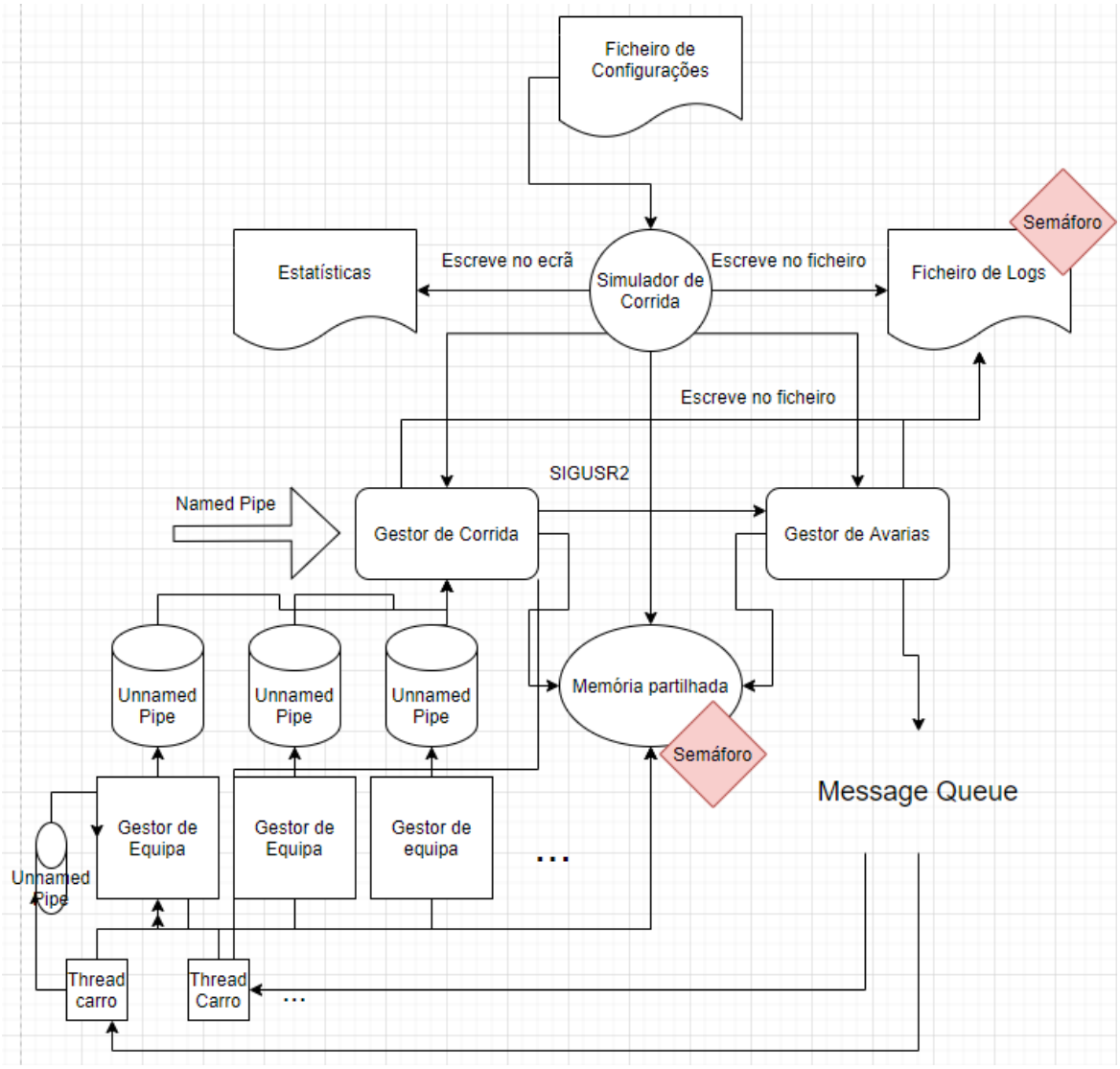

Acima encontra-se a arquitetura do nosso projeto de Sistemas Operativos, já com mecanismos de sincronização implementados.

Temos presentes dois semáforos:

• **Semáforo na memória partilhada** com o objetivo de limitar o acesso à mesma a um processo/thread de cada vez, de forma a otimizar o desempenho do programa aquando da sua execução.

• **Semáforo na escrita nos ficheiros de logs** com o intuito de limitar a quantidade de processos que escrevem neste ficheiro, otimizando assim o programa e assegurando que os dados são escritos na ordem correta.

Quanto às escolhas tomadas aquando da construção do projeto, a nossa perspetiva mudou bastante após a defesa intermédia. Algumas das alterações estão abaixo listadas:

- Uso do sinal SIGUSR2 para alertar o gestor de avarias do começo da corrida, o mesmo encontrando-se em **pause()** até à receção do sinal.
- Criação de um segundo conjunto de pipes de forma a poder estabelecer uma comunicação entre a thread carro e a sua respetiva box, para abastecer ou reparar uma avaria.
- Recurso a variáveis de condição para alertar as threads carro do início de uma corrida, paragem com sucesso na respetiva box e fim da corrida, caso o carro esteja na box, de forma a poder terminar.
- Multiplexing dos unnamed pipes no gestor de corrida de forma a poder receber alterações relativas ao estado de cada carro e assim poder terminar a corrida quando necessário (todos os carros desistiram ou completaram a corrida).

Divisão de tarefas:

De forma a estruturar e otimizar o tempo despendido aquando da realização do projeto, dividimos o projeto em etapas:

Pedro Martins:

- **Simulação da corrida.**
- **Ligação entre processos e threads.**
- **Relatório.**

António Correia:

- **Tratamento de erros.**
- **Ajuda no trabalho do código.**
- **Funções de auxílio.**

O nosso código não possui o SIGUSR1 funcional para interrupção. A estrutura principal encontra-se na mesma no código.