Agentes Inteligentes
Fase 1

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 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. De seguida, apresentaremos o Protocolo de Comunicação e, por último, o Processamento Interno dos Agentes. Em cada um dos últimos quatro tópicos, recorremos ao *Agent UML* para formalizar os protocolos de interação entre agentes.

2 Arquitetura

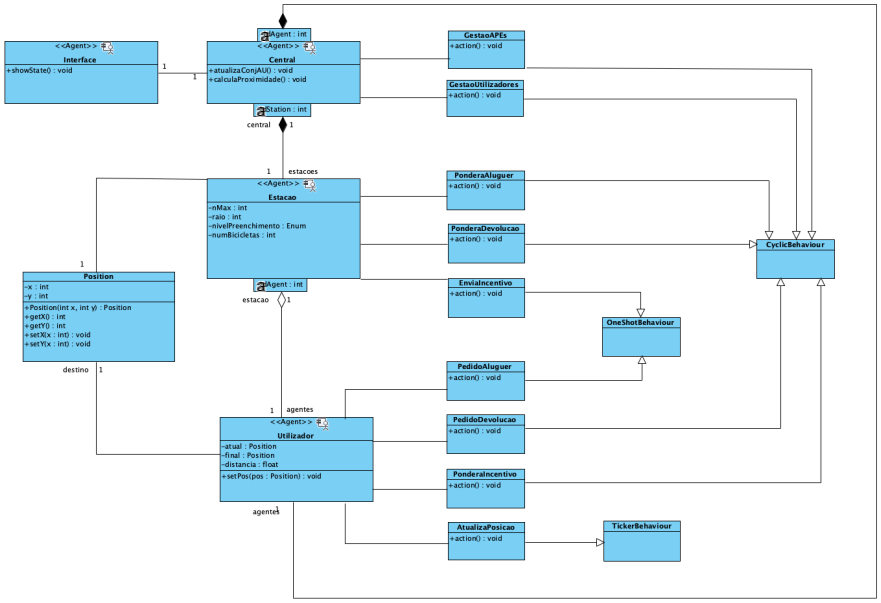

Figura 1. Diagrama de classes do sistema

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 agente tem

um conjunto de Agentes de Estação do sistema e outro de todos os seus Agentes Utilizadores. De modo a gerir os Agentes Utilizadores do sistema, este agente apresenta um método que atualiza o conjunto dos mesmos, caso um novo Agente Utilizador seja adicionado ou removido. Apresenta também outro método que calcula se um Agente Utilizador se encontra próximo de alguma estação. A receção das informações relativas à adição e remoção de Agentes Utilizadores é feita através da classe GestaoUtilizadores e a gestão das APEs de cada estação é efetuada pela classe GestaoAPEs. Foi escolhido o tipo de comportamento cíclico, *CyclicBehaviour*, para estes dois comportamentos, visto que é necessário que o Agente Central esteja constantemente à espera de alguma alteração relativa aos Agentes Utilizadores e Agentes de Estação;

- **Agente de Estação**: este agente representa uma estação que ocupa um determinado lugar do mapa do sistema. Assim, este agente apresenta o conjunto de Agentes Utilizadores presentes na sua APE, o número máximo de bicicletas disponíveis que podem estar na estação, o número de bicicletas atual, o raio da APE e, por fim, o nível de preenchimento da estação (baixo, médio ou alto). Serão utilizadas as classes PonderaAluguer e PonderaDevolucao para receber e lidar com os pedidos de aluguer e devolução de bicicletas, respetivamente. A classe EnviaIncentivo é responsável por enviar um incentivo a um determinado Agente Utilizador para persuadir o último a devolver a bicicleta na estação em questão. De modo a definir os comportamentos de ponderação de aluguer e ponderação de devolução, foi usado o tipo de comportamento cíclico, visto que o Agente de Estação está constantemente à espera de pedidos de aluguer/devolução. No caso do envio do incentivo, o Agente de Estação realiza esta ação apenas uma vez, logo esta foi implementada recorrendo ao tipo de comportamento *OneShotBehaviour*;

