Agentes Inteligentes

Fase 2

Ana Gil, Beatriz Rocha, Hugo Matias e João Abreu

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal
e-mail: {a85266,a84003,a85370,a84802}@alunos.uminho.pt

Resumo Neste documento, será apresentada a nossa proposta de solução para o Problema do Reequilíbrio de Partilha de Bicicletas. Para isso, iremos apresentar os vários agentes que iremos implementar (Agentes Utilizadores, Agentes de Estação, Agente *Interface* e Agente Central), a Arquitetura do sistema, o Fluxo de Resolução, o Protocolo de Comunicação entre os vários agentes e o Processamento Interno dos Agentes.

Keywords: Agentes inteligentes · Sistemas inteligentes · Sistemas multi-agentes · Inteligência Artificial Distribuída.

1 Introdução

Um sistema multi-agentes é composto por um conjunto de agentes inteligentes que cooperam mutuamente de maneira a atingir um objetivo comum (a solução do problema) que ultrapassa as suas competências individuais. A importância dos sistemas multi-agentes assenta no tipo de cada agente inteligente e na sua capacidade de coordenação. A interação entre estes agentes pode ser do tipo cooperativo (se os agentes incorporam o seu conhecimento e outros pertences úteis para alcançar um objetivo comum que individualmente não conseguiriam) ou do tipo competitivo (se os agentes agem de forma egoísta de maneira a atingir os seus interesses próprios) e implica a exploração, caracterização, explicação e implementação de um conjunto de funcionalidades.

O objetivo do sistema multi-agentes do presente trabalho prático é resolver o Problema do Reequilíbrio de Partilha de Bicicletas através da aplicação de vários sensores virtuais de captura de localização *GPS*, representados por agentes. Neste trabalho prático, existirão os seguintes agentes:

- **Agentes Utilizadores**: representam agentes associados a *smartphones* pessoais dos utilizadores;

– **Agentes de Estação**: representam as estações do Sistema de Partilha de Bicicletas e escutam as mensagens dos Agentes Utilizadores;

– **Agente *Interface***: é a partir deste agente que o utilizador vai interagir com os Agentes de Estação como forma de monitorizar a sua gestão de bicicletas;

– **Agente Central**: este agente tem conhecimento do mapa geral do sistema e é responsável por gerir as áreas de proximidade das várias estações.

Desta forma, iremos começar por explicar o nosso pensamento e a modelação de uma arquitetura distribuída baseada em agentes para o dado problema. Para isso, começaremos por expor a Arquitetura da nossa proposta de solução, seguida do seu Fluxo de Resolução. Apresentaremos também o Protocolo de Comunicação e o Processamento Interno dos Agentes e, para isso, recorreremos ao *Agent UML* para formalizar os protocolos de interação entre os mesmos. De seguida, passaremos, efetivamente, para a apresentação e explicação da implementação do sistema, demonstrando, por fim, os resultados obtidos.

2 Estado de Arte

A procura por uma mobilidade mais sustentável é cada vez mais frequente, principalmente pelos grandes centros urbanos. São várias as cidades que, cada vez mais, adotam medidas para incentivar a utilização de modos suaves nas pequenas deslocações diárias, tentando mostrar que são cidades interessantes e atrativas. Uma das medidas é implementar sistemas de bicicletas partilhadas na cidade, onde na última década este tipo de sistemas tem adquirido uma maior visibilidade devido ao avanço tecnológico que permitiu melhorar estes sistemas e torná-los mais atrativos.

Assim sendo, os Sistemas de Bicicletas Partilhadas consistem na disponibilização de aluguer a curto prazo de bicicletas distribuídas por uma rede de estações, tipicamente numa área urbana, permitindo viagens a baixo custo (ou sem custo) entre dois pontos. Devem atender às necessidades específicas de cada contexto (por exemplo, bicicletas elétricas em áreas com declives muito acentuados ou bicicletas adaptadas para pessoas com mobilidade reduzida).

De uma forma geral, é possível afirmar que Portugal está a dar os primeiros passos na implementação de verdadeiras estruturas de *SBP* quando comparados com os sistemas a nível internacional, existindo, segundo dados obtidos até janeiro de 2018, 23 sistemas de bicicletas partilhadas implementados em Portugal continental.

3 Arquitetura

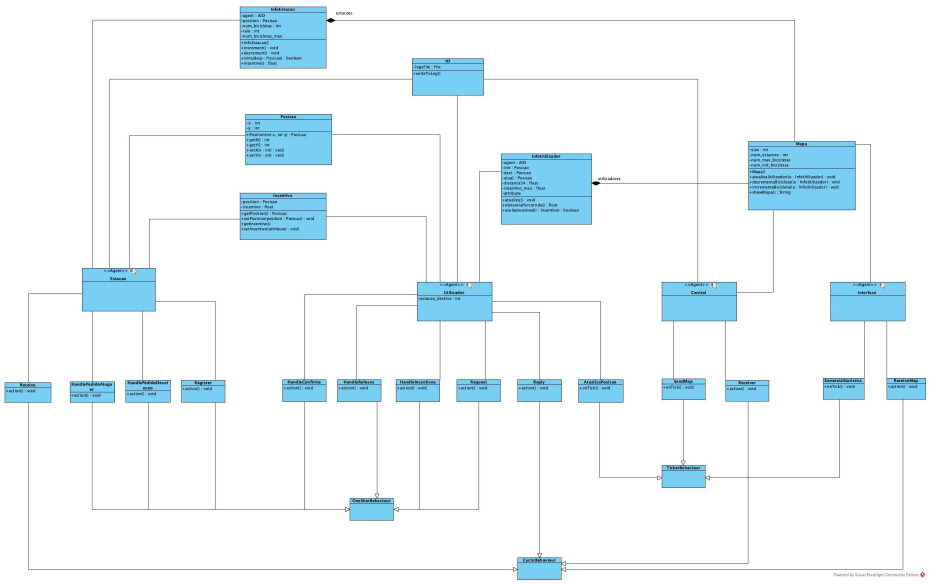

Figura 1. Diagrama de classes do sistema

A nível estrutural, o nosso sistema permaneceu com os mesmos agentes definidos na fase 1 do trabalho. No entanto, à medida que fomos implementando o projeto, surgiu a necessidade de criar mais *behaviours* que o previsto, bem como classes auxiliares.

