Matéria: Sistema Operacionais II

Professor: Fernando Gonçalves Abadia

Alunos: - Carlos Eduardo Watanabe Nunes

- Lucas Barbosa Antonelli

- Lucas de Paula Carvalho Medolla

Relatório: Sistema de Expedição Pokémon

1. Introdução e Descrição do Cenário Escolhido

O sistema em questão simula uma Pokémon League inspirada no universo dos jogos Pokémon, no qual está havendo uma grande liga Pokémon onde treinadores do mundo todo estão participando, juntamente com seus Pokémons, para épicas batalhas em arenas.

O gerenciamento central é feito pelo Controlador Mestre, que inicializa e supervisiona os processos, enquanto uma interface gráfica (GUI) em C# monitora em tempo real as estatísticas do sistema por meio de memória compartilhada.

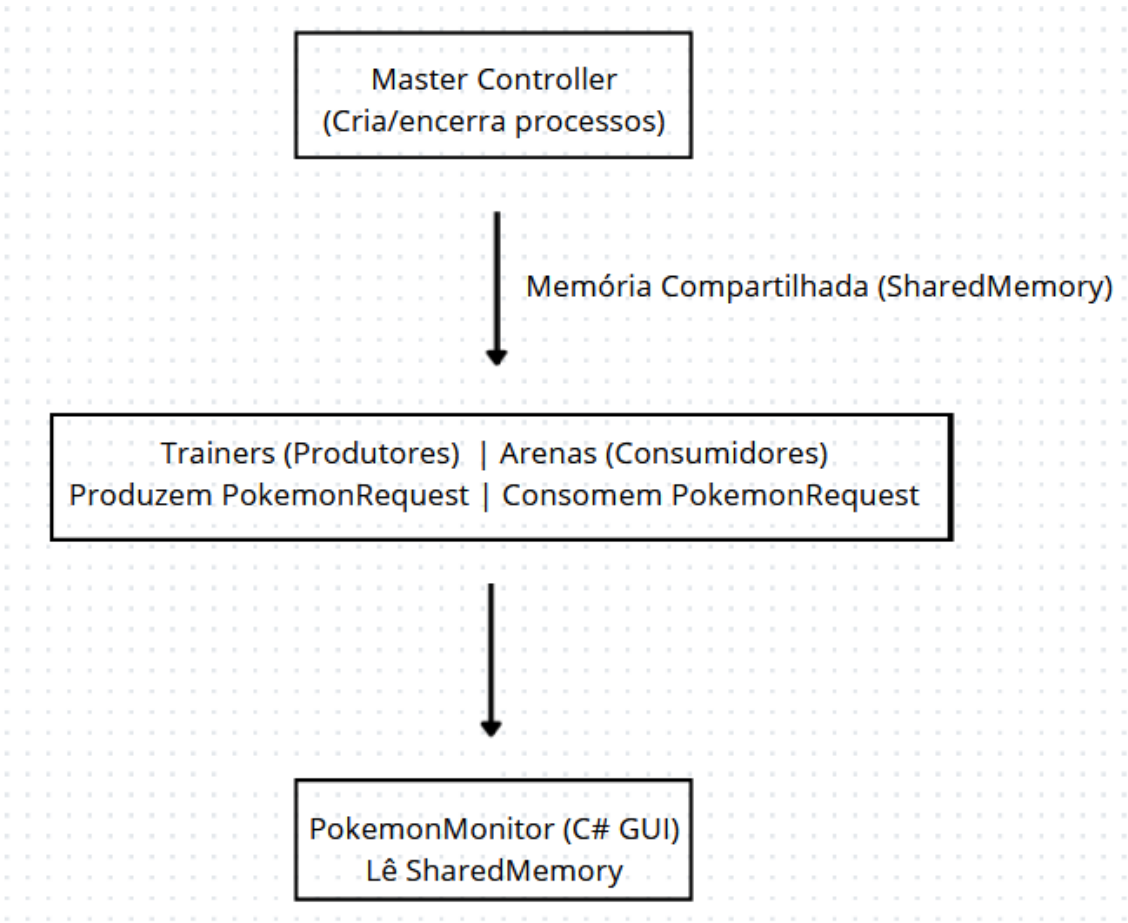
O sistema foi desenvolvido em C (Windows API), empregando processos, semáforos, mutexes e memória compartilhada para garantir sincronização e comunicação entre processos.