- **Agente Utilizador**: representa um agente associado ao dispositivo pessoal de um utilizador. Este agente tem a posição atual e a posição destino, assim como a distância entre ambas. O Agente Utilizador pode atualizar a sua posição, caso ele se movimente, utilizando a classe AtualizaPosicao. Pode também fazer um pedido de aluguer ou devolução a um Agente de Estação através das classes PedidoAluguer e PedidoDevolucao e vai poder ponderar se aceita ou rejeita o incentivo apresentado pelo Agente de Estação, usando a classe PonderaIncentivo. Para a definição dos comportamentos de pedido de devolução e ponderação de incentivo foi escolhido o tipo de comportamento cíclico, dado que, no primeiro caso, o Agente Utilizador poderá ter de ficar à espera que haja espaço para poder devolver a bicicleta na estação destino desejada e, no segundo caso, poderá rejeitar um incentivo enviado por um determinado Agente de Estação e aceitar um incentivo enviado por outro. A atualização da posição irá ser feita tendo em conta um tempo predefinido e, portanto, será necessária a aplicação de um *TickerBehaviour*. Por fim, o pedido de aluguer será feito apenas uma vez, o que implica o uso do com-

portamento *OneShotBehaviour*;

- **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 executar esta última ação, o Agente *Interface* apresenta a opção de mostrar o estado das estações ao utilizador.

3 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.

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.

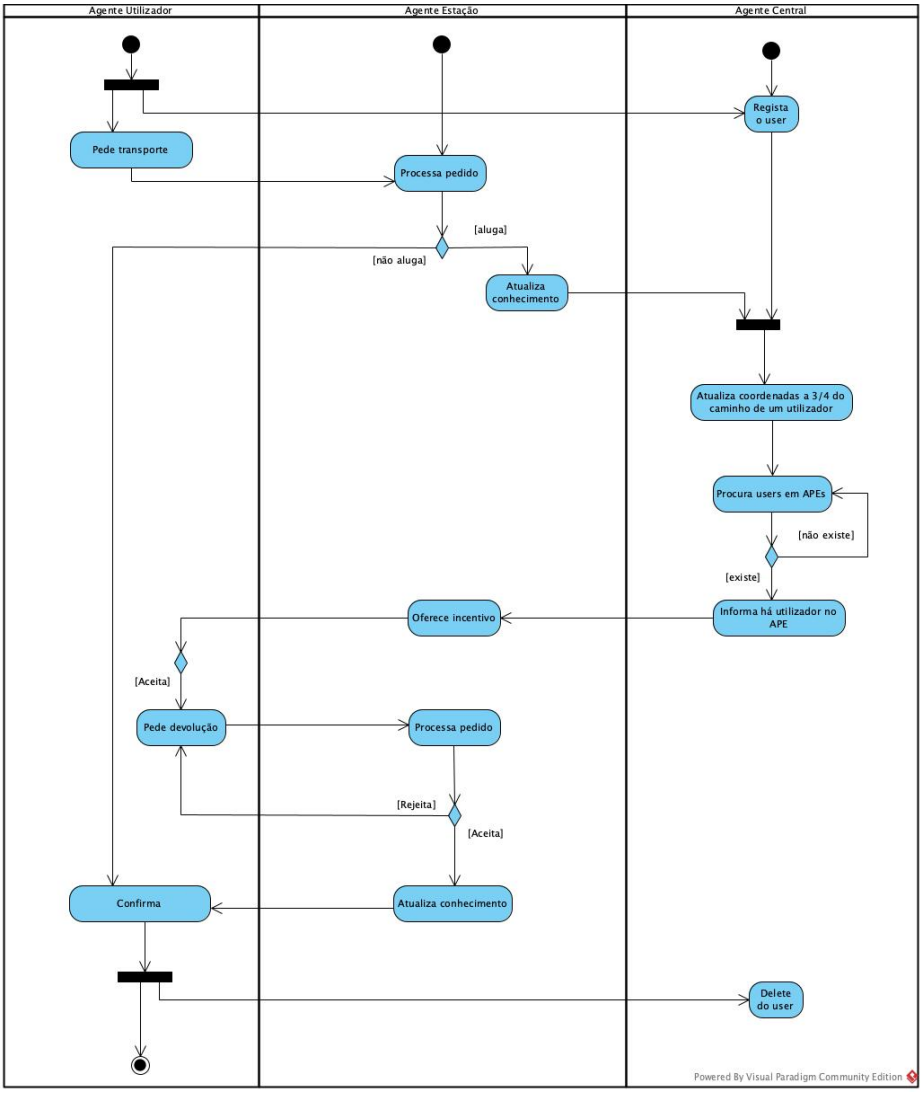

Figura 2. Fluxo de Resolução de aluguer

4 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*.