Posto isto, a estrutura do sistema é representada pelo diagrama de classes acima apresentado. Esta estrutura é constituída por quatro agentes que correspondem a quatro classes que são especificações da classe *Agent*, disponibilizada pela *framework JADE*. Estas classes são as seguintes:

- **Agente Central**: representa o agente coordenador tanto dos Agentes de Estação, como dos Agentes Utilizadores. Como controlador, este irá apenas conter uma variável de instância do tipo Mapa, onde se armazenam todas as informações do sistema;

- **Agente de Estação**: representa uma estação que ocupa um determinado lugar do mapa do sistema. Este irá ter como variável de instância uma lista de posições de todas os Agentes de Estação até ao momento. Terá também uma variável do tipo *InfoEstacao*, onde contém todas as informações necessárias

acerca deste agente, assim como os métodos úteis para o seu desenvolvimento;

- **Agente Utilizador**: representa um agente associado ao dispositivo pessoal de um utilizador. Tal como o Agente de Estação, este irá conter uma variável do tipo *InfoUtilizador*, com todos os dados referentes ao utilizador. Este também possui uma lista com as posições das estações existentes para que possa ter uma noção inicial do mapa e saber de onde para onde se vai deslocar.
  Os três agentes anteriores vão também possuir uma variável do tipo *IO*, que é uma classe auxiliar útil para desenvolver o ficheiro *log* do sistema;

- **Agente *Interface***: representa o agente pelo qual o utilizador vai comunicar com o Agente Central, de modo a monitorizar a gestão das bicicletas das estações. Para isso, este irá apenas ter uma variável do tipo Mapa, onde se encontram todos os dados do sistema atualizados.

4 Fluxo de Resolução de Aluguer

De forma a ser mais facilmente compreendido o modo como se processa a resolução de alugueres e a influência dos agentes no sistema, é apresentado, na Figura 2, o diagrama de atividade que representa a interação entre os agentes, tendo em vista a resolução completa de um aluguer de uma bicicleta.

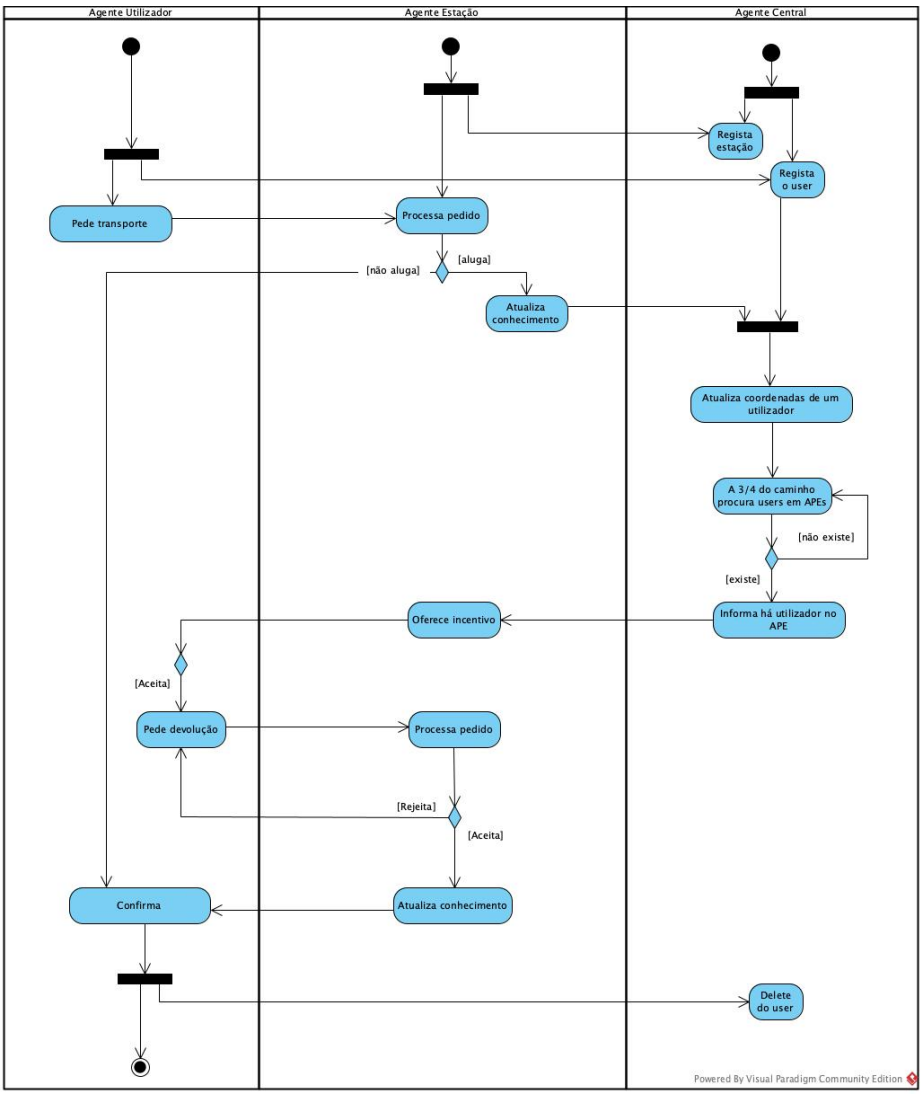

Figura 2. Fluxo de Resolução de Aluguer

Face à fase anterior do trabalho, apenas acrescentámos no Agente Central a ação de registo de um Agente de Estação. Quanto ao restante diagrama, este vai de encontro à implementação desenvolvida nesta segunda fase.

Através da observação do diagrama, podemos verificar que o processo tem início na criação de um Agente Utilizador que é imediatamente registado no Agente Central, fazendo também um pedido de aluguer ao Agente de Estação. Este processa o pedido que resultará num dos seguintes cenários:

- A estação está vazia, não sendo possível alugar nenhuma bicicleta;
- A estação tem pelo menos 1 bicicleta disponível para alugar.

