Mestrado em Engenharia Informática

PROGRAMAÇÃO CONCORRENTE E DISTRIBUÍDA

Trabalho Prático de

Programação Distribuida

a15060 - Luís Castro

a15061 - Filipe Ribeiro

Julho de 2018

Introdução e Objetivos

Com este trabalho pretende-se implementar um programa em JAVA seguindo o paradigma do modelo de Actores utilizando a biblioteca Akka[1].

O problema consiste num problema de Urban Manufacturing onde dado uma lista de viagens que se pretende realizar e uma lista de fornecedores, o objetivo é conseguir realizar essas mesmas viagens analisando os custos de cada fornecedor e escolher preferencialmente o que oferece um melhor custo.

Ao analisar o problema foi necessário identificar todos os atores que pertencem ao modelo a ser implementado bem como as trocas de mensagens entre os mesmos de forma a realizar cada uma das viagens.

O presente relatório apresenta e descreve todo o processo da construção da arquitetura bem como os resultados e análise das execuções pretendidas.

[1] https://akka.io/

Detalhes da Arquitetura da Solução

Descrição da arquitectura da solução (identificação dos actores, das mensagens e do comportamento dos mesmos, assim como o papel da função main).

Para implementar a solução pretendida foram criados um total de três atores, “MaganerActor”, “ClientActor” e, “ProviderActor”.

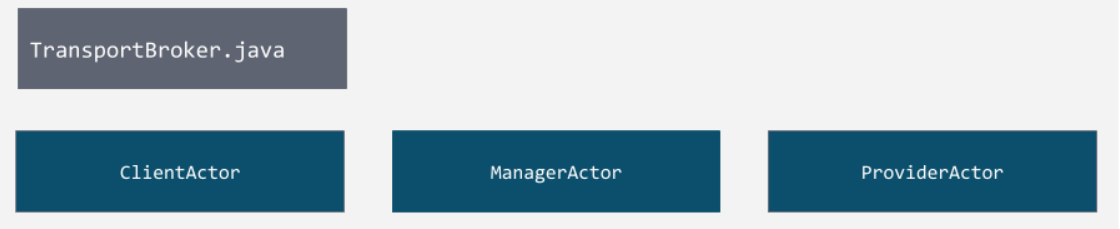

Fig. 1 - Actores criados

Ao executar a aplicação, a função main cria um novo sistema de atores, carregando todos os dados provenientes dos ficheiros JSON.

De seguida cria dois arrays de actores, um array com atores fornecedores com base nos respetivos ficheiros, contendo os diferentes custos para a realização das viagens e um array de atores clientes com base no total de clientes disponíveis. De seguida cria também um ator gestor (manager) contendo todos os dados criados anteriormente e as viagens a realizar. Por fim envia uma mensagem para o ator gestor para este iniciar a sua execução, e consecutivamente todo o processo.

Todo o processo inicial está descrito no estrato de código abaixo

```
// -- -- -- Create actor system
ActorSystem system = ActorSystem.create("transport-broker-akka");

// -- -- -- Create all provider actors
ActorRef[] providersActors = new ActorRef[providers.length];
for (int i = 0; i < providers.length; i++) {
  providersActors[i] = system.actorOf(Props.create(ProviderActor.class, ...));
}
```

```
// -- -- -- Create all clients actors
ActorRef[] clientActors = new ActorRef[NUMBER_OF_CLIENTS];
for (int i = 0; i < NUMBER_OF_CLIENTS; i++) {
  clientActors[i] = system.actorOf(Props.create(ClientActor.class, ...));
}

// -- -- -- Create manager actor
ActorRef managerActor = system.actorOf(Props.create(ManagerActor.class, ...);
managerActor.tell(new StartMessage(), ActorRef.noSender());
```

Todo o processo da realização das viagens é feito entre os três atores criados. A figura abaixo (Fig.3), mostra todas as trocas de mensagens existentes entre os diferentes atores.

