- **Estrutura das mensagens trocadas entre clientes e servidor**: para realizar a troca de mensagens do cliente para o servidor é utilizada uma *struct Request* que tem como atributos o *id* do cliente, o número de lugares que se pretende reservar e a lista de *ids* dos lugares pretendidos.

```
typedef struct
{
        int clientId; // id of the client
        int nSeats; //number seats disered
        int favoriteSeats [MAX_CLI_SEATS-1]; //array with the client's favorite seats
} Request;
```

  Já para a comunicação em sentido inverso é utilizada um *array* de *chars* que contém o número de lugares reservados e a lista dos seus *ids.*

- **Mecanismos de sincronização utilizados**: para realizar a sincronização entre *threads* são utilizados *mutexs* para proteger as secções críticas, tais como, quando um lugar é reservado (para evitar que duas *threads* reservem o mesmo lugar ao mesmo tempo) ou durante a escrita para ficheiros (se várias *threads* escreverem ao mesmo tempo, o ficheiro pode ficar desformatado).

- **Forma como é feito o encerramento do servidor**: as *threads* criadas vão correr enquanto a variável global *stop* for igual a 0 (valor inicial). Após a sua criação, é invocada uma função que irá ler pedidos durante o tempo especificado como argumento (*open_time)* e alterará 0 valor de *stop* para 1, sinalizando às *threads* que estas devem terminar. Quando isto acontecer para todas as *threads*, o programa irá destruir o *FIFO* e o *mutex* criados e irá encerrar.

Henrique José de Castro Ferreira up201605003@fe.up.pt

João Manuel Angélico Gonçalves up201604245@fe.up.pt

Ricardo Araújo Boia up201505244@fe.up.pt