite e a estação, utilizando os recursos fornecidos pelas unidades de computação MEM e Middleware.

A classe CPU, que estende Thread, utiliza uma estrututa de dados do tipo “*LinkedBlockingQueue*” que atua como um canal de comunicação entre os diferentes componentes do sistema, permitindo uma troca segura e ordenada de informações.

No método “*run*”, o CPU entra em *loop* contínuo, aguardando constantemente por novas mensagens na fila de dados. Quando uma mensagem é recebida, o módulo CPU regista a hora atual, formata a mensagem com as informações do remetente e encaminha para a unidade MEM para um armazenamento assíncrono. Após este processo, ocorre um atraso de 3 segundos antes da mensagem ser exibida, simulando o envio de dados em tempo real.

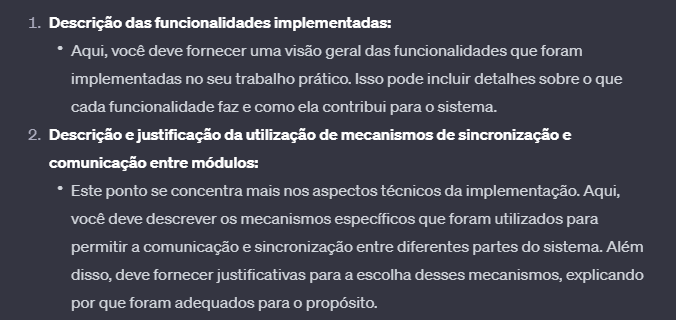
É relevante salientar que o método “*showMessageBox*” interage com o módulo Kernel para exibir uma caixa de mensagem, representando a comunicação entre o satélite e o utilizador.

* 1. Módulo Middleware

O módulo Middleware representa um componente crucial do sistema, desempenhando na comunicação entre as diferentes tarefas do sistema.

A classe CPU,

1. Descrição e justificação da utilização de mecanismos de sincronização e comunicação entre módulos
   1. Módulo Main



1. Funcionalidades adicionais

Além das funcionalidades solicitadas, foram adicionadas funcionalidades extra à aplicação. Este capítulo destaca essas adições, contribuindo para uma experiência mais completa.

* 1. Módulo Login
     1. Ficheiro JSON
  2. Módulo Gráfico Informativo

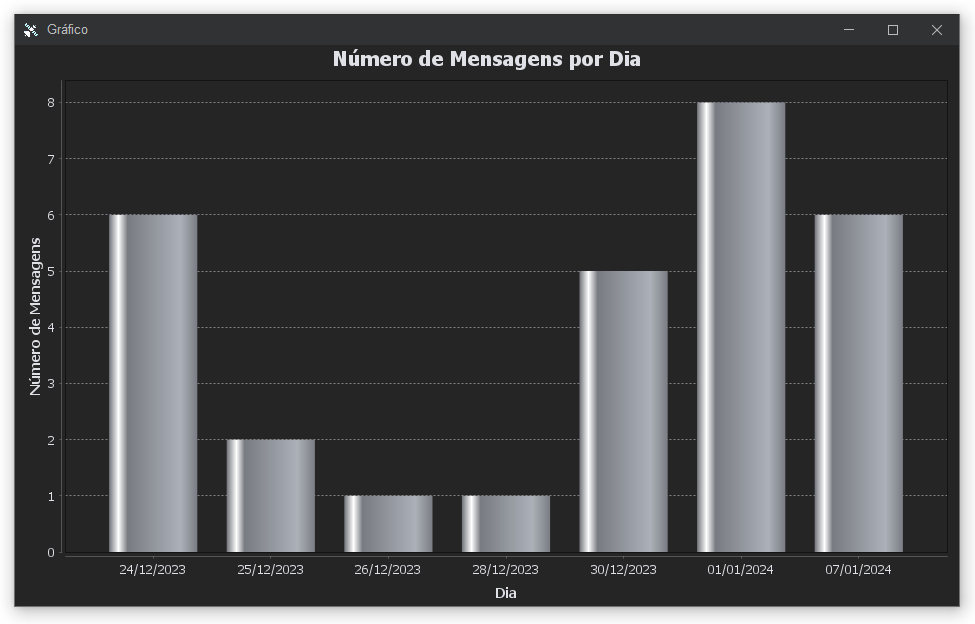


Figura 2 - Gráfico