No primeiro caso, o Agente Utilizador é descartado, sendo removido do conhecimento do Agente Central e, no segundo caso, o processo de aluguer continua normalmente. Neste caso, o Agente de Estação atualiza os seus dados (número de bicicletas) e o Agente Utilizador começa a sua viagem. De seguida, quando o Agente Utilizador se encontrar a mais de 3/4 da distância do trajeto completo, o Agente Central atualiza as suas coordenadas no seu conhecimento. Depois, procura os Agentes Utilizadores dentro das APEs e, no caso de existirem, informa o Agente de Estação correspondente que há, pelo menos, um Agente Utilizador na sua APE. O Agente de Estação oferecerá um incentivo de devolução da bicicleta ao respetivo Agente Utilizador, sendo este capaz de aceitar ou rejeitar (pode vir a ter melhores ofertas de incentivos). Se aceitar, será feito um pedido de devolução da bicicleta do Agente Utilizador para o Agente de Estação, que também pode ser aceite ou rejeitado pelo último. Esperemos que nunca seja rejeitado este pedido, porque, na verdade, o propósito de todo o trabalho é este mesmo, não deixar haver sobrecarga em estações, não permitindo sequer a devolução de bicicletas, porque atingiram a capacidade máxima. No entanto, decidimos que, se infelizmente isso vier a acontecer, o Agente Utilizador terá de esperar para ter lugar para deixar a sua bicicleta na estação, caso contrário o Agente de Estação só terá de atualizar novamente os seus dados. No fim, depois de devolvida a bicicleta, o Agente Utilizador morre e é apagado do conhecimento do Agente Central.

5 Protocolo de Comunicação

Para a implementação deste sistema é necessário que este seja dinâmico, ou seja, os agentes apresentados na secção 1 têm necessariamente de comunicar entre si.

De seguida, é apresentado o panorama geral de comunicação do sistema, apresentando as mensagens trocadas entre os diferentes agentes baseadas em *FIPA Performatives*.

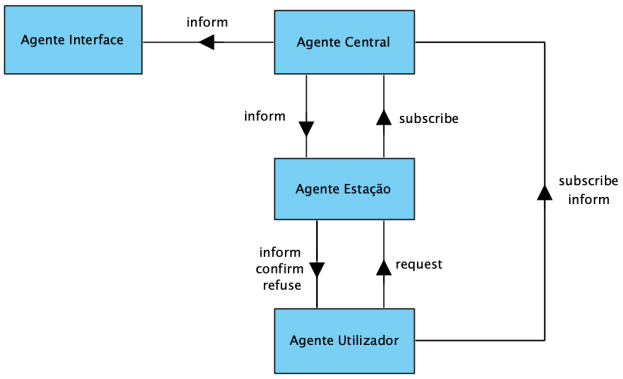

Figura 3. Protocolo de Comunicação geral

De forma a pormenorizar cada uma destas interações, recorremos a diagramas de sequência, destacando as diferentes fases do processo de aluguer de bicicletas.

Estes, de forma geral, estão muito semelhantes aos apresentados na fase anterior, na medida em que vão de encontro ao que foi implementado nesta fase.

5.1 Novo Agente Utilizador

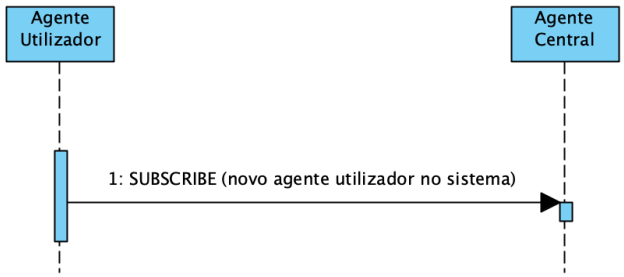

Figura 4. Diagrama de sequência correspondente à criação de um Agente Utilizador

Cada vez que um novo Agente Utilizador é criado, este irá avisar o Agente Central, de forma a que este possa atualizar o seu conjunto de Agentes Utilizadores. Para isso, recorremos à *perfomative SUBSCRIBE*.

5.2 Aluguer de uma bicicleta

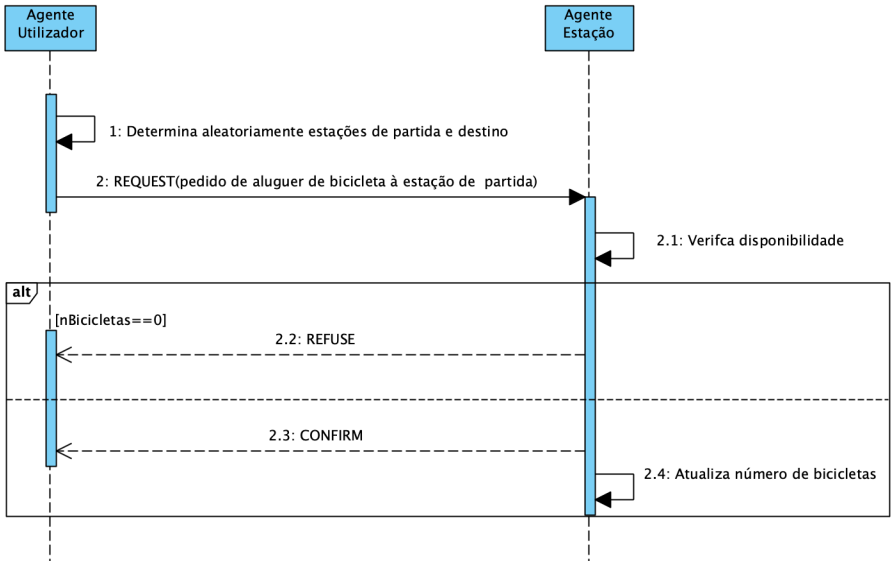

Figura 5. Diagrama de sequência correspondente ao aluguer de uma bicicleta

Após a criação de um determinado Agente Utilizador, este irá determinar aleatoriamente as posições de partida e chegada, estando ambas associadas às posições fixas das diferentes estações do sistema. Posteriormente, irá enviar um *REQUEST* ao Agente de Estação de partida a solicitar uma das suas bicicletas. Cabe ao Agente de Estação analisar o seu estado, ou seja, verificar o número de bicicletas disponíveis. Caso hajam bicicletas, este irá responder com um *CONFIRM* e atualizar o seu número de bicicletas. Caso a estação não tenha nenhuma bicicleta disponível, então irá recusar o pedido e o Agente Utilizador termina a sua execução, visto que, num contexto adaptado a casos reais, os utilizadores não iriam esperar até que houvesse uma bicicleta disponível.