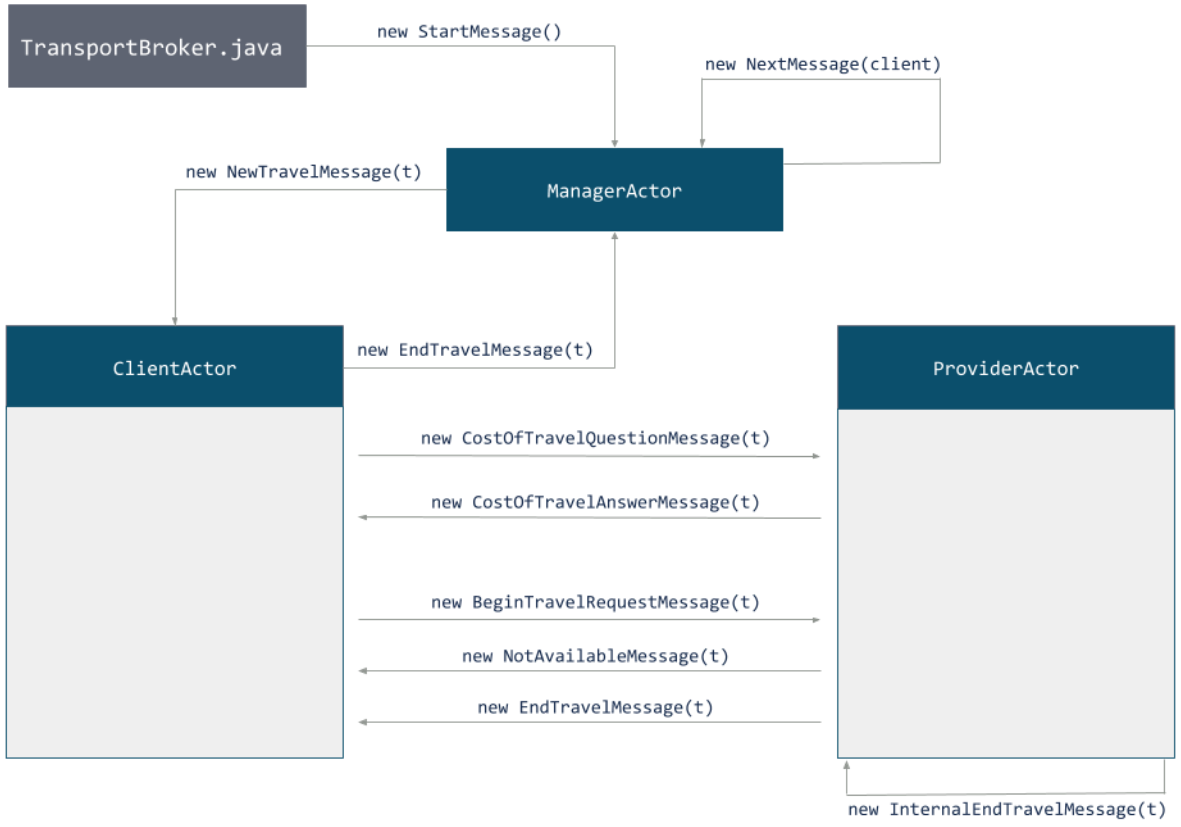

Fig. 2 - Fluxo de mensagens entre os todos os atores.

Ao iniciar o ator *manager* (*StartMessage()*), notifica todos os clientes disponíveis de uma nova viagem que este tem de realizar (*NewTravelMessage()*). Ao receber a viagem o cliente envia a todos os fornecedores disponíveis um pedido de custo para realizar essa mesma viagem, (*CostOfTravelQuestionMessage()*). Ao receber a resposta por parte do

fornecedor (*CostOfTravelAnswerMessage()*), internamente o cliente guarda a informação da viagem, o custo e o respetivo fornecedor num *TreeMap* ordenado pelo custo de forma a ter os fornecedores ordenados, do mais barato para o mais caro.

