

UNIVERSIDADE DE SÃO PAULO

Escola Politécnica da USP

PCS5703 – Exercício Prático 1

Criando Sistemas Multiagentes

Profa. Dra. Anarosa Alves Franco Brandão

06/10/2020

O objetivo deste exercício é familiarizar o aluno com os aspectos de desenvolvimento em JAVA usando Serviços REST e programação em Sistemas Multiagentes. O exercício deverá ser feito de forma **individual**, com entrega até o dia 19/10/2020 através do GitHub.

1 Requisitos:

Os alunos deverão ter instalado, em seu computador:

- Java SDK versão 1.8 ou superior
- Um ambiente de desenvolvimento JAVA (IDE) de sua preferência
- Ferramentas necessárias para submeter o código via GitHub, tais como: Git client, GitHub client, GitKraken, ou outras similares

2 Contexto:

Para este exercício prático, os alunos irão utilizar o arcabouço de sistemas multiagentes **Hecate**. Diferentemente dos outros arcabouços vistos em aula, o Hecate não requer nenhuma dependência local, uma vez que todos os agentes e artefatos são criados remotamente.

Durante a realização deste exercício, os alunos irão utilizar uma versão modificada do Hecate para fins didáticos. Um arquivo JAR será disponibilizado via GitHub, e os alunos deverão implementar um sistema local que reflita o cenário proposto (abaixo).

3 Cenário:

Um aluno de pós-graduação da Poli implementou um arcabouço para desenvolvimento de sistemas multiagentes chamado Hecate. Como forma de validar o sistema desenvolvido, sua orientadora pediu que outros alunos de pós-graduação implementassem um sistema multiagentes simples utilizando o Hecate. No entanto, tendo em vista que a primeira versão do Hecate não possuía alguns mecanismos de controle, a professora pediu que os alunos implementassem o sistema em duas partes: um **ambiente remoto**, hospedado dentro do Hecate, e um **ambiente local**, hospedado na máquina do aluno.

Os requisitos de cada parte deste sistema encontram-se abaixo:

3.1 Ambiente Remoto

- Todas as salas do ambiente devem ser monitoradas por seus ocupantes. Desta forma, toda vez que um agente sair de uma sala deverá enviar uma mensagem para seus ocupantes no seguinte formato: "AGENTE jnome do agentej, SAIU DA SALA jnome da sala;"
- Similarmente, ao ingressar em uma sala nova, um agente deverá enviar uma mensagem para seus ocupantes no seguinte formato: "AGENTE jnome do agentej, ENTROU NA SALA jnome da sala;"
- Nenhuma sala pode possuir menos que dois agentes, ou mais que quatro agentes a qualquer momento.

3.2 Ambiente Local

- O ambiente local deve possuir dois mecanismos de controle distintos: um referente à distribuição dos agentes entre as salas, e outro referente ao monitoramento das mensagens trocadas entre os agentes.
- O mecanismo de distribuição de agentes deve ser capaz de mover os agentes entre as salas, **garantindo que cada agente só esteja em uma sala de cada vez.**
- O mecanismo de distribuição de agentes deve ser capaz de registrar as salas inicialmente atribuídas a cada agente.
- O mecanismo de distribuição de agentes deve ser capaz de garantir as limitações de ocupação de sala, ou seja, garantir que a cada momento nenhuma sala tenha mais de quatro agentes ou menos de dois agentes.
- O mecanismo de monitoramento de mensagens deve ser capaz de produzir um registro consolidado de todas as mensagens trocadas entre os agentes, ordenadas da mais antiga para a mais nova

Tendo em mente os requisitos acima listados, pede-se:

