AHDgest

Projeto PCD 2016 ISCTE-IUL

Neste projeto pretende-se que seja desenvolvido um sistema de gestão dos pedidos de aterragem de aviões, a usar, p.ex., no aeroporto General Humberto Delgado. Deverá ser possível executar o sistema como uma aplicação servidor que centraliza a gestão dos pedidos de aterragem. Devem existir clientes locais e distribuídos para lançar os pedidos de aterragem e para visualizar o estado do sistema. O foco do trabalho é na aplicação de programação concorrente e distribuída, sendo que a parte gráfica das aplicações cliente deverá ser considerado um aspeto secundário.

Em termos gerais, deve considerar-se que o aeroporto tem duas pistas, que poderão estar, dada uma delas independentemente da outra, em funcionamento ou encerrada. Quando o *Aeroporto* receber um pedido de aterragem, a *Aeronave* deve esperar até que exista uma vaga numa das pistas em funcionamento. A ordem pela qual as *Aeronaves* recebem uma vaga para aterragem é descrita na secção *Servidor*. Se uma das pistas for encerrada, qualquer avião que esteja em fase de aterragem pode finalizar este ato, mas de seguida não podem ser atribuídas novas vagas para esta pista.

Arquitetura

A solução a desenvolver deverá seguir uma arquitetura cliente-servidor. A Figura 1 ilustra a arquitetura do sistema tendo em conta os principais intervenientes e a funcionalidade elementar (pedido de vaga para aterragem e estado do aeroporto). Deverá existir um *servidor* com o qual diversas aplicações *cliente* comunicam para colocar *Aeronaves* sob gestão do aeroporto, bem como para receber (e consequentemente exibir) o estado do aeroporto. Ao enviar uma *Aeronave* para o servidor, o cliente indica que esta deve ser executada logo que recebida. Esta execução consiste na chamada sequencial de dois métodos do *Aeroporto*: um para pedir uma vaga de aterragem, e, depois de um período de tempo típico, dependente do tipo de aeronave, outro método para assinalar que a aterragem acabou. O servidor mantém em memória uma fila de *Aeronaves* em espera para aterrar, sendo estas geridas de acordo com critérios explicados mais à frente.
A Figura 1 ilustra um cenário de funcionamento com um utilizador *A* e um utilizador *B*, cada um utilizando a uma instância da aplicação cliente.

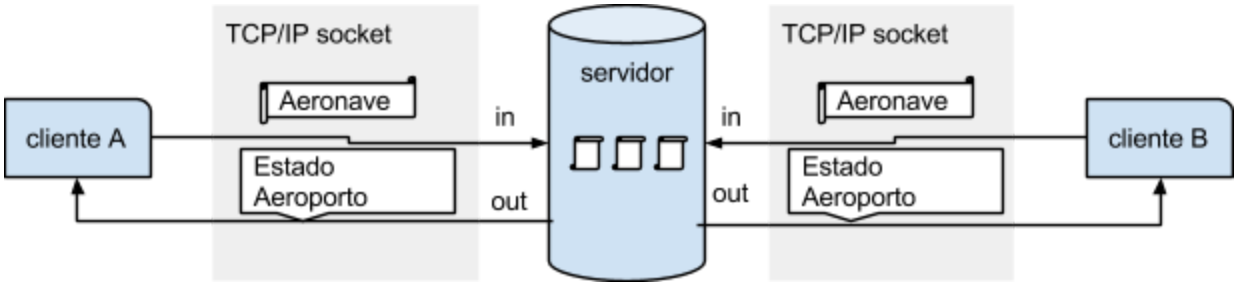


Figura 1. Arquitetura cliente-servidor. Cenário: Ambos os clientes enviam *Aeronaves* para serem executadas no servidor, e recebem informação sobre o estado das *Aeronaves* pendentes.

Servidor

O servidor deverá aceitar pedidos de conexão por parte de clientes, através de uma ligação TCP/IP num porto com um número bem definido (e do conhecimento dos clientes). Ao ser estabelecida uma ligação, os clientes enviam as *Aeronaves* que pretendem que sejam geridas pelo *Aeroporto*. Para efeitos de simplificação, não é exigido nenhum processo de autenticação dos utilizadores. Deve considerar-se que o *Aeroporto* contém apenas duas pistas.