Após ter a informação do custo de todos os fornecedores, o cliente seleciona o que oferece o custo mais baixo e efetua um pedido para realizar a viagem (*BeginTravelRequestMessage()*). Ao receber o pedido, o fornecedor pode ter dois comportamentos distintos. Caso o fornecedor esteja ocupado com uma outra viagem, este notifica o cliente que não está disponível de momento (*NotAvailableMessage()*), o cliente ao receber a mensagem executa novamente o pedido para efetuar a viagem até que o fornecedor esteja disponível. Com este comportamento é dado prioridade ao custo da viagem, ficando à espera até que seja possível a sua realização por parte do fornecedor. Por outro lado, caso o fornecedor esteja disponível inicia a viagem, ficando indisponível no tempo da sua duração.

Ao iniciar uma viagem o fornecedor muda o seu estado para indisponivel, e agenda uma mensagem para ele mesmo, (*InternalEndTravelMessage()*), com base no tempo que demora a viagem. A forma encontrada para conseguir ocupar e desocupar o fornecedor foi recorrendo a um *scheduler*[2] disponível na biblioteca *Akka:*

```
context().system().scheduler().scheduleOnce(new FiniteDuration(t.time,
TimeUnit.SECONDS), getSelf(), new InternalEndTravelMessage(t,getSender()),
context().dispatcher(), ActorRef.noSender());
```

Ao receber essa mesma mensagem, altera novamente o seu estado para disponível, podendo assim efetuar mais viagens, e envia uma mensagem para o cliente (*EndTravelMessage(t)*) indicando-lhe que a viagem terminou.

Após concluir a viagem o cliente envia uma mensagem para o *manager* indicando que a viagem que lhe foi associada foi concluída com sucesso, este ao receber envia uma mensagem para ele mesmo (*NextTravelMessage()*) com o identificador do cliente que, caso existam mais viagens inicia novamente todo o processo.

Quando todas as viagem forem realizadas, o manager apresenta na consola os detalhes de cada viagem realizada e termina a execução da aplicação.

[2] https://doc.akka.io/docs/akka/2.5/scheduler.html

Resultados de Execução

Considere-se os seguintes cenários de simulação:

Execução I: Um único agente cliente, três agentes fornecedores (A, B, C). O agente cliente simula uma sequência de pedidos de viagem (conforme lista anterior de 15 viagens), contactando e escolhendo o melhor dos três fornecedores para cada viagem. As viagens têm duração nula neste
ponto.

Nesta execução, para cada viagem apenas foi recolhido a informação do identificador do cliente, os detalhes da viagem, o fornecedor que oferece o melhor custo e o respetivo valor.

```
$
----------------------------------------------------------------------
Cliente ID 0 - Viagem: 1 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 2 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 3 (barcelos, braga) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 4 (braga, braga) - Cost: 3 (Fornecedor B)
Cliente ID 0 - Viagem: 5 (braga, porto) - Cost: 10 (Fornecedor B)
Cliente ID 0 - Viagem: 6 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 0 - Viagem: 7 (braga, lisboa) - Cost: 25 (Fornecedor C)
Cliente ID 0 - Viagem: 8 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 9 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 10 (lisboa, lisboa) - Cost: 5 (Fornecedor C)
Cliente ID 0 - Viagem: 11 (lisboa, porto) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 12 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 0 - Viagem: 13 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 0 - Viagem: 14 (porto, barcelos) - Cost: 12 (Fornecedor A)
Cliente ID 0 - Viagem: 15 (porto, braga) - Cost: 10 (Fornecedor B)
```

Ao analisar o resultado obtido é possível ver quais os fornecedores escolhidos para efetuar cada uma das viagens.

Execução II: Dois agentes clientes. Com base num ficheiro de texto, cujo conteúdo será a lista de viagens anteriores repetida quatro vezes, num total de 60 viagens, cada agente cliente executa uma viagem desta lista, sendo todas as viagens repartidas pelos dois agentes. As viagens têm duração nula neste ponto.

