Universidade do Minho
Escola de Engenharia

Agentes e Sistemas Multiagente

Trabalho de Grupo

Mestrado Integrado em Engenharia Biomédica

Informática Médica

Sistemas Inteligentes

2º semestre

2020/2021

Autores:

57768 Bruno Rebelo Lopes

83624 João Miguel da Silva Alves

84480 Paulo Jorge Alves

Docentes:

Paulo Novais

Filipe Gonçalves

Braga

2 de junho 2021

Resumo

Este trabalho visa articular de forma autónoma a gestão e manutenção de farmácias atendendo à redução de stocks e à facilidade de acesso a medicamentos por parte dos cidadãos. Apresenta uma melhoria na gestão de recursos e simplifica o processo de compra de medicamentos.

Permite ao cidadão de forma simples e rápida adquirir medicamentos das farmácias mais próximas usando para esse efeito as suas localizações, permite ainda uma gestão eficiente do stock nas farmácias prevendo os stocks necessários e emitindo encomendas a fornecedores de forma automática. O sistema é gerido por um agente Gestor, sendo responsável por manter os stocks das farmácias calculando para esse efeito quando é que é necessário comprar mais medicamentos. Nesta fase este escolhe qual o melhor fornecedor para determinado medicamento atendendo ao valor que cada fornecedor pratica para esse medicamento efetuando, assim, a encomenda das unidades necessárias do mesmo e requisitando ao fornecedor que estes sejam enviados diretamente para a farmácia em questão. Para acompanhar o desempenho das farmácias o agente gestor envia alguns dos dados para o agente interface que imprime os mesmos de modo que o utilizador da plataforma (administrador/gestor) possa verificar o lucro, as vendas e mesmo o stock de cada farmácia.

Através deste relatório, tentou-se recriar uma aplicação do setor farmacêutico, criando agentes para cada interveniente deste tão complexo setor da área da saúde. Esta recriação foi baseada na atualidade, no sentido em que tentou simular um serviço online de compra e venda de medicamentos ou produtos usualmente vendidos em farmácias.

Keywords: Farmácias; Sistemas Inteligentes; Sistemas Multiagente; Medicamentos; JADE.

Índice

Índice de Figuras

1 Introdução

O presente trabalho tem como objetivo o desenvolvimento de uma arquitetura autónoma distribuída para a melhoria dos serviços de saúde para cidadãos, pacientes e profissionais, através da monitorização, gestão e antecipação das suas necessidades de saúde.

Foi proposta a realização de um Sistema Multiagente que possa caracterizar a qualidade dos serviços de saúde e a sua acessibilidade com o uso da tecnologia digital. Para além disso, necessita também de refletir a evolução temporal do lucro das farmácias, bem como a melhoria do trabalho no setor farmacêutico, contribuindo para a prevenção eficaz de doenças. Por fim, deve fortalecer a igualdade dos cidadãos e economizar o seu tempo, através da limitação do número de consultas médicas.

Realizou-se, assim, um Sistema Multiagente constituído por 5 agentes: Cidadão, Farmácia, Gestor, Fornecedor e Interface. Cada um dos agentes apresenta características próprias (classes, *behaviours* e *performatives*).

De uma forma geral, o objetivo deste trabalho remete-nos para a construção de um Sistema Multiagente dentro do setor farmacêutico. Com isto, supõe-se que o SMA implementado consiga responder a todos os pedidos dos cidadãos. Após a receção dos pedidos, os agentes intervenientes terão que verificar stocks, reestabelecê-los em caso de falta de produtos e enviar o pedido do cidadão de forma correta. Além disto, obtêm-se também os resultados finais sobre a eficiência do sistema.

Hoje em dia já existem alguns softwares para a criação de SMA, contudo para este projeto foi usado o JADE - *Java Agent DEvelopment Framework*, implementado a partir do Eclipse.

Este relatório está dividido em 5 grandes capítulos: uma breve introdução sobre o que são Sistemas Multiagentes, o estado da arte, evidenciando plataformas farmacêuticas online, um outro capítulo, onde se mostram todos os diagramas efetuados durante o planeamento do projeto. De seguida, um capítulo relacionado sobre a framework Jade, uma vez que foi a usada para este trabalho e, por fim, um capítulo sobre a implementação do SMA, mostrando o comportamento e características dos agentes e, posteriormente, as demonstrações dos resultados deste mesmo Sistema Multiagente.