- O aluno deve criar, no ambiente remoto, **cinco** salas distintas

- O aluno deverá criar **quinze** agentes distintos, atribuindo **três** agentes por sala
- Uma vez criado o ambiente inicial, o aluno deverá mover os agentes entre as salas, de forma que:
 - + Cada agente deverá se mover para uma nova sala, diferente da sala a que foi originalmente atribuído
 - + A nova sala de destino para o agente deverá ser escolhida **de forma aleatória** entre as salas restantes
 - + Todos os agentes devem mudar de sala
 - + Após todos os agentes mudarem de sala, **apenas uma das salas poderá ter três agentes**. As outras salas deverão possuir dois ou quatro agentes.
 - + Após todos os agentes mudarem de sala, todas as mensagens trocadas entre os agentes deverão ser listadas de forma cronológica, em uma única lista (contendo todas as mensagens enviadas por todos os agentes). Esta lista deverá conter apenas entradas únicas: considerando que cada agente gera somente duas mensagens a cada troca de sala (uma de saída e uma de entrada) e se todos os agentes trocam de sala uma e só uma vez, espera-se que a lista contenha trinta mensagens distintas
 - + A lista de mensagens deve ser salvo em um arquivo de texto chamado "mensagens.txt"

4 Entregáveis:

- Código-fonte completo do ambiente local, incluindo instruções de execução
- Arquivo "mensagens.txt" produzido

5 Documentação técnica:

Para este exercício, o arcabouço Hecate utiliza três estruturas principais: **agentes**, **salas** e **mensagens**. Cada uma dessas estruturas pode ser vista como um objeto (classe) com as seguintes propriedades:

Agente:

- ID: Um número de identificação único no sistema
- Name: Um identificador descritivo (texto)
- Uma lista (coleção) de mensagens

Sala:

- ID: Um número de identificação único no sistema
- Name: Um identificador descritivo (texto)

- Uma lista (coleção) de agentes ocupando a sala em um dado momento
- Uma lista (coleção) de mensagens

Mensagem:

- From: Agente que enviou a mensagem
- To: Agente a quem foi destinada a mensagem
- Room: Sala a que foi destinada a mensagem
- Message: O texto contido na mensagem
- LocalDateTime: Horário (timestamp) em que a mensagem foi enviada

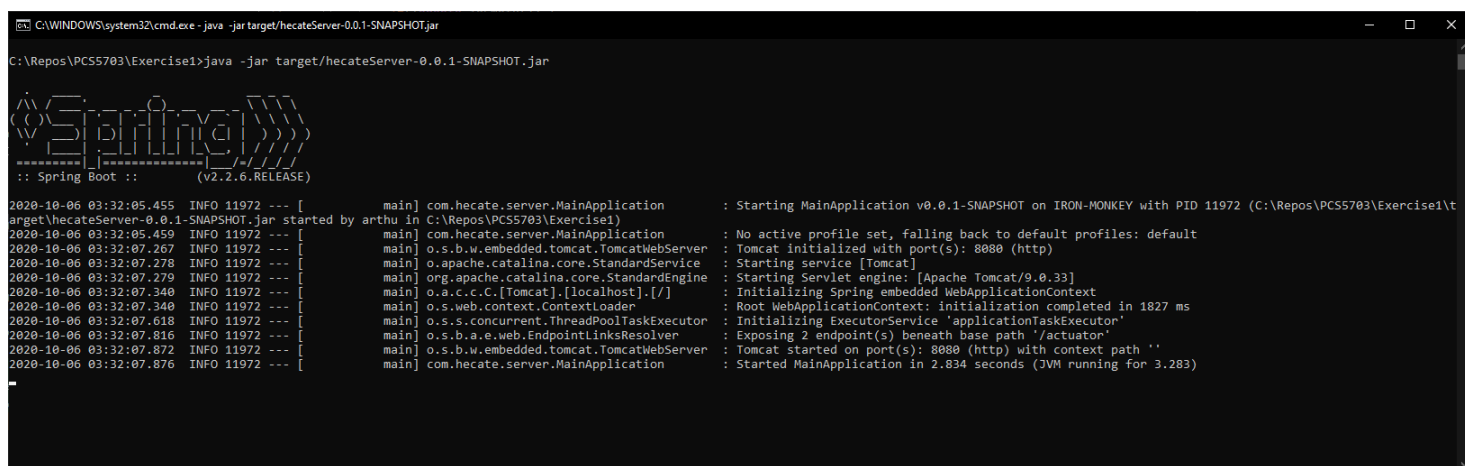
A criação de um novo agente no sistema deverá ser feita através de seu nome. O ID do agente criado será fornecido pelo sistema após sua criação.