Para esta execução é possível escolher número de clientes pretendidos, para esta demonstração foi utilizado apenas 2 clientes. Para um total de 60 viagens, o resultado da execução consiste em:

```
$ Número de Clientes? 2
------------------------------------------------------------------------
Cliente ID 0 - Viagem: 1 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 1 - Viagem: 2 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 1 - Viagem: 3 (barcelos, braga) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 4 (braga, braga) - Cost: 3 (Fornecedor B)
Cliente ID 1 - Viagem: 5 (braga, porto) - Cost: 10 (Fornecedor B)
Cliente ID 0 - Viagem: 6 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 1 - Viagem: 7 (braga, lisboa) - Cost: 25 (Fornecedor C)
Cliente ID 0 - Viagem: 8 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 1 - Viagem: 9 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 10 (lisboa, lisboa) - Cost: 5 (Fornecedor C)
Cliente ID 1 - Viagem: 11 (lisboa, porto) - Cost: 15 (Fornecedor C)
Cliente ID 1 - Viagem: 12 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 1 - Viagem: 13 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 1 - Viagem: 14 (porto, barcelos) - Cost: 12 (Fornecedor A)
Cliente ID 1 - Viagem: 15 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 1 - Viagem: 16 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 1 - Viagem: 17 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 1 - Viagem: 18 (barcelos, braga) - Cost: 3 (Fornecedor A)
Cliente ID 1 - Viagem: 19 (braga, braga) - Cost: 3 (Fornecedor B)
Cliente ID 1 - Viagem: 20 (braga, porto) - Cost: 10 (Fornecedor B)
Cliente ID 1 - Viagem: 21 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 0 - Viagem: 22 (braga, lisboa) - Cost: 25 (Fornecedor C)
Cliente ID 1 - Viagem: 23 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 24 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 1 - Viagem: 25 (lisboa, lisboa) - Cost: 5 (Fornecedor C)
Cliente ID 0 - Viagem: 26 (lisboa, porto) - Cost: 15 (Fornecedor C)
Cliente ID 1 - Viagem: 27 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 0 - Viagem: 28 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 1 - Viagem: 29 (porto, barcelos) - Cost: 12 (Fornecedor A)
Cliente ID 0 - Viagem: 30 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 1 - Viagem: 31 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 32 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 1 - Viagem: 33 (barcelos, braga) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 34 (braga, braga) - Cost: 3 (Fornecedor B)
Cliente ID 0 - Viagem: 35 (braga, porto) - Cost: 10 (Fornecedor B)
Cliente ID 0 - Viagem: 36 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 0 - Viagem: 37 (braga, lisboa) - Cost: 25 (Fornecedor C)
Cliente ID 1 - Viagem: 38 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 39 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 1 - Viagem: 40 (lisboa, lisboa) - Cost: 5 (Fornecedor C)
Cliente ID 0 - Viagem: 41 (lisboa, porto) - Cost: 15 (Fornecedor C)
Cliente ID 1 - Viagem: 42 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 0 - Viagem: 43 (porto, porto) - Cost: 3 (Fornecedor C)
```

```
Cliente ID 1 - Viagem: 44 (porto, barcelos) - Cost: 12 (Fornecedor A)
Cliente ID 0 - Viagem: 45 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 1 - Viagem: 46 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 47 (barcelos, barcelos) - Cost: 3 (Fornecedor A)
Cliente ID 1 - Viagem: 48 (barcelos, braga) - Cost: 3 (Fornecedor A)
Cliente ID 0 - Viagem: 49 (braga, braga) - Cost: 3 (Fornecedor B)
Cliente ID 1 - Viagem: 50 (braga, porto) - Cost: 10 (Fornecedor B)
Cliente ID 0 - Viagem: 51 (porto, braga) - Cost: 10 (Fornecedor B)
Cliente ID 1 - Viagem: 52 (braga, lisboa) - Cost: 25 (Fornecedor C)
Cliente ID 0 - Viagem: 53 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 1 - Viagem: 54 (porto, lisboa) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 55 (lisboa, lisboa) - Cost: 5 (Fornecedor C)
Cliente ID 1 - Viagem: 56 (lisboa, porto) - Cost: 15 (Fornecedor C)
Cliente ID 0 - Viagem: 57 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 1 - Viagem: 58 (porto, porto) - Cost: 3 (Fornecedor C)
Cliente ID 1 - Viagem: 59 (porto, barcelos) - Cost: 12 (Fornecedor A)
Cliente ID 1 - Viagem: 60 (porto, braga) - Cost: 10 (Fornecedor B)
```

Execução III: Agentes fornecedores ficam ocupados durante a duração da viagem. Simulação idêntica à anterior, mas quando um fornecedor é solicitado para uma deslocação, fica ocupado pelo tempo de duração dessa deslocação (que varia entre 1 segundo e 8 segundos, dependendo dos locais). Enquanto ocupado, pode responder a pedidos de custo mas não pode realizar serviços a outros clientes.