2 Sistemas Multiagente

É importante perceber o que faz com que um sistema seja capaz de representar conhecimento e, para isso, foram criados agentes. Um agente é uma entidade de software que aplica técnicas de Inteligência Artificial para escolher o melhor conjunto de ações a serem executadas de forma a atingir uma meta proposta pelo utilizador. Deve reagir de um modo autónomo, dinâmico, proativo, flexível e inteligente às mudanças produzidas no seu ambiente.

Embora a construção de agentes individuais seja muito importante, na grande maioria das aplicações de agentes autónomos, estes não vão trabalhar isolados, mas sim em conjunto com outros agentes e/ou humanos. Desta forma é importante explorar as formas de colocar os agentes a trabalhar em conjunto com outros agentes, ou seja, construir sociedades de agentes designadas vulgarmente por Sistemas Multiagente.

Um Sistema Multiagente compreende um conjunto de entidades (agentes) que cooperam por forma a solucionar um dado problema, o que normalmente está além das suas capacidades individuais. Minsk, em *The Society of Mind*, lançou os SMA como a base para a concretização da inteligência em sistemas computacionais, assumindo, contudo, que cada entidade constituinte pode ser o mais simples possível e que, da sua interação, podem emergir novas formas de inteligência.

Com o surgimento destes sistemas, a *Foundation for Intelligent Physical Agents* (FIPA), criou um protocolo e normas para a comunicação feita entre agentes, designada por ACL. Este passo facilitou ainda mais a forma como os agentes comunicam, assim como a forma de estes serem implementados, diminuindo assim quaisquer conflitos que possam surgir com os diferentes tipos de mensagens trocadas.

3 Estado da Arte

A Indústria Farmacêutica é responsável por produzir medicamentos. É uma atividade licenciada para pesquisar, desenvolver, comercializar e distribuir drogas farmacêuticas. O crescimento da Indústria Farmacêutica potenciou a descoberta de novos medicamentos, para doenças até então incuráveis. Novos fármacos trouxeram maior esperança média de vida, melhor qualidade de vida e enormes ganhos em saúde para a Humanidade nas últimas décadas. O desenvolvimento destas tecnologias envolve várias fases, que vão desde a pesquisa microscópica à produção em massa. Em todo este processo, a supervisão farmacêutica assegura o conhecimento técnico-científico e o respeito pelas boas práticas de fabrico. A Ordem dos Farmacêuticos atribui o Título de Especialista em Indústria Farmacêutica, uma qualificação profissional legalmente exigida para cargos de direção técnica nas unidades de produção [1].

A inteligência artificial é uma das grandes promessas do futuro com impacto em várias indústrias, como a farmacêutica. Supercomputadores são capazes de recolher e interpretar *big data* e ajudar as grandes farmacêuticas no desenvolvimento de novos medicamentos ou modificação de antigos. Um outro exemplo da aplicação da inteligência artificial nesta área, é o "Pillo", um pequeno robô doméstico que pretende ser um farmacêutico pessoal: responde a questões sobre saúde e bem-estar, conecta-se com profissionais desta área e faz a gestão de medicamentos [2].

É de conhecimento geral que a grande revolução social e económica deste século foi a Internet. Sendo esta a era da tecnologia e da criatividade, a Internet revolucionou as noções de rapidez e eficiência e originou novos métodos de a sociedade se organizar.

As empresas farmacêuticas necessitam de adaptar os seus negócios a esta nova realidade digital, e encará-la não como uma inimiga, mas como uma aliada para lidar com este novo perfil de consumidores. Para alcançar mais clientes e potencializar as vendas farmacêuticas é uma mais-valia investir em vendas online e realizar o seu marketing nesse mesmo ambiente. Para o consumidor, tornou-se mais cómodo, rápido e barato proceder a pagamentos, e adquirir bens ou serviços através de plataformas online.

Vista a importância do uso da tecnologia no setor farmacêutico, procedeu-se à pesquisa de serviços/aplicações já existentes.

A aplicação "Farmácias Portuguesas" dá, ao utilizador, através de qualquer smartphone, tablet ou computador, acesso aos produtos e medicamentos da farmácia. Este pode comprar, reservar, levantar na farmácia ou receber em casa os produtos pretendidos. Além disto, esta app permite também localizar as farmácias de serviço, reservar os medicamentos de

uma receita médica e monitorizar a correta toma da medicação, com alertas enviados para o smartphone no momento de cada toma [3].