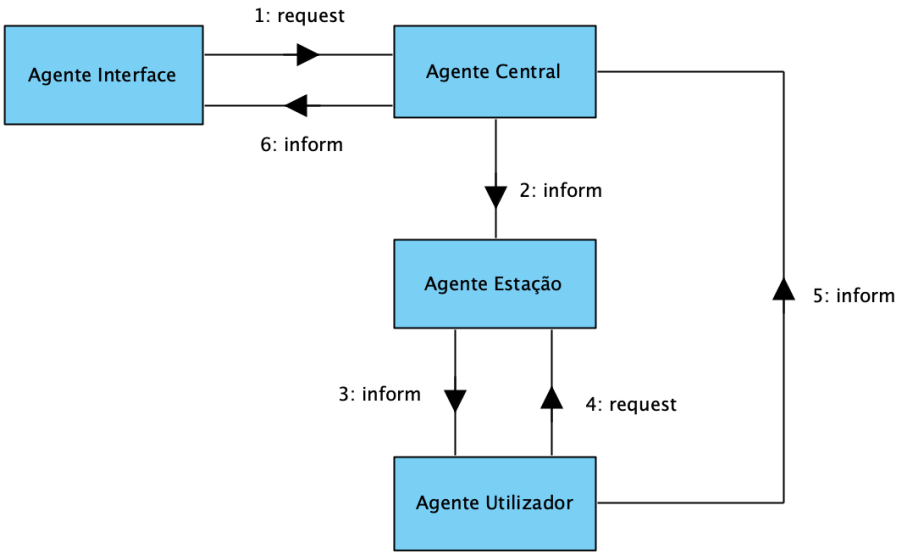

Figura 3. Protocolo de Comunicação geral

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

4.1 Novo Agente Utilizador

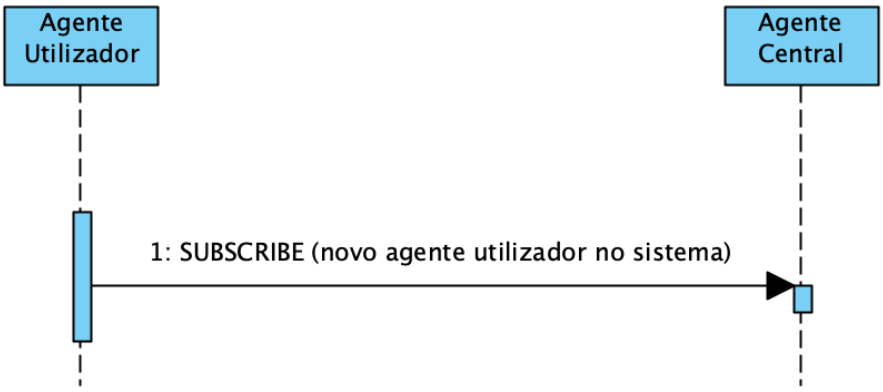

Figura 4. Diagrama de sequência correspondente à criação de um Agente Utilizador

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*.