5.3 Devolução de bicicleta

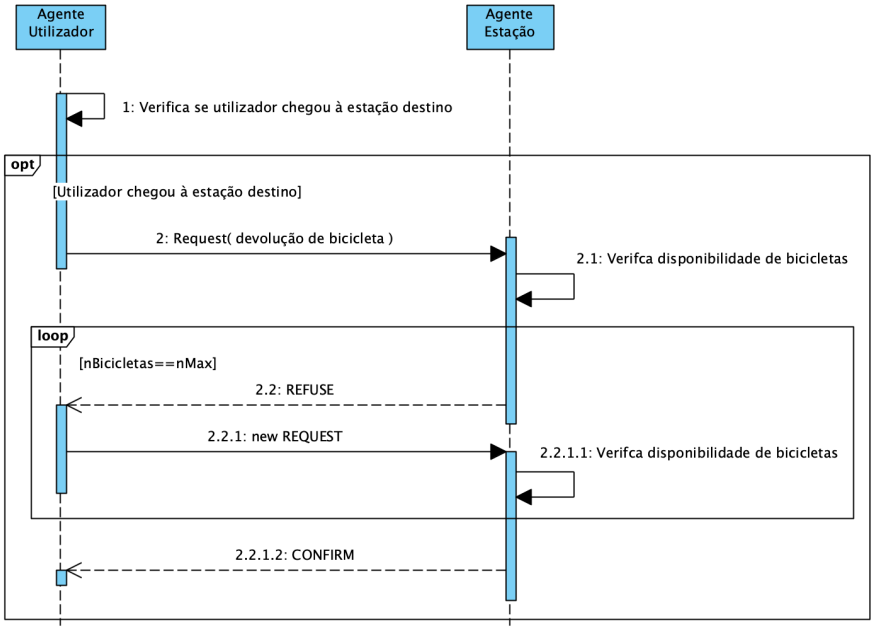

Figura 6. Diagrama de sequência correspondente à devolução de uma bicicleta

Neste diagrama, está representado o processo de devolução de uma bicicleta na estação destino. Para que esta ação aconteça, a posição do Agente Utilizador tem que coincidir com a posição do respetivo Agente de Estação. Posto isto, o Agente Utilizador envia um *REQUEST* associado à entrega da bicicleta. Cabe ao Agente de Estação analisar o seu estado e ver se existem lugares vazios para aceitar a bicicleta. Caso existam, então responde com um *CONFIRM* e o Agente Utilizador termina a sua execução. Se a estação estiver cheia, então este terá que esperar.

5.4 Gestão de APEs

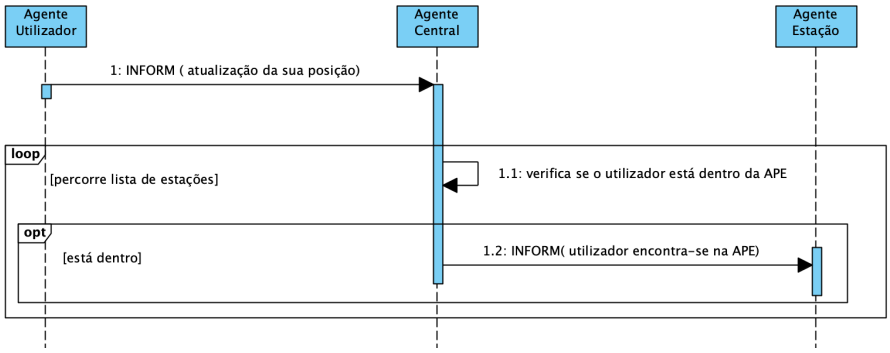

Figura 7. Diagrama de sequência correspondente à gestão de APEs

A gestão da Área de Proximidade de uma Estação (APE) é efetuada pelo Agente Central. Cada vez que um Agente Utilizador atualiza a sua posição, o Agente Central é informado e irá verificar se essa posição pertence à APE de alguma estação. Em caso afirmativo, irá enviar uma mensagem do tipo *INFORM* ao respetivo Agente de Estação a informar que um novo Agente Utilizador entrou na sua APE.

5.5 Envio de incentivos

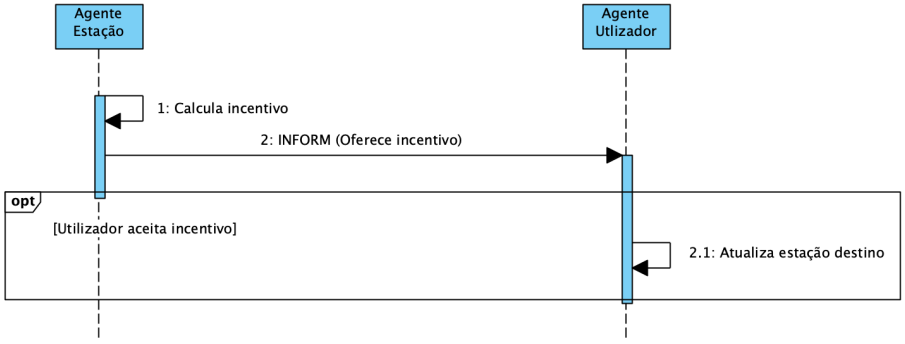

Figura 8. Diagrama de sequência correspondente ao envio de incentivos

Seguindo a lógica do protocolo de comunicação anterior (Gestão de APEs), o Agente de Estação irá enviar um incentivo ao Agente Utilizador que se encontra dentro da sua APE, calculado tendo em conta a sua lotação e enviando-o através de uma mensagem do tipo *INFORM*. Já o Agente Utilizador, através de um conjunto de regras, decide se quer aceitar ou não. Caso aceite, então deve atualizar a sua estação destino.

