

UNIVERSIDADE DE SÃO PAULO
Escola Politécnica da USP

PCS5703 – Exercício Prático 2
Planejamento

Profa. Dra. Anarosa Alves Franco Brandão

26/10/2020

O objetivo deste exercício é familiarizar o aluno com os conceitos de planejamento em Sistemas Multiagentes. O exercício deverá ser feito de forma **individual**, com entrega até o dia 09/11/2020 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** e o Exercício Prático 1. 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:

Durante a realização do Exercício Prático 1, os alunos de pós-graduação da Poli implementaram um sistema multiagentes simples utilizando o Hecate. Como parte do exercício, os alunos implementaram um sistema de controle simples em **ambiente local**, hospedado na máquina de cada aluno.

Como forma de aplicar os conhecimentos sobre planejamento e protocolos de comunicação adquiridos em aula, a professora pediu que cada aluno implementasse um mecanismo de planejamento local simples. Com esse mecanismo, os alunos poderiam então observar como o mecanismo de controle implementado anteriormente poderia ser traduzido em formato de planos posteriormente executados por cada agente.

Após as considerações acima, o novo exercício prático foi estruturado com as seguintes tarefas e restrições:

- Revisar o mecanismo de controle originalmente implementado de forma a garantir que todos os requisitos originais estejam implementados (inclusive aleatoriedade na escolha de salas, conforme discutido em sala de aula).
- Modificar o mecanismo de controle originalmente implementado de forma que os agentes **NÃO SEJAM** acionados individualmente. Ao invés disso, o mecanismo de controle deverá produzir um **plano de execução** para cada agente.
- Cada plano deverá ser gerado de acordo com uma sintaxe previamente definida (abaixo), no formato JSON, de forma que possa ser compreendido pelo Hecate.
- Cada plano individual deve ser executado sequencialmente no Hecate, de forma que ao final da execução de todos os planos os agentes satisfaçam os requisitos do cenário proposto (abaixo).

Consequentemente, os requisitos originais do ambiente local foram modificados :

3.1 Ambiente Local

- O ambiente local deve possuir três mecanismos de controle distintos: um mecanismo para a alocação inicial dos agentes, um mecanismo referente ao planejamento de distribuição dos agentes entre as salas, e um mecanismo referente ao monitoramento das mensagens trocadas entre os agentes.
- O mecanismo para alocação inicial de agentes será responsável por criar as salas e agentes utilizados pelo sistema.
- O mecanismo para alocação inicial de agentes deverá, também, atribuir **três agentes** a cada sala criada.
- O mecanismo de planejamento deve ser capaz de criar um plano para mover os agentes entre as salas.

+ Durante a criação de cada plano, deve-se levar em consideração que **cada agente só pode estar em uma sala de cada vez**.

- O mecanismo de planejamento deve ser capaz de registrar as condições antes e após a execução de todos os planos: cada sala e seus ocupantes, todos identificados. Tais condições serão denominadas, respectivamente, "contexto inicial" e "contexto final".
- O mecanismo de planejamento deve considerar as limitações de ocupação de sala, ou seja, garantir que a cada momento (após a execução de cada plano individual) 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 criar um plano para a movimentação dos 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
 - + Em **nenhum momento** uma sala pode possuir mais que **quatro** agentes alocados
 - + 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 salva em um arquivo de texto chamado "mensagens.txt"
- Tanto o contexto inicial quanto o contexto final deverão ser registrados em arquivos de texto (respectivamente: "contexto-inicial.txt" e "contexto-final.txt").
- Os planos criados deverão ser executados individualmente no Hecate através da chamada apropriada. Além disso, todos os planos produzidos deverão ser salvos em arquivos de texto (no formato "plano-< IDdoagente >.txt").

4 Entregáveis:

- Código-fonte completo do ambiente local, incluindo instruções de execução
- Arquivos texto produzidos (conforme descrição acima).

5 Documentação técnica:

Para este exercício, o arcabouço Hecate utiliza quatro estruturas principais: **agentes**, **salas**, **mensagens** e **ações**. 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: **ID do agente** que enviou a mensagem
- To: **ID da sala** a quem foi destinada a mensagem
- Message: O texto contido na mensagem
- LocalDateTime: Horário (timestamp) em que a mensagem foi enviada

Ação:

- Tipo: Os tipos de ação executados pelo agente podem ser "move", "leave" (referente à movimentação entre salas) e "message" (referente ao envio de mensagens)
- Parâmetro: No caso do tipo "move", o parâmetro necessário é o ID da sala para qual o agente deve se mover. No caso do tipo "leave", o parâmetro necessário é o ID da sala da qual o agente deve se mover. No caso do tipo "message", o parâmetro é um objeto de mensagem (descrito acima).

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.