4.2 Aluguer de uma bicicleta

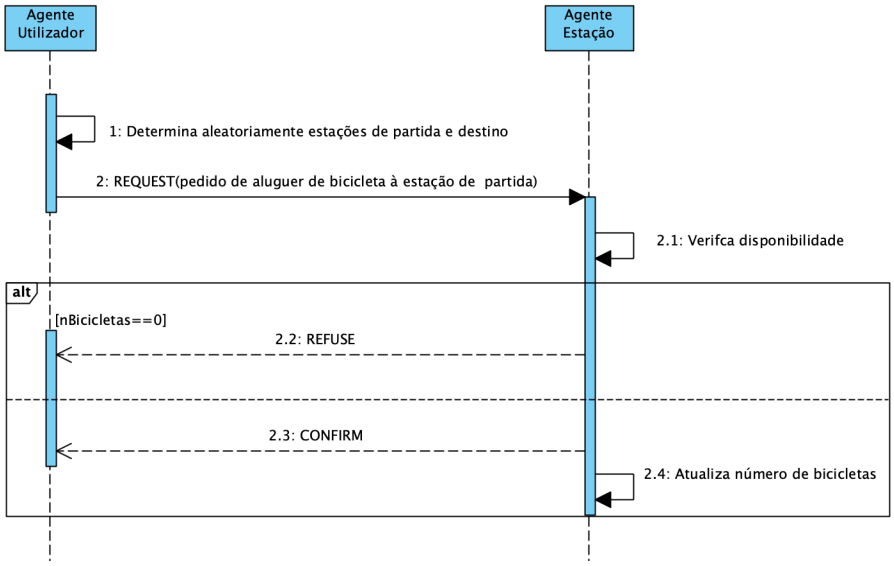

Figura 5. Diagrama de sequência correspondente ao aluguer de uma bicicleta

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.

4.3 Devolução de bicicleta

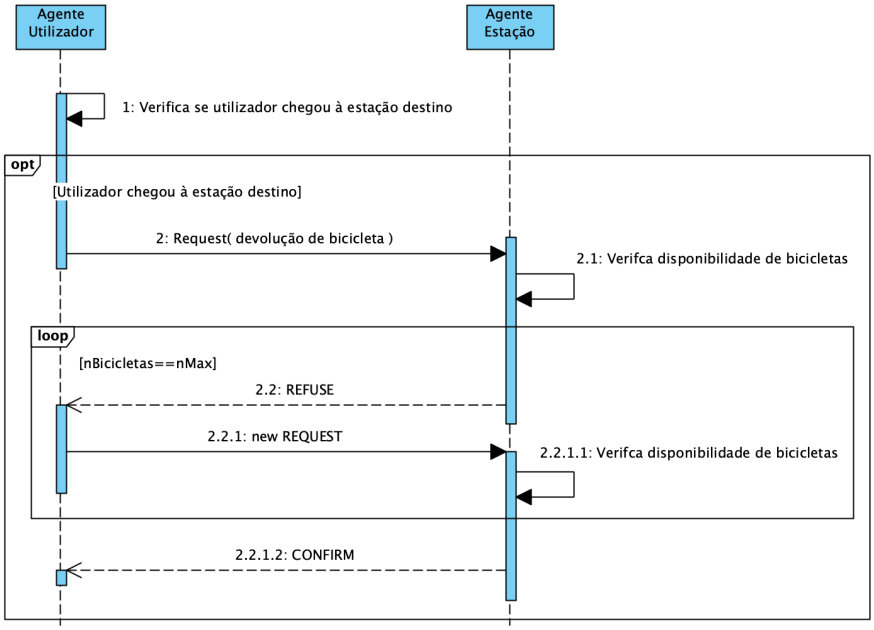

Figura 6. Diagrama de sequência correspondente à devolução de uma bicicleta

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.

4.4 Gestão de APEs

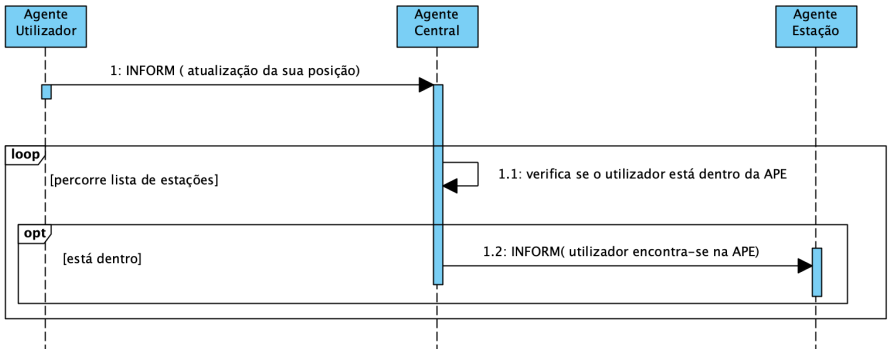

Figura 7. Diagrama de sequência correspondente à gestão de APEs

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.