A imagem abaixo mostra as funcionalidades desta app.

Figura 1: App Farmácias Portuguesas [3]

A Farmácia Entrega é uma outra plataforma que permite encomendar produtos ou medicamentos, com ou sem receita médica, e receber em casa. Neste momento, só está em funcionamento no concelho de Lisboa, Amadora e Pontinha.

No caso do utilizador ter receita médica, é só indicar o número da Guia de Prescrição - **figura 2**. No caso de não ter, este poderá descrever os sintomas que apresenta para receber aconselhamento farmacêutico, ou se já souber o que pretende, apenas indica os produtos que necessita - **figura 3**. Após a confirmação da encomenda, a farmácia entrará em contacto para aconselhar o utilizador, esclarecer qualquer questão existente e agendar a entrega [4].

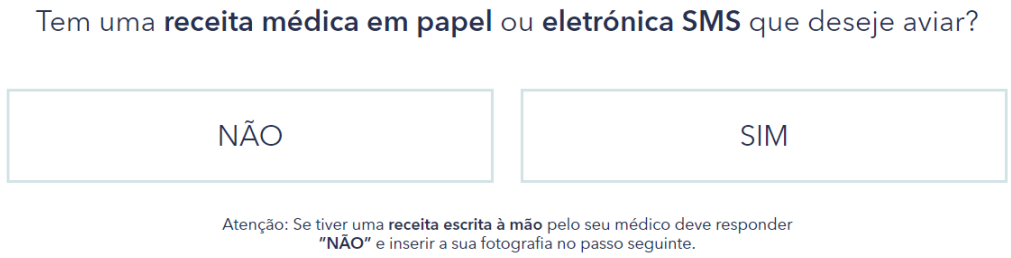

Figura 2: Página Inicial da Encomenda da *Farmácia Entrega* [4]

Figura 3: Encomenda sem receita médica digital em *Farmácia Entrega* [4]

Estes são apenas 2 exemplos, entre muitos outros serviços e aplicações, da vantagem do uso das novas tecnologias no acesso, por parte do cidadão comum, a produtos e medicamentos de uma farmácia no conforto da sua casa.

4 Esquema Inicial e Diagramas

Tendo por base os objetivos do trabalho já referidos anteriormente, foi, então, desenvolvido um esquema - **figura 4** - que conseguisse mostrar, de uma forma resumida, todos os passos e interações entre os diferentes agentes implementados no Sistema Multiagente.

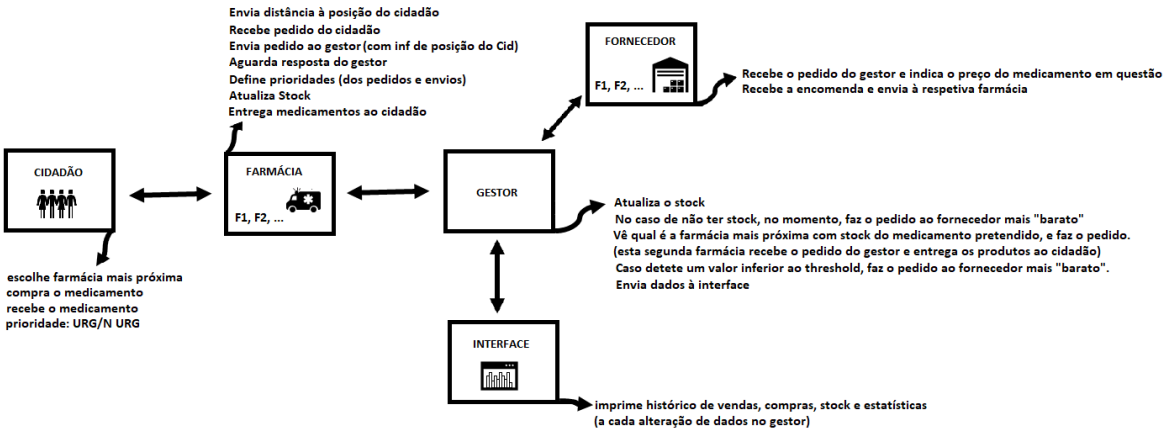

Figura 4: Esquema inicial

Para desenvolver o SMA pretendido foram criados 5 agentes: Cidadão, Farmácia, Gestor, Fornecedor e Interface. No esquema anterior estão descritas as ações de cada agente e a correspondente interação com outro agente.