5.6 Monitorização da gestão de bicicletas

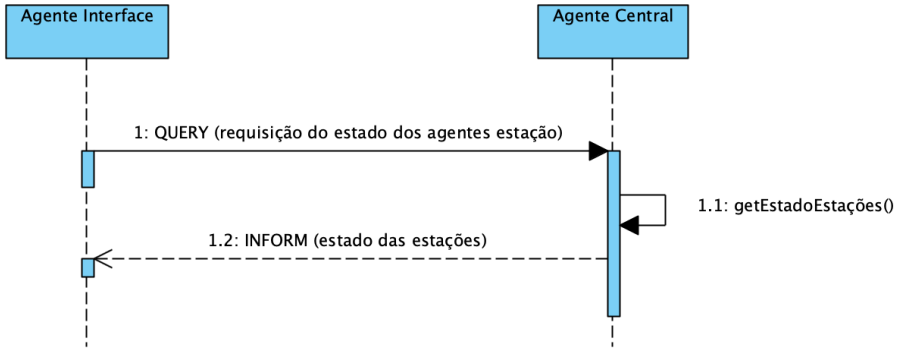

Figura 9. Diagrama de sequência correspondente à monitorização da gestão de bicicletas

Este diagrama corresponde ao protocolo de comunicação entre o utilizador e a *interface* do programa de forma a monitorizar a gestão das bicicletas de cada Agente de Estação. Como o Agente Central possui as informações gerais de todo o sistema, esta comunicação efetua-se entre ele e o Agente *Interface*. Para isso, o Agente *Interface* solicita o estado atual das estações do sistema e o Agente Central, após obter os respetivos dados, devolve-os através de uma mensagem do tipo *INFORM*.

6 Processamento Interno dos Agentes

Nesta secção, iremos apresentar os diagramas que descrevem o processamento interno do Agente Utilizador, do Agente Central e do Agente de Estação.

6.1 Agente Utilizador

Como podemos ver na Figura 10, o Agente Utilizador possui três estados, **Em repouso**, **Em andamento** e **À espera**. Quando este agente nasce, encontra-se **Em repouso** numa estação até decidir alugar uma bicicleta. Se não existir nenhuma bicicleta disponível, este morre, ou seja, é a nossa forma de simular que o Agente Utilizador não quer esperar e prefere optar por outra estação. Por outro lado, caso haja alguma bicicleta disponível, muda para o estado **Em andamento** e, consequentemente, inicia a sua viagem. À medida que se aproxima do destino, vai começar a receber incentivos. Poderá optar por aceitar o incentivo em questão e terminar a sua viagem, poderá rejeitar esse incentivo e aceitar um outro que lhe agrade mais ou poderá não aceitar nenhum dos incentivos. Caso escolha a última opção, significa que o Agente Utilizador optou por entregar a bicicleta no destino desejado e, portanto, será necessário ter em conta se a estação destino está cheia. Caso não esteja, o Agente Utilizador poderá terminar a sua viagem, caso contrário ficará **À espera** até que seja possível entregar a bicicleta e, posteriormente, terminará a sua viagem.

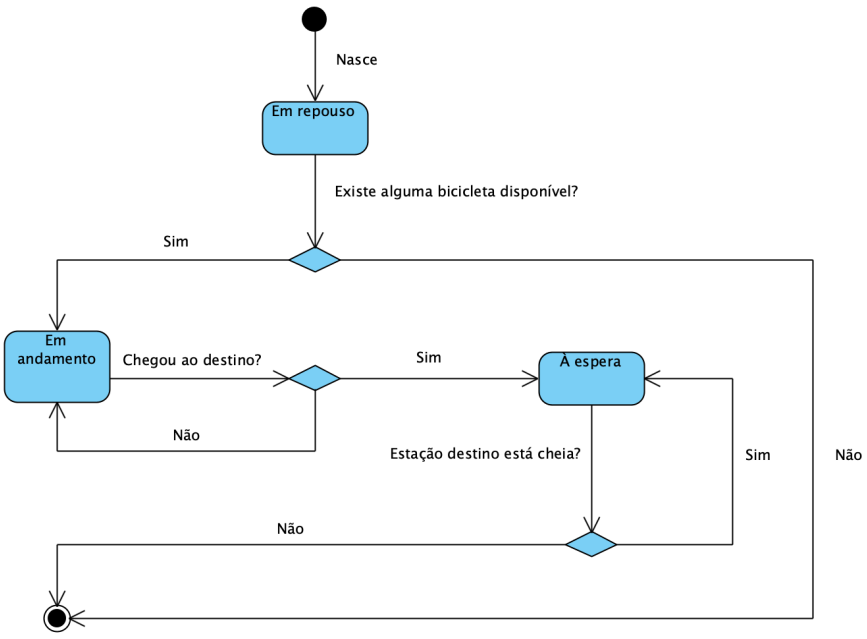

Figura 10. Máquina de estados do Agente Utilizador

6.2 Agente Central

Como podemos ver na Figura 11, o Agente Central possui três estados, **Em espera**, **A calcular proximidades** e **A atualizar conjunto de AUs**. Quando o programa se inicia, o Agente Central encontra-se **Em espera** até haver alterações das posições dos Agentes Utilizadores ou até haver uma inserção/remoção (nascimento/morte) de um Agente Utilizador. Caso não haja alterações nas posições dos Agentes Utilizadores, o Agente Central continua **Em espera**, caso contrário começa **A calcular proximidades** entre os Agentes Utilizadores e as várias APEs. De seguida, informa os Agentes de Estação respetivos e volta a ficar **Em espera**. Por outro lado, caso não haja uma inserção/remoção de um Agente Utilizador, o Agente Central continua **Em espera**, caso contrário passa **A atualizar conjunto de AUs** e volta a ficar **Em espera**. De frisar que, quando o programa termina, o Agente Central também termina.