Para demonstrar a execução foi utilizado as viagens da Execução I, sendo esta mais fácil de demonstrar mas funciona de igual forma para as viagens da Execução II.

Para conseguir demonstrar a duração, para cada viagem foi registado a hora de início e a hora de fim da sua execução, podemos assim ver que a diferença de tempo corresponde ao tempo de duração de cada uma das viagens.

```
01:26:48 Viagem 1 - Fornecedor A - Custo: 3 - 1s ... 01:26:49 End
01:26:49 Viagem 2 - Fornecedor A - Custo: 3 - 1s ... 01:26:50 End
01:26:50 Viagem 3 - Fornecedor A - Custo: 3 - 2s ... 01:26:52 End
01:26:52 Viagem 4 - Fornecedor B - Custo: 3 - 1s ... 01:26:53 End
01:26:53 Viagem 5 - Fornecedor B - Custo: 10 - 2s ... 01:26:55 End
01:26:55 Viagem 6 - Fornecedor B - Custo: 10 - 2s ... 01:26:57 End
01:26:57 Viagem 7 - Fornecedor C - Custo: 25 - 6s ... 01:27:03 End
01:27:03 Viagem 8 - Fornecedor C - Custo: 15 - 4s ... 01:27:07 End
01:27:07 Viagem 9 - Fornecedor C - Custo: 15 - 4s ... 01:27:11 End
01:27:11 Viagem 10 - Fornecedor C - Custo: 5 - 1s ... 01:27:12 End
01:27:12 Viagem 11 - Fornecedor C - Custo: 15 - 4s ... 01:27:16 End
01:27:16 Viagem 12 - Fornecedor C - Custo: 3 - 1s ... 01:27:17 End
```

```
01:27:17 Viagem 13 - Fornecedor C - Custo: 3 - 1s ... 01:27:18 End
01:27:18 Viagem 14 - Fornecedor A - Custo: 12 - 4s ... 01:27:22 End
01:27:22 Viagem 15 - Fornecedor B - Custo: 10 - 2s ... 01:27:24 End
```