Com o auxílio da API *Visual Paradigm* foram elaborados 4 diagramas, nomeadamente, o diagrama de Classe, diagrama de Atividade, diagrama de Comunicação e diagrama de Sequência.

De salientar que o recurso aos diagramas anteriormente mencionados permitiu organizar e planear toda a interação entre agentes, descrita no esquema da **figura 4**. Por essa razão e pelo facto das ações/interações serem apresentadas no esquema, toda a explicação sobre as mesmas é feita na secção de cada diagrama do presente relatório.

4.1 Diagrama de Classe

Um Diagrama de Classe apresenta a estrutura de cada classe e a forma como estas se correlacionam entre si. Cada classe é constituída por um nome, por atributos, os quais são representados pelo respetivo nome e tipo de dados que irão armazenar e por operações, que são, também, representadas por um nome, uma lista de parâmetros de entrada e pelo tipo de dados que são retornados.

Na **figura 5** está apresentado o Diagrama de Classe deste projeto.

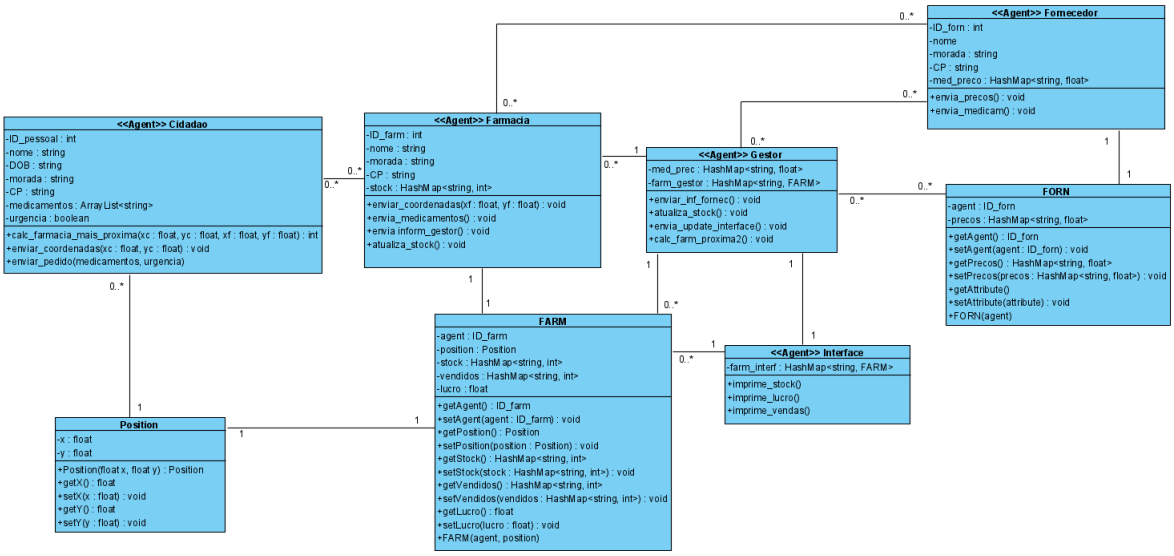

Figura 5: Diagrama de Classe

A classe *«Agent»Cidadao* pretende mostrar que quando um cidadão é "criado", este é constituído por um ID, nome, data de nascimento, morada, código-postal, e por um *ArrayList*, onde se encontram os medicamentos. O cidadão também apresenta um tipo de urgência do pedido (atributo booleano): não urgente (false) ou urgente (true).

No que diz respeito às operações, o agente "Cidadao", após pedir as coordenadas a todas as farmácias e ao recebê-las, calcula, com base na sua posição (Classe "Position", a qual contém uma posição (x,y)), a farmácia mais próxima para poder fazer o seu pedido - de modo a ter o menor gasto possível gasto, operação designada no esquema por "calc_farmacia_mais_proxima". De seguida, envia as suas coordenadas juntamente com o pedido para a farmácia escolhida anteriormente, operação representada por "enviar_coordenadas", que necessita da respetiva posição (x,y) do cidadão como entrada e por "enviar_pedido", que contém como input o medicamento pretendido e o tipo de urgência do pedido.

