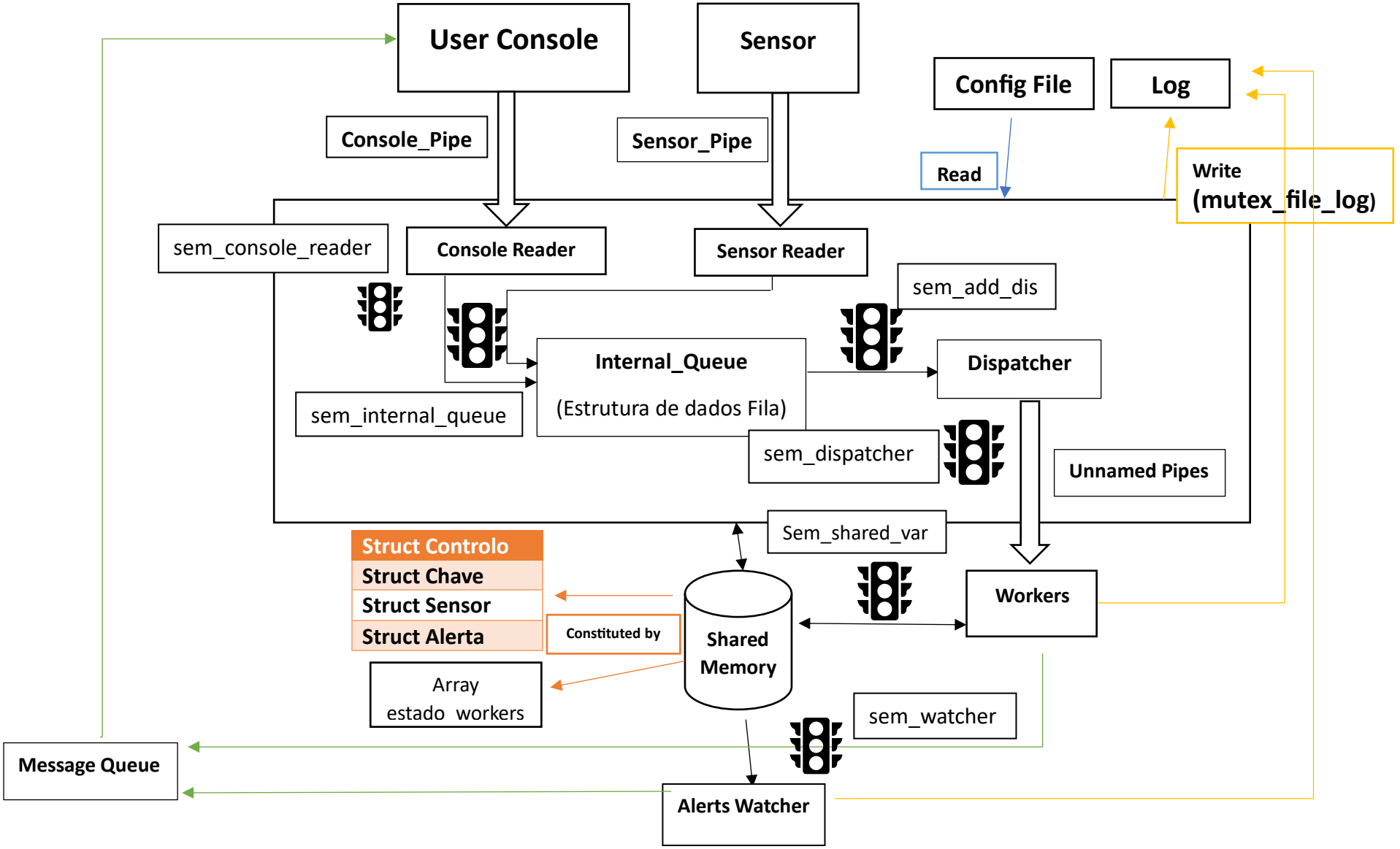
Relatório do Projeto SO



Descrição:

Este relatório, consiste em duas páginas com diferentes orientações, para que seja possível uma melhor visualização do esquema da arquitetura.

De modo a cumprir com o objetivo e regras do trabalho, construiu-se várias soluções e mecanismos para os diferentes constituintes do sistema.

Uma vez que o dispatcher tem que dar prioridade aos pedidos do user console, decidiu-se retirar da internal queue, sempre em primeiro lugar, os pedidos existentes do mesmo. Se se verificar que não existem mais deste tipo de pedidos, então segue-se para o tratamento das mensagens do sensor. A memória partilhada é constituída por memória alocada para armazenar vários tipos de elementos que tenham a necessidade de serem partilhados por todo o sistema. Definiu-se um ponteiro "array_estado_workers" que aponta para um array de inteiros para armazenar o estado de cada worker, como também, um ponteiro "mapa_shared_var" para apontar para uma estrutura Controlo. Dentro dessa estrutura, existem ponteiros para três arrays diferentes: "Sensor", "Chave" e "Alerta".

De maneira a tratar dos pedidos do user console, uma vez que não foi explícita a forma de ser feita, utilizou-se a mesma sintaxe que foi pedida no enunciado para serem tratados as mensagens do sensor. Os dados enviados pelo user console são formatados numa string, com o caracter "c" em primeiro lugar e os dados separados pelo caracter "#" a seguir.

De forma a que não existissem problemas de conflito e concorrência de dados, implementou-se vários mecanismos de sincronização. Foi utilizado um mutex "mutex_file_log", para que os responsáveis por escrever neste ficheiro, não corrompessem os dados. Na shared memory, internal queue, dispatcher, console reader são utilizados semáforos, "sem_shared_var", "sem_internal_queue", "sem_dispatcher" "sem_console_reader", respetivamente, para fazer o devido controlo do fluxo de entrada e alteração de dados no sistema.

O semáforo usado na memória partilhada e internal queue têm como objetivo não permitir que as entidades que armazenam ou escrevem neles, o façam ao mesmo tempo.

O semáforo do dispatcher é utilizado para que não haja busy wait, se todos os workers tiverem com o seu estado ocupado, esta thread aguarda até um Worker ficar disponível.

O semáforo usado no console reader, caso a fila estiver cheia, faz com que esta thread bloqueie até haver espaço livre. Assim, o pipe vai sendo enchido de informação até ser lido posteriormente.

O semáforo "sem_add_dis", uma vez que é inicializado a 0, quando é adicionado um elemento, a função "insert" dá post para incrementar o semáforo, e a seguir é feito wait no dispatcher decrementando o semáforo após a remoção desse elemento da internal queue.

Por fim, o semáforo "sem_watcher" tem como função verificar quando é que existe necessidade do watcher trabalhar (utiliza o mesmo método de funcionamento do semáforo anteriormente descrito), casos como, quando é adicionado um alerta ou adicionado/alterado uma chave.

Autores:

Alexandre Ferreira 2021236702 (Total de tempo despendido: aproximadamente 25h)

João Tinoco 2021223708 (Total de tempo despendido: aproximadamente 25h)