A criação de uma sala funciona de forma similar à criação de um agente. Cada sala é inicialmente criada sem nenhum agente. Agentes devem ser atribuídos à sala após sua criação.

Cada mensagem é obrigatoriamente enviada por um agente, mas pode ser destinada a uma sala ou a um agente específico. Caso seja destinada a uma sala, todos os ocupantes da mesma naquele momento receberão a mesma mensagem.

Todo o acesso ao Hecate é feito através de chamadas REST. O resultado de todas as chamadas é processado no console do servidor, de forma que o aluno pode acompanhar o que está acontecendo após cada chamada de forma visual.

Abaixo encontra-se uma figura do console no momento em que o ambiente é iniciado:



```
C:\WINDOWS\system32\cmd.exe - java -jar target/hecateServer-0.0.1-SNAPSHOT.jar
C:\Repos\PCS5703\Exercise1>java -jar target/hecateServer-0.0.1-SNAPSHOT.jar

:: Spring Boot :: (v2.2.6.RELEASE)

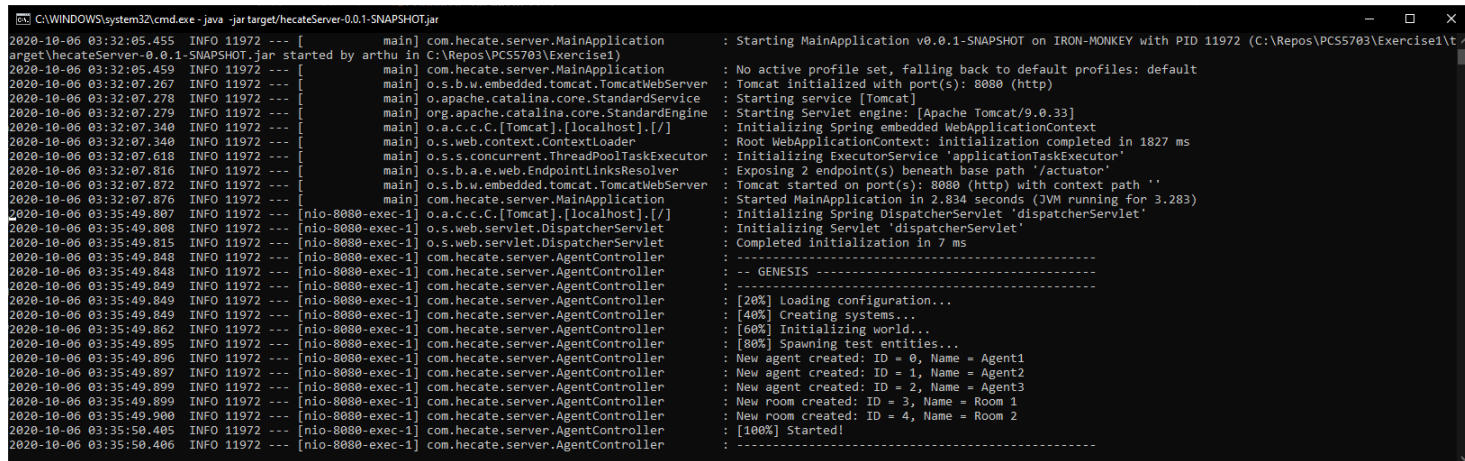
2020-10-06 03:32:05.455 INFO 11972 --- [main] com.hecate.server.MainApplication : Starting MainApplication v0.0.1-SNAPSHOT on IRON-MONKEY with PID 11972 (C:\Repos\PCS5703\Exercise1\t
arget\hecateServer-0.0.1-SNAPSHOT.jar started by arthu in C:\Repos\PCS5703\Exercise1)
2020-10-06 03:32:05.459 INFO 11972 --- [main] com.hecate.server.MainApplication : No active profile set, falling back to default profiles: default
2020-10-06 03:32:07.267 INFO 11972 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-10-06 03:32:07.278 INFO 11972 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-10-06 03:32:07.279 INFO 11972 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2020-10-06 03:32:07.340 INFO 11972 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-10-06 03:32:07.340 INFO 11972 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1827 ms
2020-10-06 03:32:07.618 INFO 11972 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-10-06 03:32:07.816 INFO 11972 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2020-10-06 03:32:07.872 INFO 11972 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-10-06 03:32:07.876 INFO 11972 --- [main] com.hecate.server.MainApplication : Started MainApplication in 2.834 seconds (JVM running for 3.283)
```