A classe *«Agent»Farmacia* apresenta como atributos: o id da farmácia, o nome, morada, código-postal e o respetivo stock, o qual é implementado usando um *HashMap*, em que as chaves correspondem ao nome do medicamento e os valores são o correspondente stock desse medicamento.

Apresenta como operações: a operação "enviar_coordenadas"ao cidadão que as pediu, a operação "envia_medicamentos", na qual é enviado o medicamento que o cidadão solicitou e a operação "atualiza_stock", que irá ser realizada quando recebe medicamentos pelo fornecedor ou quando envia um medicamento ao cidadão. Ao mesmo tempo, envia essa informação ao

gestor para este poder também atualizar o stock da respetiva farmácia.

A classe *«Agent Gestor»* apresenta como atributos dois *HashMaps*: um designado por "med_prec", onde cada chave corresponde a um medicamento e o valor é o preço do respetivo medicamento em todas as farmácias, e outro designado por "farm_gestor", no qual as chaves são o nome de cada farmácia e o valor corresponde à respetiva instância da classe "FARM".

A classe "FARM"permite representar cada farmácia por um objeto com os seus próprios atributos, nomeadamente o seu "agent ID", a sua posição, o seu stock, os seus produtos vendidos e o seu lucro. Desta forma, também apresenta um construtor e os métodos get() e set() associados a cada atributo.

Para além do que já foi mencionado anteriormente, o agente gestor apresenta como *behaviours*: "enviar_inf_fornec", no qual o gestor envia uma mensagem ao fornecedor para que este possa enviar o preço dos medicamentos pretendidos. Depois disto, seleciona o mais barato e faz o pedido; "calc_farm_proxima2", no qual encontra uma outra farmácia com o stock pretendido e mais próxima do cidadão que enviou o pedido, e envia os dados do pedido e da posição do mesmo; "atualiza_stock", onde atualiza o stock da(s) farmácia(s) envolvidas em qualquer um dos pedidos anteriores; "envia_update_interface", no qual envia informação à interface sempre que ocorre uma atualização no stock de uma farmácia.

A classe *«Agent Fornecedor»* apresenta como atributos: ID, nome, morada, código-postal e um *HashMap*, designado por "med_preco", onde cada chave corresponde a um medicamento e o valor é o preço do respetivo medicamento. Apresenta como *behaviours*: "envia_precos", onde, a partir da classe "FORN", envia as informações de medicamentos e preços ao gestor; "envia_medicam", onde envia os produtos para a farmácia em questão.

Por fim, a classe *«Agent Interface»* apresenta como atributos: um *HashMap* "farm_interf", no qual a chave corresponde ao nome de cada farmácia e o valor corresponde à respetiva instância da classe "FARM". No que diz respeito às suas operações, tem-se o "imprime_stock"para imprimir na consola o stock de cada farmácia, "imprime_lucro"para imprimir o lucro de cada farmácia e o "imprime_vendas", no qual imprime as vendas de cada farmácia.

4.2 Diagrama de Atividade

Um Diagrama de Atividade permite mostrar toda a atividade que ocorre num Sistema Multiagente. Para tal, é necessário colocar todos os agentes e os respetivos *behaviours*, e a correspondente interação entre os *behaviours* de diferentes agentes. Além disso, poderá ser necessário representar eventuais tomas de decisão que levarão à ocorrência de diferentes ações.

Na **figura 6** está apresentado o Diagrama de Atividade deste trabalho.