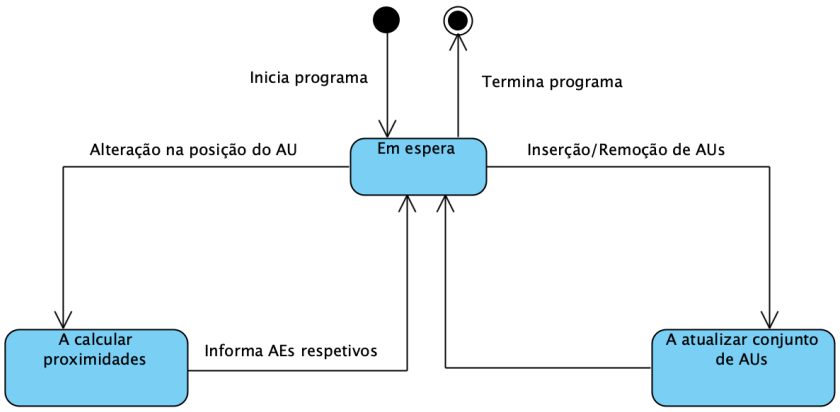

Figura 11. Máquina de estados do Agente Central

6.3 Agente de Estação

Como podemos ver na Figura 13, o Agente de Estação possui dois estados, **À espera de notificações/pedidos** e **A processar notificação ou pedido**. Quando o programa se inicia, o Agente de Estação encontra-se **À espera de notificações/pedidos**, ou seja, à espera de pedidos de devolução, pedidos de aluguer ou notificações de que um Agente Utilizador se aproximou da sua APE. Quando ocorre algum destes cenários, passa **A processar notificação ou pedido** (envia/não envia incentivo ou aceita/rejeita pedido de aluguer/devolução,

respetivamente) e volta a estar **À espera de notificações/pedidos**. De frisar que, quando o programa termina, o Agente de Estação também termina.

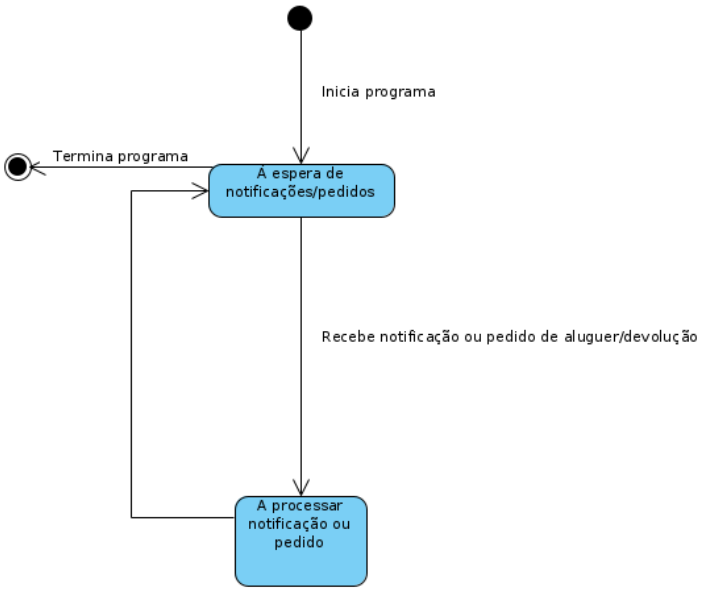

Figura 12. Máquina de estados do Agente de Estação

7 Implementação

Após definirmos a estratégia e as especificações a adotar, passamos à implementação do projeto. Nesta secção iremos explicar como esta está elaborada. Para melhor entender a estrutura do projeto, apresenta-se de seguida um esquema da estrutura da pasta referente ao código.

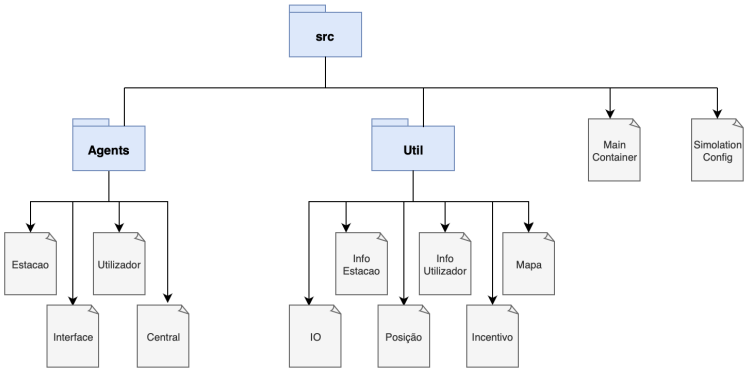

Figura 13. Estrutura de pastas do Projeto

Assim sendo, o package Agents engloba cada uma das classes Agent referente aos quatro agentes do sistema, sendo estes descritos posteriormente de forma mais detalhada. O package Util reúne um conjunto de classes auxiliares à execução do projeto, sendo que algumas podem ser conteúdos de mensagens (InfoUtilizador, InfoEstacao, Mapa e Incentivo) enquanto que outras são para operações de mais baixo nível (IO para escrever num ficheiro os logs do programa e Posicao para representar uma posição no mapa de floats e respetivos métodos auxiliares). Por último, temos o ficheiro MainContainer onde se encontra o método main que executa o programa, assim como temos o ficheiro SimulationConfig, onde se encontram todas as variáveis fixas que configuram o sistema, como por exemplo, o tamanho do mapa e o número de estações/utilizadores. Esta classe serve principalmente para facilitar a mudança de variáveis de maneira a testar o programa de várias maneiras diferentes.

7.1 Ambiente inicial

No desenvolvimento inicial do projeto, optamos por ter um tamanho fixo do mapa, um número fixo de estações e uma posição fixa para cada estação, para se tornar mais fácil o começo de todo o projeto. Mais tarde, pensamos em tornar o programa mais realista e ser possível que o mapa tenha tamanho variável, haja um número variável de estações no mapa e também uma posição variável para cada estação. No entanto, como tanto o número de estações e as suas posições no mapa podem variar, surgiram muitos problemas que tivemos de resolver.

