Relatório – Meta Final

Coimbra, Maio 2024

Autores

**Hugo Pereira nº 2012011594**

**Diogo Coelho nº 2019143273**

**Sistemas Operativos 2**

LICENCIATURA EM ENGENHARIA INFORMÁTICA



INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR

DE ENGENHARIA

DE COIMBRA

# Índice

[Índice 1](#_Toc166945242)

[1 Identificação 2](#_Toc166945243)

[2 Introdução 2](#_Toc166945244)

[3 Diagrama Representativo 3](#_Toc166945245)

[4 Descrição dos Mecanismos 4](#_Toc166945246)

[4.1 Mecanismos de Comunicação 4](#_Toc166945247)

[4.2 Mecanismos de Sincronização 5](#_Toc166945248)

[4.3 Estruturas 6](#_Toc166945249)

[5 Tabela Requisitos 7](#_Toc166945250)

[6 Conclusão 8](#_Toc166945251)

# Identificação

Este documento refere-se à Meta Final relativamente a Sistemas Operativos 2.

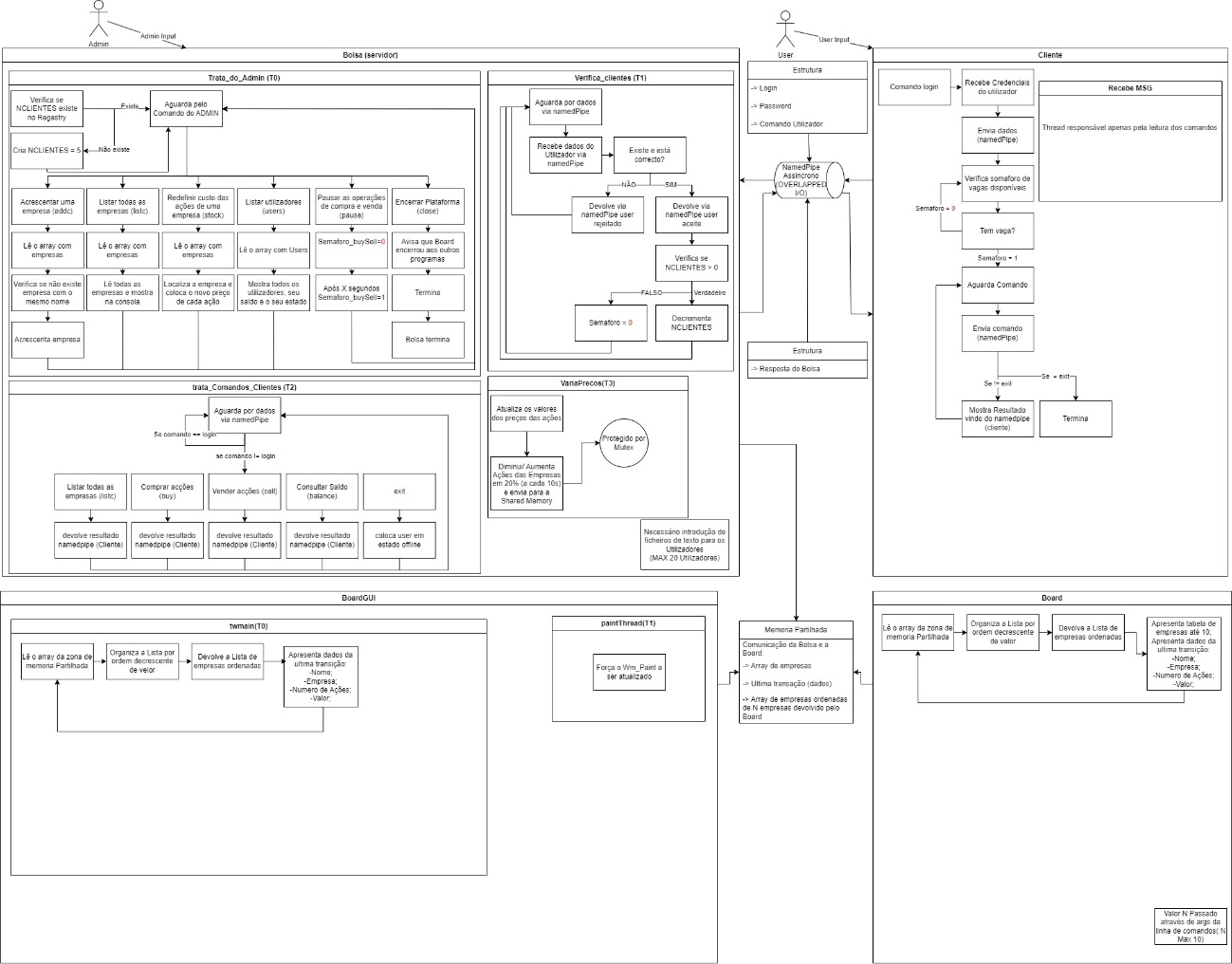
O trabalho foi realizado por: Hugo Pereira, cujo número de aluno 2012011594 e email a21220079@isec.pt, Diogo Coelho, cujo número de aluno 2019143273 e email a2019143273@isec.pt.