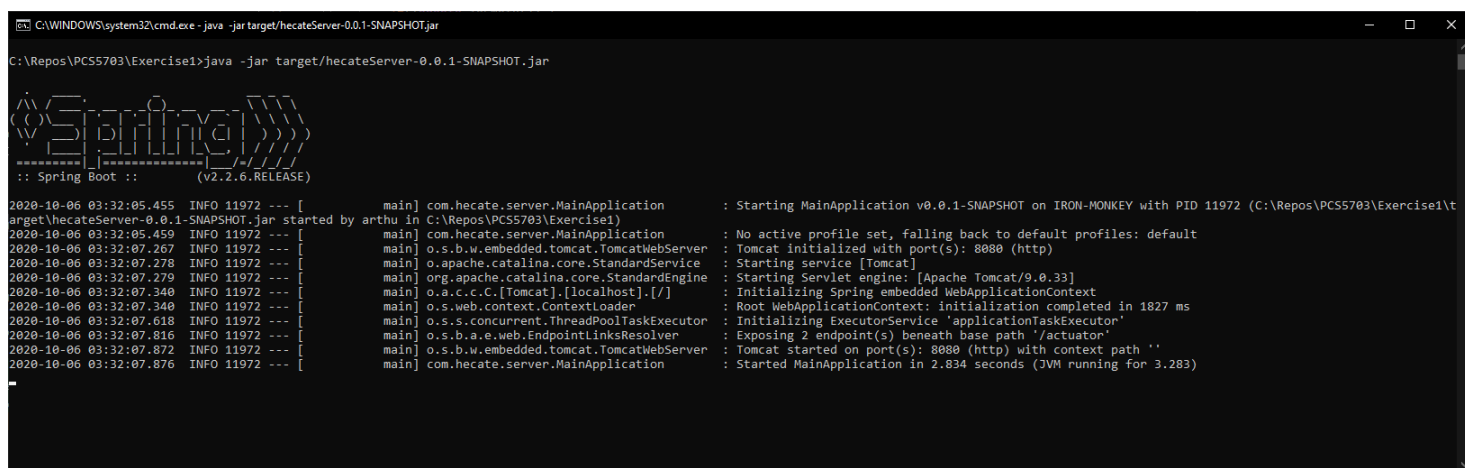
A criação de uma sala funciona de forma similar à criação de um agente. Cada sala é inicialmente criada sem nenhum agente. Agentes devem ser atribuídos à sala após sua criação.

Cada mensagem é obrigatoriamente enviada por um agente, mas pode ser destinada a uma sala **ou** a um agente específico (nunca aos dois ao mesmo tempo). Caso seja destinada a uma sala, todos os ocupantes da mesma naquele momento receberão a mesma mensagem.

Após a execução de cada plano, espera-se que os agentes se comportem de acordo com os requisitos do sistema, respeitando os limites de ocupação de cada sala e enviando as mensagens apropriadas.

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
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:

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. **Todas as chamadas retornam um objeto JSON (salvo observação específica).** Algumas dessas

```

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 arthur in C:\Repos\PCS5703\Exercise1\t
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)
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 : -- GENESIS -----
2020-10-06 03:35:49.848 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

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. Retorna mensagem simples de confirmação (não é JSON). Nesta chamada também são criadas entidades para teste, pré-populadas (ver log do console).
- **/createAgent** : Cria um novo agente no mundo. Retorno: representação em JSON do agente criado. Parâmetros necessários: **name** contendo o nome do agente.
- **/createRoom** : Cria uma nova sala no mundo. Retorno: representação em JSON da sala criada. Parâmetros necessários: **name** contendo o nome da sala.
- **/agent/id** : Recupera uma representação do agente em JSON.
- **/room/id** : Recupera uma representação da sala em JSON.
- **/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"`.
- **/agent/plan/id** : Recupera o plano (lista de ações) de um determinado agente representado em JSON. Por exemplo: para recuperar o plano do agente de ID igual a 2, o caminho da chamada correspondente é `"/agent/plan/2"`.
- **/addAgentsToRoom** : Adiciona uma lista de agentes a uma determinada sala. Retorno: representação em JSON da sala modificada. 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. Retorno: representação em JSON da sala modificada. Parâmetros necessários: **room** contendo o ID da sala; **agents** contendo uma lista de IDs de agentes a serem removidos da sala.

- **/agent/setActions**: Atribui uma lista (array) de ações ao agente especificado. Retorna uma representação do agente modificado em JSON. Parâmetros necessários: **id** contendo o ID do agente remetente; **actions** contendo a lista (array) de ações. Observem que cada ação é um objeto JSON; assim, o parâmetro **actions** é um array nomeado de ações: **"actions": [...]**.
- **/executePlan/id** : Executa o conjunto de ações de um determinado agente. Retorna mensagem simples de confirmação (não é JSON). Parâmetros necessários: **id** contendo o ID do agente referenciado.