Em seguida é apresentado o resultado utilizando as 60 viagens da execução II utilizando dois clientes, que tal como o execução anterior este valor ser dinâmico.

```
$Número de Clientes? 2
19:30:55 Viagem 2 - Fornecedor A - Custo: 3 - 1s ... 19:30:56 End
19:30:56 Viagem 1 - Fornecedor A - Custo: 3 - 1s ... 19:30:57 End
19:30:57 Viagem 3 - Fornecedor A - Custo: 3 - 2s ...
19:30:57 Viagem 4 - Fornecedor B - Custo: 3 - 1s ... 19:30:58 End
19:30:58 Viagem 5 - Fornecedor B - Custo: 10 - 2s ... 19:30:59 End
19:31:00 End
19:31:00 Viagem 6 - Fornecedor B - Custo: 10 - 2s ...
19:31:00 Viagem 7 - Fornecedor C - Custo: 25 - 6s ... 19:31:02 End
19:31:06 End
19:31:06 Viagem 8 - Fornecedor C - Custo: 15 - 4s ... 19:31:10 End
19:31:10 Viagem 9 - Fornecedor C - Custo: 15 - 4s ... 19:31:14 End
19:31:14 Viagem 11 - Fornecedor C - Custo: 15 - 4s ... 19:31:18 End
19:31:18 Viagem 10 - Fornecedor C - Custo: 5 - 1s ... 19:31:19 End
19:31:19 Viagem 12 - Fornecedor C - Custo: 3 - 1s ... 19:31:20 End
19:31:20 Viagem 13 - Fornecedor C - Custo: 3 - 1s ...
19:31:20 Viagem 14 - Fornecedor A - Custo: 12 - 4s ... 19:31:21 End
19:31:21 Viagem 15 - Fornecedor B - Custo: 10 - 2s ... 19:31:23 End
19:31:24 End
19:31:24 Viagem 16 - Fornecedor A - Custo: 3 - 1s ... 19:31:25 End
19:31:25 Viagem 17 - Fornecedor A - Custo: 3 - 1s ... 19:31:26 End
19:31:26 Viagem 19 - Fornecedor B - Custo: 3 - 1s ...
19:31:26 Viagem 18 - Fornecedor A - Custo: 3 - 2s ... 19:31:27 End
19:31:27 Viagem 20 - Fornecedor B - Custo: 10 - 2s ... 19:31:28 End
19:31:29 End
19:31:29 Viagem 21 - Fornecedor B - Custo: 10 - 2s ...
19:31:29 Viagem 22 - Fornecedor C - Custo: 25 - 6s ... 19:31:31 End
19:31:35 End
19:31:35 Viagem 23 - Fornecedor C - Custo: 15 - 4s ... 19:31:39 End
19:31:39 Viagem 24 - Fornecedor C - Custo: 15 - 4s ... 19:31:43 End
19:31:43 Viagem 25 - Fornecedor C - Custo: 5 - 1s ... 19:31:44 End
19:31:44 Viagem 26 - Fornecedor C - Custo: 15 - 4s ... 19:31:48 End
19:31:48 Viagem 27 - Fornecedor C - Custo: 3 - 1s ... 19:31:49 End
19:31:49 Viagem 28 - Fornecedor C - Custo: 3 - 1s ...
19:31:49 Viagem 29 - Fornecedor A - Custo: 12 - 4s ... 19:31:50 End
19:31:50 Viagem 30 - Fornecedor B - Custo: 10 - 2s ... 19:31:52 End
19:31:53 End
19:31:53 Viagem 31 - Fornecedor A - Custo: 3 - 1s ... 19:31:54 End
19:31:54 Viagem 32 - Fornecedor A - Custo: 3 - 1s ... 19:31:55 End
19:31:55 Viagem 33 - Fornecedor A - Custo: 3 - 2s ...
19:31:55 Viagem 34 - Fornecedor B - Custo: 3 - 1s ... 19:31:56 End
```

```
19:31:56 Viagem 35 - Fornecedor B - Custo: 10 - 2s ... 19:31:57 End
19:31:58 End
19:31:58 Viagem 36 - Fornecedor B - Custo: 10 - 2s ...
19:31:58 Viagem 37 - Fornecedor C - Custo: 25 - 6s ... 19:32:00 End
19:32:04 End
19:32:04 Viagem 38 - Fornecedor C - Custo: 15 - 4s ... 19:32:08 End
19:32:08 Viagem 39 - Fornecedor C - Custo: 15 - 4s ... 19:32:12 End
19:32:12 Viagem 40 - Fornecedor C - Custo: 5 - 1s ... 19:32:13 End
19:32:13 Viagem 41 - Fornecedor C - Custo: 15 - 4s ... 19:32:18 End
19:32:18 Viagem 42 - Fornecedor C - Custo: 3 - 1s ... 19:32:19 End
19:32:19 Viagem 43 - Fornecedor C - Custo: 3 - 1s ...
19:32:19 Viagem 44 - Fornecedor A - Custo: 12 - 4s ... 19:32:20 End
19:32:20 Viagem 45 - Fornecedor B - Custo: 10 - 2s ... 19:32:22 End
19:32:23 End
19:32:23 Viagem 46 - Fornecedor A - Custo: 3 - 1s ... 19:32:24 End
19:32:24 Viagem 47 - Fornecedor A - Custo: 3 - 1s ... 19:32:25 End
19:32:25 Viagem 48 - Fornecedor A - Custo: 3 - 2s ...
19:32:25 Viagem 49 - Fornecedor B - Custo: 3 - 1s ... 19:32:26 End
19:32:26 Viagem 50 - Fornecedor B - Custo: 10 - 2s ... 19:32:27 End
19:32:28 End
19:32:28 Viagem 51 - Fornecedor B - Custo: 10 - 2s ...
19:32:28 Viagem 52 - Fornecedor C - Custo: 25 - 6s ... 19:32:30 End
19:32:34 End
19:32:34 Viagem 53 - Fornecedor C - Custo: 15 - 4s ... 19:32:38 End
19:32:38 Viagem 54 - Fornecedor C - Custo: 15 - 4s ... 19:32:42 End
19:32:42 Viagem 55 - Fornecedor C - Custo: 5 - 1s ... 19:32:43 End
19:32:43 Viagem 56 - Fornecedor C - Custo: 15 - 4s ... 19:32:47 End
19:32:47 Viagem 57 - Fornecedor C - Custo: 3 - 1s ... 19:32:48 End
19:32:48 Viagem 58 - Fornecedor C - Custo: 3 - 1s ...
19:32:48 Viagem 59 - Fornecedor A - Custo: 12 - 4s ... 19:32:49 End
19:32:49 Viagem 60 - Fornecedor B - Custo: 10 - 2s ... 19:32:51 End
19:32:52 End
```