# Introdução

Este trabalho prático consiste no desenvolvimento de uma aplicação, com recurso à *Application Programming Interface* (API) do *Windows*, um programa denominado por bolsa de valores online, onde se vai abordar os vários conceitos abordados ao longo do semestre. O programa cliente, vai ser a interface que os clientes irão utilizar para interagir com o bolsa. O *board*, é um programa que irá ser responsável pela reorganização dos valores das ações das empresas. Por fim, iremos ter o *boardGUI* que servir apenas para visualizar os valores das empresas de forma gráfica. A proposta visa simular o funcionamento básico de uma bolsa de valores, utilizando múltiplos processos que interagem entre si em um ambiente local.

Nesta meta final iremos explicar os pontos essenciais da implementação de cada um dos programas envolvidos, as estruturas de dados definidas e a sua utilidade juntamente com o diagrama representativo atualizado.

# Diagrama Representativo



# Descrição dos Mecanismos

## Mecanismos de Comunicação

* Na nossa bolsa utilizámos vários mecanismos de comunicação, nomeadamente a nossa *Shared Memory* em que esta permite que diferentes *threads* e processos acedam e atualizem dados compartilhados, como a informação das empresas e das ações de forma ordenada**.** Essa abordagem de comunicação baseada em *Shared Memory* contribui para a robustez e estabilidade do sistema, garantindo que as informações sejam consistentes e acessíveis a todos os componentes envolvidos.
* Relativamente a minha board utilizamos mecanismos de comunicação associado ao armazenamento de dados das empresas da bolsa de valores, fazendo com que através da nossa *shared Memory* diferentes partes do programa possam aceder e atualizar os dados de forma eficiente.
* No meu cliente utilizamos *namedpipes*, com recurso ao *overlapped* I/O para escrever e receber dados do utilizador.
* Finalmente na nossa BoardGUI utilizámos a *shared Memory* apenas para ler os dados que passam por esta memoriapara podermos ver em tempo real as mudanças na nossa interface Gráfica.

## Mecanismos de Sincronização

* Utilizamos **semáforos** para controlar o acesso concorrente à aplicação. Em particular, limitamos o número máximo de clientes que podem entrar simultaneamente na bolsa para 5, garantindo assim uma utilização controlada e eficiente dos recursos do sistema.
* Usamos **eventos** para controlar os valores aleatórios do aumento e diminuição das ações. Também usamos para o controlo de compra e venda de ações para estas serem realizadas ordenadamente e sincronizada. A utilização de eventos permitiu sincronizar as *threads* de acordo com as condições necessárias, contribuindo para um funcionamento fluido e eficiente da aplicação.
* A utilização de **mutexes** foi uma prática consistente em todo o projeto. Esses mecanismos foram empregados para proteger o acesso à memória compartilhada onde estão armazenados os dados das empresas. Além disso, garantiram a exclusão mútua dos dados compartilhados, assegurando que apenas uma *thread* por vez pudesse acessar ou modificar esses dados. Isso foi fundamental para evitar condições de corrida e garantir a consistência dos dados em ambiente multithread.
* No cliente, utilizamos o ***overlapped* I/O** para operações de leitura e escrita em arquivos de forma assíncrona (através das threads ) . Essa abordagem permitiu que as operações de I/O pudessem ser realizadas de forma eficiente, sem bloquear a execução do programa principal, melhorando assim o desempenho e a responsividade da aplicação.
* Relativamente ás *threads* foi necessário em todo o nosso código, para não alongar muito, vamos apenas destacar a utilização destas nas nossas funções **verificaClientes, trataClientes, variaPreços.** Estas threads desempenham funções importantes no sistema, como monitorar clientes, tratar dos comandos dos clientes, ajustar preços de ações e sincronizar dados entre memória compartilhada e memória local. A criação dessas *threads* permite que essas tarefas sejam executadas em segundo plano, sem bloquear a execução do programa principal.