2. Arquitetura da Solução

2.1. Componentes Principais

Componente	Descrição
master_controller.exe	Inicia, supervisiona e encerra os processos de arenas e treinadores.
trainer.exe	Simula um treinador que envia Pokémon (produtor).
arena.exe	Simula uma arena que consome Pokémon e realiza batalhas (consumidor).
shared_memory.[ch]	Implementa o mecanismo de comunicação entre processos com semáforos e mutexes.
PokemonMonitor.cs	Interface gráfica em C# para visualização em tempo real das estatísticas.

2.2. Diagrama de Comunicação e Memória



2.3. Estrutura da Memória Compartilhada

A estrutura SharedMemory contém:

- Buffer circular de Pokémon (requests[BUFFER_SIZE]);
- Contadores de controle (frente, fim, total);
- Estatísticas do sistema (arenas_ocupadas, trainers_active, total_batalhas, etc.);
- Flag de desligamento seguro (shutdown).

3. Justificativa das Decisões de Projeto

- **Memória Compartilhada + Semáforos**
    - Permite comunicação rápida entre processos sem necessidade de sockets ou arquivos.
    - Facilita o acesso simultâneo com controle fino de concorrência.

- **Mutex para Seções Críticas**
    - Evita acesso simultâneo ao buffer circular.
- **Semáforos “empty” e “full”**
    - Implementam o padrão Produtor–Consumidor, garantindo controle do tamanho do buffer.
- **Interface em C# (WinForms)**
    - Justifica-se pela facilidade de integração com memória compartilhada do Windows e visualização amigável.
- **Controlador Mestre**
    - Responsável por inicializar e finalizar corretamente todos os processos, garantindo limpeza dos recursos.

4. Mecanismos de Sincronização e Evitação de Problemas

4.1. Mecanismos Utilizados

Recurso	Função	Tipo
HANDLE hMutex	Protege o acesso ao buffer e contadores	Mutex
HANDLE hSemEmpty	Indica espaços livres no buffer	Semáforo
HANDLE hSemFull	Indica itens disponíveis no buffer	Semáforo

4.2. Evitação de Problemas

Problema Potencial	Solução Implementada
Condição de Corrida	Uso do WaitForSingleObject(hMutex, ...) em cada acesso crítico.
Deadlock	Ordem consistente de aquisição/liberação dos recursos.
Starvation	Semáforos controlam a alternância justa entre produtores e consumidores.
Finalização Segura	Flag shm->shutdown sinaliza término a todos os processos.

5. Testes Realizados

5.1. Testes de Carga e Concorrência

- Foram executados 5 treinadores e 3 arenas simultaneamente.
- O buffer de tamanho 10 foi constantemente preenchido e esvaziado sem travamentos.

- Testou-se envio rápido de Pokémon (1s) e consumo mais lento (3–5s), garantindo fila circular funcional.

5.2. Encerramento Limpo

- O comando shutdown do controlador define shm->shutdown = 1, encerrando arenas e treinadores de forma cooperativa.
- Handles e views da memória são devidamente fechados (cleanup_shared_memory).

5.3. Testes de Interface Gráfica

- O programa PokemonMonitor.exe exibiu corretamente:
  - Total de batalhas;
  - Número de arenas ocupadas;
  - Pokémon feridos atendidos;
  - Fila dinâmica simulada com dados de exemplo.