Figure 1: Hecate: Inicialização

Ao ser iniciado, o aluno deverá inicialmente criar o mundo no sistema através da chamada apropriada (abaixo). Durante esse processo, o sistema cria três salas já populadas por três diferentes agentes. Estas salas (e seus agentes) deverão ser desconsideradas pelo aluno durante a realização do

exercício prático (ou seja, nem as salas inicialmente criadas, nem os agentes nelas contidos fazem parte dos requisitos de implementação do sistema).

Abaixo encontra-se uma figura do console no momento em que o mundo é criado:



```

C:\WINDOWS\system32\cmd.exe - java -jar target\hecateServer-0.0.1-SNAPSHOT.jar
2020-10-06 03:32:05.455 INFO 11972 --- [main] com.hecate.server.MainApplication : Starting MainApplication v0.0.1-SNAPSHOT on IRON-MONKEY with PID 11972 (C:\Repos\PCS5703\Exercise1\t
target\hecateServer-0.0.1-SNAPSHOT.jar started by arthu in C:\Repos\PCS5703\Exercise1)
2020-10-06 03:32:05.459 INFO 11972 --- [main] com.hecate.server.MainApplication : No active profile set, falling back to default profiles: default
2020-10-06 03:32:07.267 INFO 11972 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-10-06 03:32:07.278 INFO 11972 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-10-06 03:32:07.279 INFO 11972 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2020-10-06 03:32:07.340 INFO 11972 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-10-06 03:32:07.340 INFO 11972 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1827 ms
2020-10-06 03:32:07.618 INFO 11972 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorsService 'applicationTaskExecutor'
2020-10-06 03:32:07.816 INFO 11972 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2020-10-06 03:32:07.872 INFO 11972 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-10-06 03:32:07.876 INFO 11972 --- [main] com.hecate.server.MainApplication : Started MainApplication in 2.834 seconds (JVM running for 3.283)
2020-10-06 03:35:49.807 INFO 11972 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-10-06 03:35:49.808 INFO 11972 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2020-10-06 03:35:49.815 INFO 11972 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 7 ms
2020-10-06 03:35:49.848 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : -----
2020-10-06 03:35:49.848 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : -- GENESIS -----
2020-10-06 03:35:49.849 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : -----
2020-10-06 03:35:49.849 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : [20%] Loading configuration...
2020-10-06 03:35:49.849 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : [40%] Creating systems...
2020-10-06 03:35:49.862 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : [60%] Initializing world...
2020-10-06 03:35:49.895 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : [80%] Spawning test entities...
2020-10-06 03:35:49.896 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : New agent created: ID = 0, Name = Agent1
2020-10-06 03:35:49.897 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : New agent created: ID = 1, Name = Agent2
2020-10-06 03:35:49.899 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : New agent created: ID = 2, Name = Agent3
2020-10-06 03:35:49.899 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : New room created: ID = 3, Name = Room 1
2020-10-06 03:35:49.900 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : New room created: ID = 4, Name = Room 2
2020-10-06 03:35:50.405 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : [100%] Started!
2020-10-06 03:35:50.406 INFO 11972 --- [nio-8080-exec-1] com.hecate.server.AgentController : -----
  