## Estruturas

* **empresaData:** Esta estrutura representa os dados de uma empresa, incluindo o seu nome, o número de ações disponíveis e o preço de cada ação. É fundamental para armazenar e manipular informações sobre as empresas listadas na bolsa.
* **clienteData:** Esta estrutura contém os dados de um cliente, como nome de utilizador, palavra-passe e comandos que o cliente envia para interagir com o sistema. É útil para autenticar e processar comandos enviados pelos clientes.
* **clienteResposta:** Esta estrutura é utilizada para armazenar a resposta que será enviada de volta ao cliente após o processamento dos seus comandos. Inclui uma resposta e possivelmente dados adicionais, como informações sobre empresas.
* **userData:** Contém os dados de um utilizador, como nome de utilizador, palavra-passe, saldo e estado (ativo/inativo). É essencial para gerir informações de utilizadores, como clientes que estão registados no sistema.
* **carteiraAcoes:** Esta estrutura representa a carteira de ações de um utilizador, contendo informações sobre ações específicas de uma empresa que um utilizador possui, como nome da empresa, nome de utilizador, número de ações e valor total das ações.
* **UltimaTransacao:** Guarda informações sobre a última transação realizada, incluindo o nome da empresa, número de ações e preço da ação. É útil para exibir informações sobre transações recentes.
* **ControlPause:** Uma estrutura simples que indica se o sistema está pausado ou não. Esta informação pode ser útil para controlar o fluxo de operações em determinados momentos.
* **boardData:** Contém os dados partilhados da bolsa, incluindo informações sobre empresas, carteiras de ações e a última transação. Estes dados são partilhados entre diferentes partes do sistema e são protegidos por mecanismos de sincronização, como mutexes.
* **tDataInfo e tDataInfo\_EXTRA:** Estruturas que mantêm informações sobre threads e pipes, usadas para comunicação e sincronização entre diferentes partes do sistema.
* **DATA:** Contém dados relacionados a uma operação de comunicação com um cliente, como identificador de pipe, indicador de continuação, handle de evento, dados do cliente e resposta a ser enviada.
* **BoardGUIDados:** Usada para armazenar dados relacionados à interface gráfica do utilizador (GUI) da bolsa de valores, como handle da janela, indicador de continuação e limites inferior e superior.

# Tabela Requisitos

|  |  |  |
| --- | --- | --- |
| ID | Descrição Requisito | Estado |
| 1 | Comandos Bolsa(Admin) | Implementado |
| 2 | Comandos Cliente | Implementado |
| 3 | Operações - BoardGui | Tudo Implementado menos o LimInferior (Não conseguimos resolver) |
| 4 | *Shared Memory* | Implementado |
| 5 | *Overlapped* *I/O* | Implementado |
| 6 | Eventos | Implementado |
| 7 | *Mutex* | Implementado |
| 8 | *Threads* | Implementado |
| 9 | Semáforos | Implementado |
| 10 | *Namedpipes* | Implementado |
| 11 | Variação de preços (Aleatório) | Implementado |

# Conclusão

Estamos extremamente satisfeitos e orgulhosos do trabalho realizado no desenvolvimento da nossa bolsa de valores. Ao longo do projeto, aplicamos diligentemente todas as técnicas e conceitos aprendidos em aula, incluindo *mutexes*, eventos, memória compartilhada e *threads*, para garantir a sincronização e comunicação adequada entre as diferentes partes do sistema.

Apesar dos esforços dedicados, reconhecemos que existem áreas que precisam de melhorias. Por exemplo, notámos que a saída do servidor pode ser melhorada, pois atualmente requer que o cliente pressione uma tecla para encerrar, em vez de fechar instantaneamente quando o servidor é encerrado. Além disso, encontramos desafios na implementação do limite inferior na parte gráfica, que não conseguimos resolver completamente.

No entanto, apesar desses pequenos obstáculos, estamos muito satisfeitos com o resultado alcançado. A nossa bolsa de valores é uma plataforma viável e confiável, oferecendo estabilidade e eficácia para as negociações. Este projeto representou uma valiosa oportunidade de aplicar os conhecimentos adquiridos em sala de aula e demonstrou nossa capacidade de criar soluções práticas e funcionais para desafios complexos.