Execução IV: Os agentes fornecedores usam a informação sobre o custo das viagens usando uma tabela em ficheiro de texto, carregado no início da simulação. A restante simulação é idêntica aos pontos anteriores.

Todos os custos associados a cada fornecedor estão a ser carregados através de ficheiros JSON. Foi criado um ficheiro para cada fornecedor composto por um título, sendo este o nome do fornecedor e um objeto *data,* contendo as diferentes combinações das vagens que este realiza e o respetivo valor.

```
{
  "title" : "Fornecedor A",
```

```
  "data": {
    "barcelos": {
      "barcelos": 3,
      "braga": 3,
      "porto": 12,
      "lisboa": 33
    },
    "braga": {
      "barcelos": 3,
      "braga": 5,
      "porto": 12,
      "lisboa": 26
    },
    "porto": {
      "barcelos": 12,
      "braga": 12,
      "porto": 10,
      "lisboa": 24
    },
    "lisboa": {
      "barcelos": 33,
      "braga": 26,
      "porto": 24,
      "lisboa": 20
    }
  }
}
```

Para além dos fornecedores, a informação dos tempos das viagens também são carregados através de um ficheiro JSON.

```
{
  "title": "time costs",
  "data": {
    "barcelos": {
      "barcelos": 1,
      "braga": 2,
      "porto": 4,
      "lisboa": 8
    },
    "braga": {
      "barcelos": 2,
      "braga": 1,
      "porto": 2,
      "lisboa":6
    },
```

```
    "porto": {
      "barcelos": 4,
      "braga": 2,
      "porto": 1,
      "lisboa":4
    },
    "lisboa": {
      "barcelos": 8,
      "braga": 6,
      "porto": 4,
      "lisboa": 1
    }
  }
}
```

Conclusão

A utilização da biblioteca Akka foi um verdadeiro desafio mas recompensante após compreender o funcionamento e o fluxo do modelo de atores que este implementa e as vantagens que trás ao modelo utilizando apenas threads.

Após identificar todos os atores necessários para implementar a solução, estruturar e implementar todas as mensagens foi um processo delicado.

Apesar de serem implementadas todas as execuções que eram propostas, uma otimização que poderia ser implementada seria na decisão que um cliente toma ao receber a informação que o fornecedor desejado está ocupado. Foi implementado seguindo o pensamento de escolher obrigatoriamente o menor custo possível, uma otimização interessante seria definir uma diferença máxima no custo da viagem para o segundo fornecedor, por exemplo, se definirmos uma diferença máxima de 2, caso o fornecedor com menor custo seja 5 se o segundo fornecedor oferecer um custo até 7, o cliente aceita fazer a viagem com o segundo caso o primeiro esteja ocupado.