Exemplo de execução:

Controlador Mestre do Sistema Pokémon

=== Iniciando Sistema de Expedição Pokémon ===

Arena 0 iniciada (PID: 4512)

Arena 1 iniciada (PID: 4568)

Arena 2 iniciada (PID: 4579)

Treinador 0 iniciado (PID: 4723)

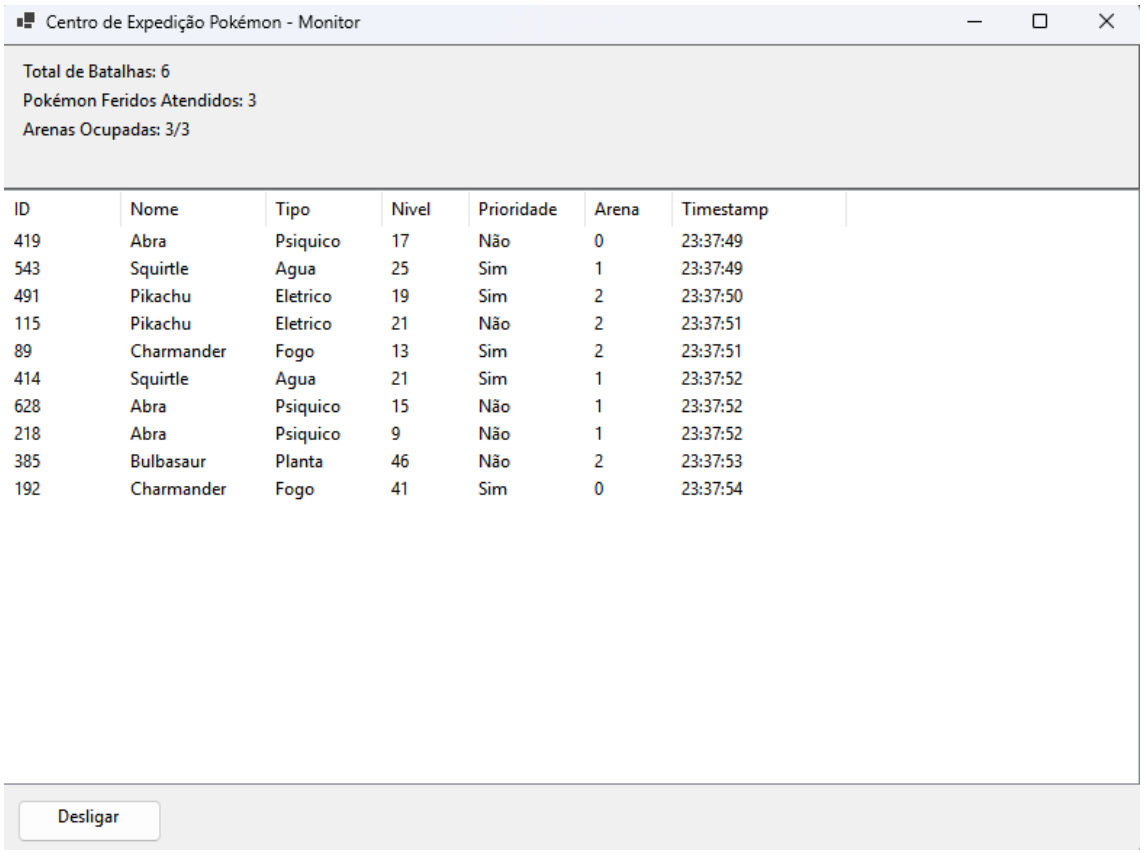
Treinador 1 iniciado (PID: 4744)

...

Captura da Interface Gráfica (exemplo ilustrativo)

A interface exibe:

- Painel superior com estatísticas (total de batalhas, arenas, feridos);
- Tabela central com Pokémon na fila;
- Botão inferior “Desligar Sistema”.



6. Conclusão

O sistema demonstra com sucesso o uso de comunicação interprocessual e sincronização concorrente em ambiente Windows, aplicando o padrão Produtor–Consumidor de forma estável e extensível.

A integração com uma GUI em C# reforça a visibilidade e controle das operações, resultando em um projeto completo e modular, com controle de concorrência robusto e encerramento seguro.