- **Como escolher as posições sem estarem demasiado próximas?** - foi criado uma função de cálculo de proximidade de estações. Quando uma estação é criada tem de ter atenção às posições das outras para não ficar

demasiado próxima. O valor da proximidade foi um valor arbitrário escolhido pelo grupo.
- **Como saber o limite de estações permitidas sabendo o tamanho do mapa?** - Usando a função anterior, criamos um limite de tentativas que uma estação tem para determinar a sua posição. O valor limite escolhido foi 10 tentativas e caso não arranje espaço no mapa para ser criada, a estação simplesmente não chega a ser criada.

7.2 Agente Utilizador

Para a implementação do Agente Utilizador foram definidos os seguintes comportamentos JADE:

- `Request extends OneShotBehaviour`

  Responsável por definir as estações de partida e destino, enviando uma mensagem do tipo *Request* à respetiva estação de partida.

- `Reply extends CyclicBehaviour`

  Responsável por receber respostas da estação. Estas podem ser dos seus pedidos de aluguer ou de devolução ou também os incentivos. No entanto, como são comportamentos bastantes diferentes entre si, após receber a mensagem da estação, redireccionamos para 1 de 3 *OneShotBehaviours*, sendo estes *HandleConfirms*, que trata as confirmações de aluguer/devolução da estação, *HandleRefuses*, que trata as rejeições de aluguer/devolução da estação, e *HandleIncentivos* que escolhe se quer aceitar ou não o incentivo que a estação propôs.

- `AtualizaPosicao extends TickerBehaviour`

  Responsável por manter o utilizador no seu trajecto da estação inicial para a destino. É aqui que a sua posição actual é actualizada, utilizando para isso cálculo de vectores unitários e direccionando o utilizador no sentido e direcção do vector. Após fazer os cálculos, testa se chegou ao seu destino ou se ainda para lá caminha. No primeiro caso faz um pedido à estação destino que pretende devolver a bicicleta, e no outro caso, informa ao agente central a sua nova posição. Este último é um processo muito importante para depois a central conseguir saber se o utilizador já passou dos 3/4's do seu trajecto total e informar todas as estações que tem esse utilizador nas suas APEs que já podem começar a enviar incentivos.

7.3 Agente Estação

Para a implementação do Agente Estação foram definidos os seguintes comportamentos JADE:

- `Register extends OneShotBehaviour`

  Responsável por definir a sua posição, enviando uma mensagem do tipo *Request* à central para o seu registo. Como foi referido anteriormente, foi usado funções de proximidade para calcular de forma realista os lugares aleatórios de cada estação, e é neste comportamento que esse processo acontece.

- `Receiver extends CyclicBehaviour`

  Responsável por receber pedidos de aluguer/devolução de bicicletas vindas de um utilizador e também por receber mensagens do central, informando que um utilizador se encontra na sua APE. No entanto, como são comportamentos bastantes diferentes entre si, após receber a mensagem, redireccionamos para 1 de 3 *OneShotBehaviours*, sendo estes *HandlePedidoAluguer*, que trata dos pedidos de aluguer e calcula se é possível realizar tal pedido, enviando uma mensagem como resposta do tipo *Confirm* ou *Refuse*, *HandlePedidoDevolucao*, que faz o mesmo que o comportamento anterior só que para o caso de ser um pedido de devolução e por fim *EnviaIncentivo* que trata de calcular o incentivo e propõe-no ao utilizador que a central lhe disse que estava na sua APE. O cálculo do incentivo é feito a partir do seu grau de ocupação, quanto maior for menor será o incentivo.

7.4 Agente Central

Para a implementação do Agente Central foram definidos os seguintes comportamentos JADE:

- `Receiver extends CyclicBehaviour`

  Responsável por receber os pedidos de registos, tanto do utilizador como da estação, e também mensagens vindas do utilizador a informar as suas novas posições ou a informar que realizou a devolução e está pronto para ser removido do sistema. No caso de informar as novas posições, vai verificar se o utilizador ultrapassou dos 3/4's do trajeto, e se sim, vai buscar todas as estações cujas APEs englobam a posição atual do utilizador e envia depois para elas que têm tal utilizador na sua APE.

- `SendMap extends TickerBehaviour`

  Responsável por enviar periodicamente para o agente interface o estado do programa através de uma mensagem do tipo *Inform*.

7.5 Agente Interface - Mapa

O agente interface recebe várias informações por parte do agente central de modo a apresentar o mapa com todas as estações e com todos os utilizadores e respectivas posições dos mesmos em tempo real. Como tal, através de um *Cyclic-Behaviour* verificamos se a mensagem era *Performative Inform* (ou seja, o central informa a interface do estado do mapa) e prosseguimos com a apresentação, caso contrário assumimos que o agente interface recebeu uma mensagem inválida. Todo este processo acontece no *ReceiveMap behaviour* enquanto que o envio de informação sobre o estado do mapa no central acontece no *SendMap behaviour* que se trata de um *TickerBehaviour*.

Quanto à apresentação da informação em si, optamos por não usar nenhuma biblioteca externa do Java (*Swing* p.e.) e sim versão texto no terminal, não só pela facilidade como também não sentimos que fosse uma componente necessária no contexto desta unidade curricular. De seguida, será demonstrada a nossa abordagem à apresentação do estado da aplicação.