Cada vez que o servidor inicia a execução de uma *Aeronave*, deve registar o momento em que tal acontece.

A atribuição de vagas para aterragem é feita, em primeira prioridade, a qualquer aeronave que tenha menos de 10% da sua capacidade de combustível. A segunda prioridade vai para *Aeronaves* médicas. Finalmente, as vagas devem ser atribuídas por tempo de espera da aeronave.

Ligações

O servidor deverá funcionar em *multi-threading*, tendo para cada ligação ativa duas *threads*, que designaremos por *in* e *out*:

- ***in*** para receção de *Aeronaves* (*input stream*)
  Esta *thread* estará bloqueada à espera de mensagens do cliente. As *Aeronaves* que o servidor recebe deverão ser executadas em *Threads* dedicadas.

- ***out*** para partilha do estado do *Aeroporto* (*output stream*)
  Esta *thread* estará em espera, tornando-se ativa no momento em que existam alterações ao estado do *Aeroporto*, p.ex. a atribuição de vaga a uma *Aeronave*, ou a

finalização da aterragem de uma outra.

Execução e consola de monitorização

O servidor deverá poder ser lançado num processo Java normal, por exemplo executando:

```
java Aeroporto config.txt
```

A partir do momento que o servidor é lançado está disponível para receber ligações de clientes, e cria instâncias de *Aeronave* correspondentes à informação contida no ficheiro de configuração (‘*config.txt*’), cujo formato é descrito mais à frente. Deve ser possível fazer a monitorização do aeroporto e controlar o estado de abertura das pistas do *Aeroporto* através de uma janela, conforme exemplo a seguir. Esta janela pode ter o aspeto que se segue, ou qualquer outro que seja considerado preferível, desde que de fácil e simples utilização.

Cliente

O cliente deverá poder ser lançado num processo Java normal, por exemplo executando:

```
java ClienteAeroporto config.txt
```

Quando o cliente é lançado, deve ligar-se ao servidor, transmitindo-lhe as instâncias de *Aeronave* que correspondam à informação no ficheiro de configuração. Deve então ficar disponível para receber do servidor o estado do *Aeroporto*. Deverá também ser lançada uma interface gráfica a correr localmente, com o componente gráfico descrito mais à frente. Esta interface gráfica deve apenas mostrar a informação do aeroporto.

Tipos de *Aeronave*

Devem existir vários tipos de aeronave: médicas, de passageiros e de carga. Todas têm uma identificação, e caracterizam-se pela capacidade e conteúdo atual do seu depósito.

As aeronaves de carga são caracterizadas pela sua tonelagem, enquanto que as de passageiros são pelo número de passageiros. As *Aeronaves* médicas não têm nenhuma informação particular.

Formato do ficheiro de configuração

Quando quer a aplicação local (lançada simultaneamente com o servidor) quer uma aplicação remota são lançadas, devem poder carregar informação sobre *Aeronaves* de um ficheiro de configuração. Este ficheiro deve conter em cada linha informação sobre uma *Aeronave*, em particular a designação da aeronave, a capacidade do depósito, o conteúdo atual do depósito e o consumo da aeronave, em litros por hora. A designação da aeronave identifica igualmente o tipo desta: se o primeiro caracter for ‘U’, é uma aeronave médica, se for ‘C’ é de carga e se for ‘P’ é de passageiros. As *Aeronaves* de carga têm ainda a tonelagem, enquanto as de passageiros a lotação máxima.

Seguem-se um exemplo de um ficheiro de configuração:

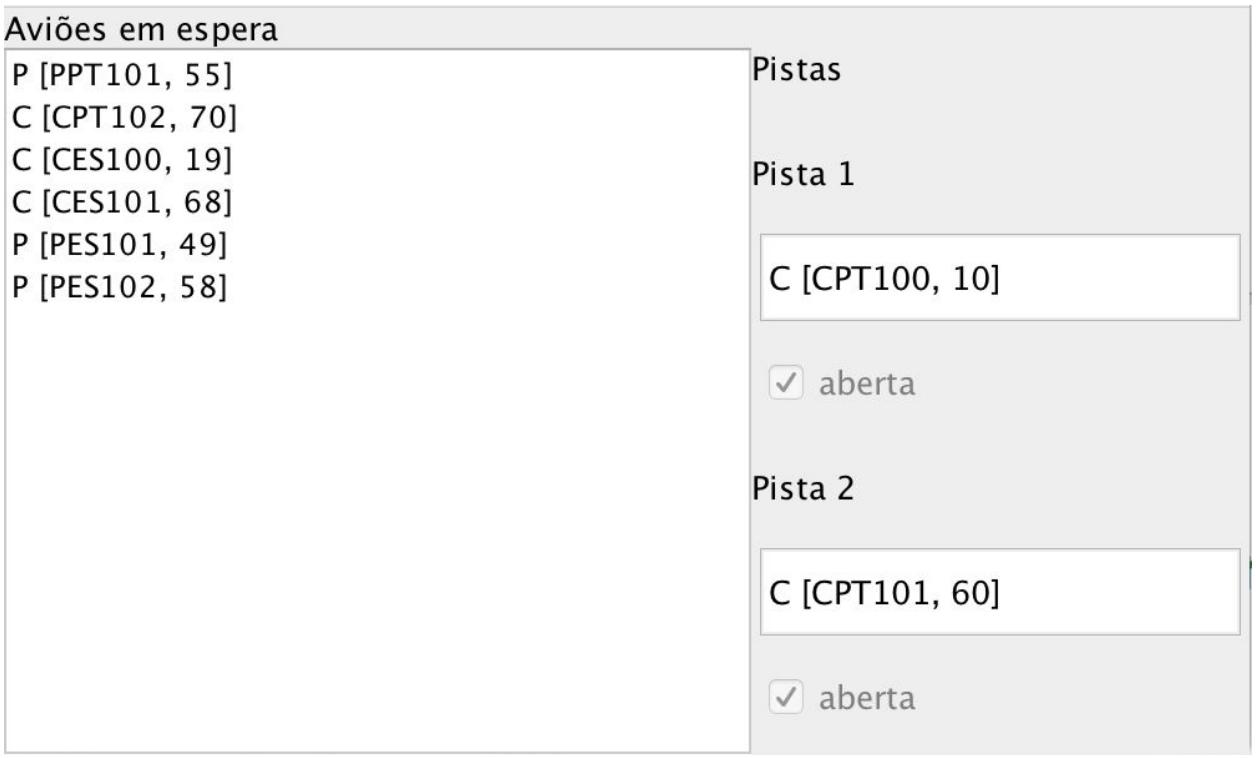
```
UPT100 100 50 5
CPT100 100 11 5 20
PPT100 100 5 5 120
```

Componente gráfica de monitorização

Deve usar o mesmo componente quer nas janelas do servidor ou dos clientes remotos para mostrar o estado do aeroporto. Para isso deve criar um componente gráfico com os seguintes elementos:

- uma lista das *Aeronaves* em espera,
- a identificação das *Aeronave* atualmente em cada uma das pistas.
- o estado de abertura de cada uma das pistas.

O aspecto geral desta componente deve ser como se segue:



Fases, Avaliação, e Entrega

São propostas as seguintes fases de desenvolvimento como metas para a avaliação, e de forma a que seja mais fácil abordar o problema:

Fase 1: Desenvolver uma versão básica da interface gráfica, sendo nela colocadas *Aeronaves* sem nenhum tipo de gestão ou critério.

Fase 2: Desenvolver o servidor, suportando a apenas o funcionamento local.

Fase 3: Desenvolver o cliente de monitorização remota.

Fase 4: Evoluir a solução para o cliente para suportar envio de *Aeronaves* para o servidor.

As notas do projeto serão atribuídas de acordo com a realização das fases propostas:

A: completar todas as fases
B: completar as fases (1, 2, 3) ou (1, 2, 4)
C: competar as fases (1, 2)
D: não são cumpridos os requisitos mínimos (reprovação à UC)

Grupos: Cada grupo de trabalho é composto por dois alunos, preferencialmente da mesma turma prática.

Entrega Intercalar: Para a entrega intercalar é necessário terminar a fase 1 e a leitura dos ficheiros. A demonstração será feita na aula prática a partir de um JAR.

Entrega Final: O projeto desenvolvido deve ser entregue sob a forma de um projeto arquivado usando a funcionalidade de *Export/Archive File* do Eclipse. As entregas serão feitas no e-learning até às 12h de dia 13 de Dezembro. Os grupos deverão comparecer na última semana à aula prática onde estão inscritos para realizarem a discussão do trabalho.