4.5 Envio de incentivos

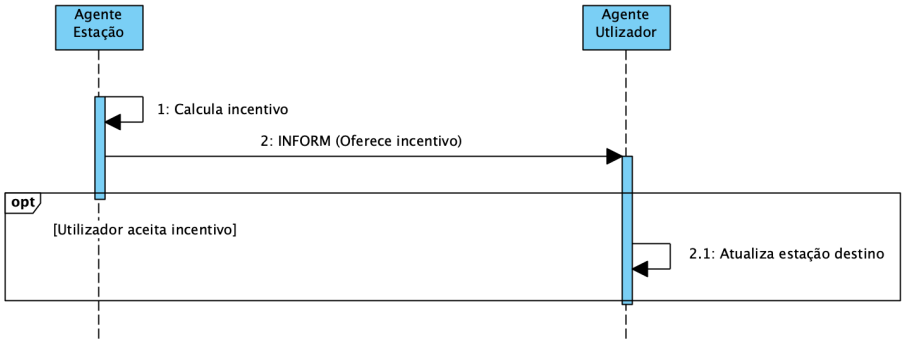

Figura 8. Diagrama de sequência correspondente ao envio de incentivos

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.

4.6 Monitorização da gestão de bicicletas

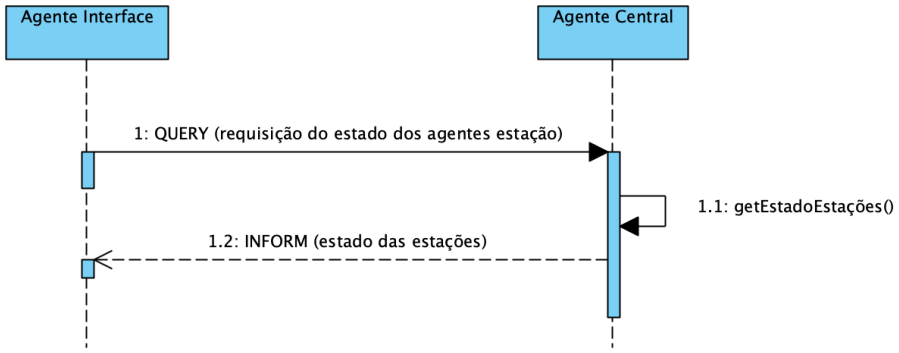

Figura 9. Diagrama de sequência correspondente à monitorização da gestão de bicicletas

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*.

5 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.

5.1 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.