```

Figure 2: Hecate: Criação do mundo

Uma vez criado o mundo, o aluno deverá utilizar as chamadas REST associadas a cada funcionalidade do sistema. Cada chamada é uma URL composta pelo endereço do servidor (que nesse caso roda localmente na porta 8080) e o caminho para a funcionalidade específica. Exemplificando: a criação de mundo é disponibilizada no caminho `"/createWorld"`. Assim, considerando-se o endereço local do servidor, o endereço completo para esta chamada é: `"http://localhost:8080/createWorld"`. **IMPORTANTE: os caminhos são case-sensitive.**

Abaixo encontram-se descritas as chamadas disponíveis nesta versão do Hecate. Algumas dessas chamadas requerem parâmetros específicos; todos os parâmetros listados abaixo podem ser enviados na forma de texto:

- `/createWorld` : Cria o mundo dentro do Hecate. Esta chamada deve ser sempre executada após a inicialização do sistema.
- `/createWorld/test` : Executa os testes de envio de mensagem pré-programados no sistema. O aluno pode executar estes testes caso deseje observar o fluxo interno de envio e recebimento de mensagens.
- `/createAgent` : Cria um novo agente no mundo. Retorno: mensagem contendo o ID do agente criado. Parâmetros necessários: **name** contendo o nome do agente.
- `/createRoom` : Cria uma nova sala no mundo. Retorno: mensagem contendo o ID da sala criada. Parâmetros necessários: **name** contendo o nome da sala.
- `/addAgentsToRoom` : Adiciona uma lista de agentes a uma determinada sala. Parâmetros necessários: **room** contendo o ID da sala; **agents** contendo uma lista de IDs de agentes a serem adicionados na sala.

- **/removeAgentsFromRoom** : Remove uma lista de agentes de uma determinada sala. Parâmetros necessários: **room** contendo o ID da sala; **agents** contendo uma lista de IDs de agentes a serem removidos da sala.
- **/sendMessageToRoom** : Envia uma mensagem de um agente para uma sala. Parâmetros necessários: **from** contendo o ID do agente remetente; **to** contendo o ID da sala destinatária; **message** contendo o texto da mensagem enviada.
- **/sendMessageToAgent** : Envia uma mensagem de um agente para um outro agente. Parâmetros necessários: **from** contendo o ID do agente remetente; **to** contendo o ID do agente destinatário; **message** contendo o texto da mensagem enviada.
- **/agent/messages/id** : Recupera todas as mensagens recebidas por um determinado agente. Por exemplo: para recuperar a lista de mensagens recebidas pelo agente de ID igual a 2 até o presente momento, o caminho da chamada correspondente é `"/agent/messages/2"`.

É importante observar que uma lista de mensagens recuperada para um determinado agente é um documento JSON contendo todos os campos descritos anteriormente. Estes campos são reflexos dos objetos associados no momento em que a lista de mensagens é recuperada. Desta forma, pode-se esperar, por exemplo, que o campo `"room"` venha populado por uma representação do objeto `"Sala"` naquele momento, incluindo a lista de agentes ocupando a sala e as mensagens recebidas anteriormente por cada agente. Abaixo encontram-se exemplos de objetos JSON recuperados contendo uma única mensagem enviada para (a) uma sala e (b) para um único agente:



Figure 3: (a): Mensagem enviada para uma sala

JSON		Raw Data	Headers
Save	Copy	Collapse All	Expand All
		Filter JSON	
▼ 0:			
▼ from:			
id:		"0"	
name:		"Agent1"	
▼ messages:			
▼ 0:			
▼ from:			
id:		"0"	
name:		"Agent1"	
messages:		[]	
▼ room:			
id:		"3"	
name:		"Room 1"	
▼ agents:			
▼ 0:			
id:		"0"	
name:		"Agent1"	
messages:		[]	
▼ 1:			
id:		"1"	
name:		"Agent2"	
messages:		[]	
message:		"Test #1"	
timestamp:		"2020-10-06T03:37:49.205"	
▼ to:			
id:		"2"	
name:		"Agent3"	
messages:		[]	
message:		"Test #2"	
timestamp:		"2020-10-06T03:37:50.273"	

Figure 4: (b): Mensagem enviada para um agente