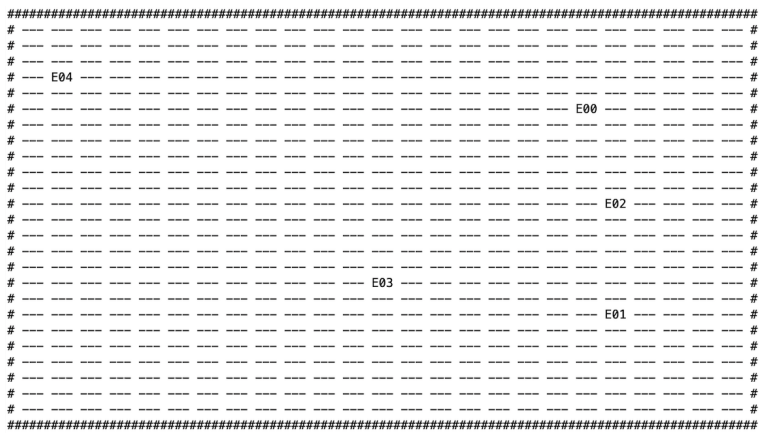

Figura 14. Estado inicial

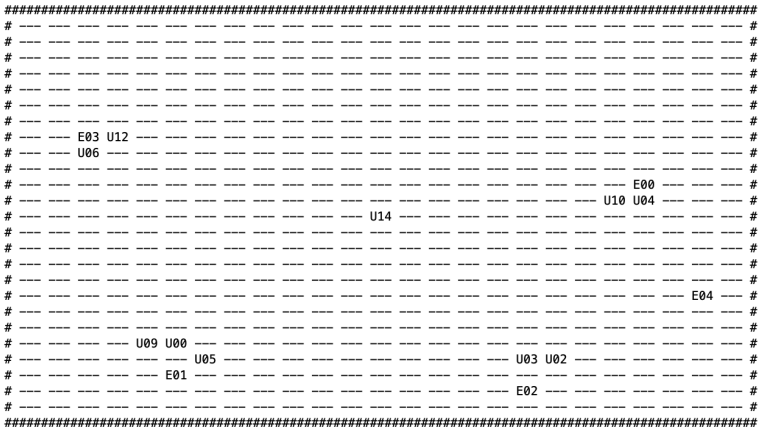

Figura 15. Estado normal

Como podemos ver nas figuras, as estações (representadas por "E"seguido do número) diferem de locais, isto porque foram tiradas em diferentes execuções do programa. Importante referir que a segunda imagem é o estado "normal"do programa e não o final porque, ao nosso ver, o programa não deve ter um fim, mas sim um contínuo processo de criação de mais utilizadores querendo usar a ferramenta. Na segunda imagem também podemos ver onde é que os utilizadores (representados por "U"seguido do número) terminam o seu trajeto, situando-se próximos da estação destino, e também podemos ver os utilizadores a aproximarem-se do seu destino (p.e. o U14 partiu de E04 e está a meio do caminho para E03).

Através de operações de *clear screen* e rápidas execuções dos comandos de impressão, esta apresentação no ecrã é em tempo real, sendo possível acompanhar todo o trajeto de um utilizador desde a sua estação inicial até à final.

7.6 Agente Interface - Estatísticas

De forma a apresentar a análise de performance dos agentes estação, decidimos fazê-los através de gráficos, recorrendo à API *JFreeChart*. Deste modo, desenvolvemos 3 gráficos. Um de barras que contém o número de requisições e devoluções de cada estação até ao momento. Temos também um *pie chart* com a percentagem de preenchimento de cada estação que é calculada com o número de bicicletas presentes e com o número máximo de bicicletas permitidas numa estação. E por fim, temos um gráfico de linhas que vai mostrando, ao longo do tempo de execução, quantas bicicletas disponíveis cada estação tinha a cada segundo desde o início do programa.

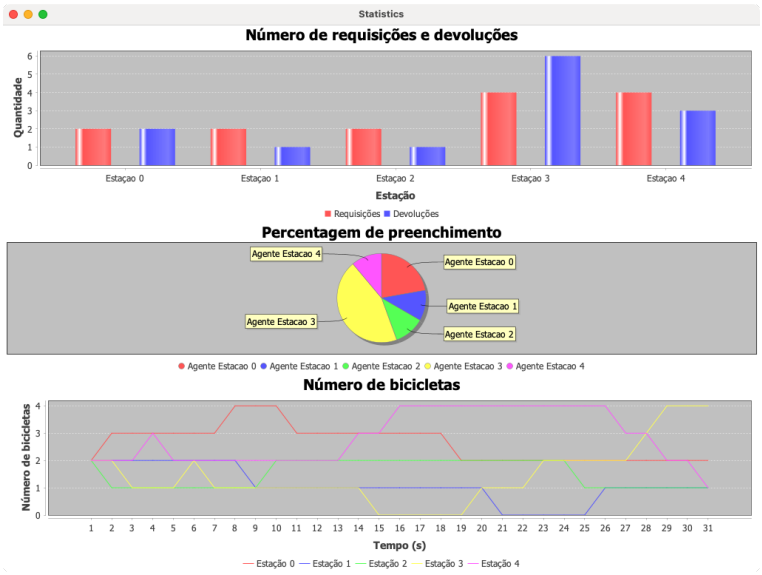

Figura 16. Resultados estatísticos

8 Conclusão e Trabalho Futuro

Concluído o projeto, podemos agora afirmar que foi adquirido conhecimento sobre os conceitos de Agentes, tendo sido estes aplicados quer na modelação como posteriormente na implementação deste que é um problema real.

Tendo como objetivo a implementação de um sistema multi-agentes para solucionar o Problema do Reequilíbrio de Partilha de Bicicletas, na primeira fase, foi apresentada toda a modelação e documentação necessária para caracterizar a nossa proposta de solução. Contudo, tentámos que esta fosse o mais abstrata e o mais geral possível para que houvesse espaço de manobra para alterar alguns pormenores aquando do desenvolvimento do código correspondente.

A fase 2 deste trabalho prático consistiu na implementação da arquitetura proposta nesta etapa. Consideramos que foi desafiante integrar todos os pormenores definidos na fase 1, pois alguns tiveram que ser modificados e até simplificados.

Como trabalho futuro o objetivo seria tornar o simulador o mais verídico possível, implementando, por exemplo, um tamanho variável do raio de cada APE e também o números de bicicletas máximo por cada estação. Outra possibilidade seria criar um *randomizer* viciado para inclinar mais a escolha para certas estações para simular a popularidade que as estações poderiam ter.