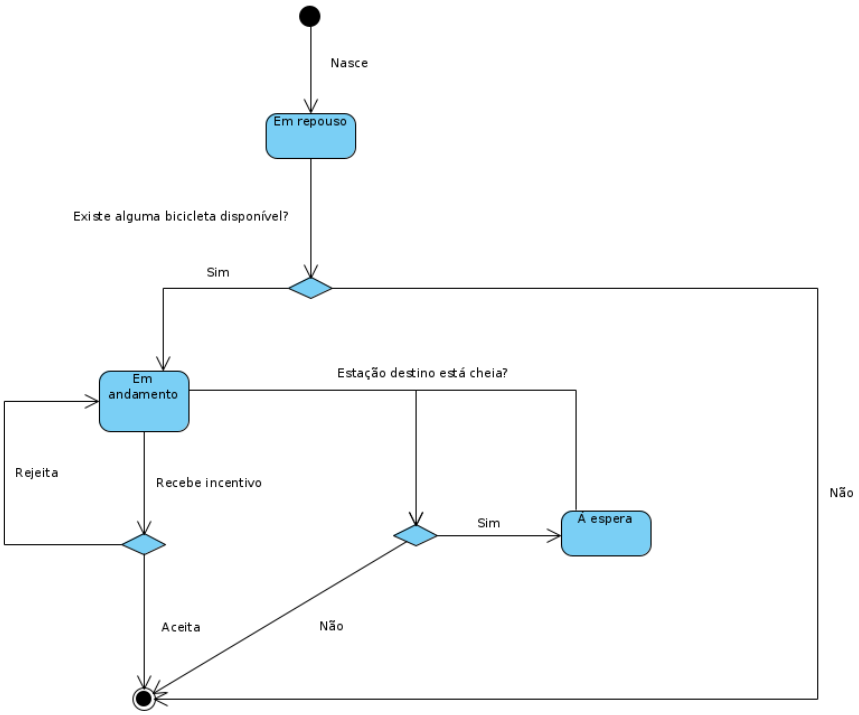

Figura 10. Máquina de estados do Agente Utilizador

5.2 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.

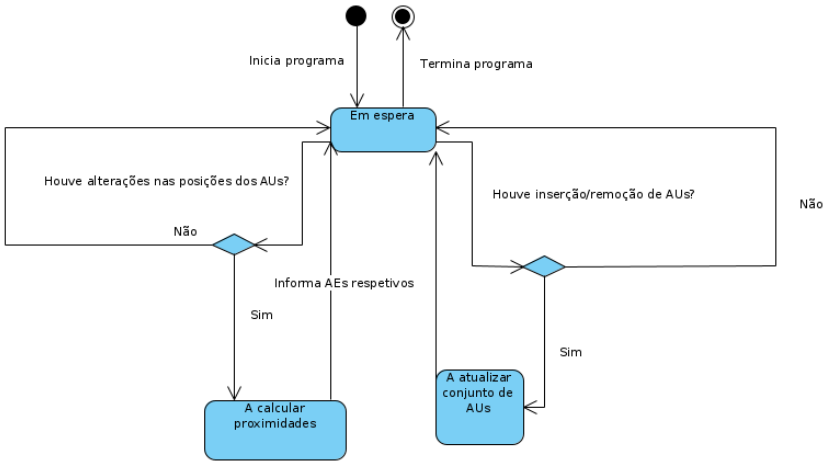

Figura 11. Máquina de estados do Agente Central

5.3 Agente de Estação

Como podemos ver na Figura 12, 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.

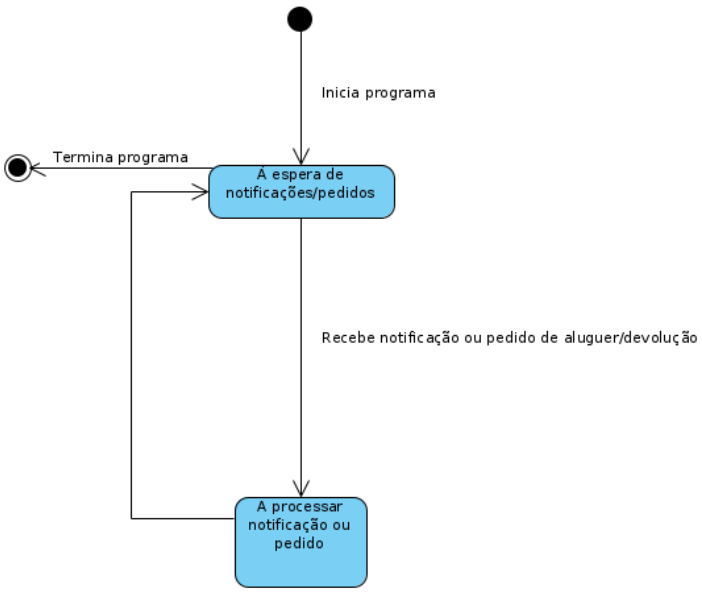

Figura 12. Máquina de estados do Agente de Estação

6 Conclusão

Tendo como objetivo a implementação de um sistema multi-agentes para solucionar o Problema do Reequilíbrio de Partilha de Bicicletas, nesta 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 consistirá na implementação da arquitetura proposta nesta etapa. Avaliando o tempo restante, consideramos que será desafiante integrar todos os pormenores definidos neste documento, sendo que alguns destes possivelmente serão modificados (como já referimos anteriormente) e/ou simplificados. No final, pretendemos ter uma aplicação com todos os requisitos e capaz de responder ao problema em questão.