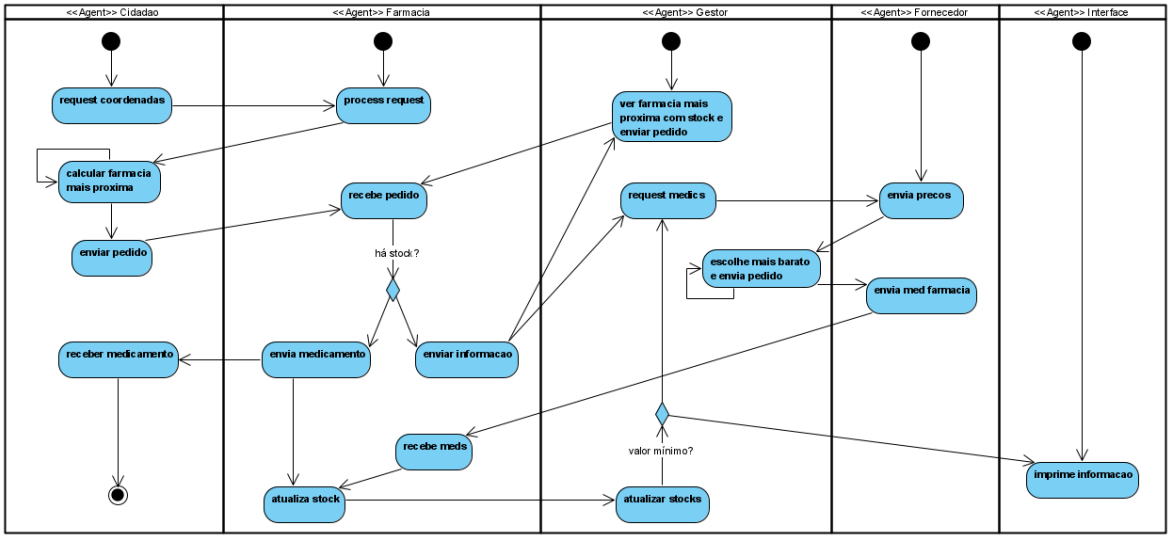

Figura 6: Diagrama de Atividade

Em primeiro lugar, começando pelo agente cidadão, este, através do "request_coordenadas" faz um *request* das coordenadas das farmácias que, por sua vez, vão processar esse pedido e enviam as suas coordenadas ao mesmo. De seguida, este calcula a farmácia que se encontra mais próxima da sua posição e envia um pedido do(s) medicamento(s) a essa mesma farmácia. A farmácia ao receber o pedido, verifica se há stock.

A partir desse momento poderão ocorrer duas situações distintas: a situação caso haja stock para esse medicamento nessa farmácia ou caso não haja stock.

No caso de haver stock, a farmácia, para o qual o pedido do cidadão foi feito, envia o(s) medicamento(s) para o cidadão, atualiza o seu stock e, por sua vez, envia mensagem ao gestor a informar a venda, para este poder também atualizar o stock da respetiva farmácia. Nesse momento, o gestor irá analisar este valor de stocks. Se o stock de algum medicamento (dos vendidos naquele momento) for superior a um valor de *threshold* (neste trabalho, vamos usar um valor de 5 para o *threshold*), então o gestor apenas envia mensagem ao agente interface e este, pelo facto de ter havido *updates*, acaba por imprimir na consola informações que já foram mencionadas no diagrama de Classe (lucro, vendas, etc. por farmácia). Pelo contrário, se o seu valor de stock for inferior a 5 (valor de threshold), então o gestor pede ao fornecedor informações sobre o preço do medicamento em causa e, após receber resposta, vê qual é o mais barato, pedindo a esse mesmo fornecedor que envie o medicamento em questão à farmácia que, por sua vez, atualiza o stock. Este volta a enviar os dados de atualização ao gestor, que atualiza o stock daquela farmácia e o agente interface imprime os resultados atualizados.

Por outro lado, caso não haja stock na farmácia, então essa farmácia envia uma mensagem ao gestor a dizer que não tem stock. Por sua vez, o gestor irá procurar a farmácia com stock

mais próxima do cidadão para que essa possa mandar esse medicamento ao mesmo e irá, com base na informação dos fornecedores, calcular o mais barato para esse medicamento e dizer a esse fornecedor que envie os medicamentos para a primeira farmácia, que não tinha stock. De seguida, ambas as farmácias (tanto a que recebe produtos, como a que vende ao cidadão) atualizam o stock e, consequentemente, o gestor também o faz, verificando se a segunda farmácia que enviou o pedido ao cidadão, tem ou não um valor superior ao *threshold*, acontecendo, depois, o mesmo que na situação acima. No fim, após cada atualização de stock seja da primeira, seja da segunda farmácia, é enviada uma mensagem ao agente interface para que imprima os resultados.

4.3 Diagrama de Comunicação

O Diagrama de Comunicação apresenta toda a comunicação entre os diferentes agentes.

Para este trabalho, dividimos este diagrama em dois blocos. Um para a situação de haver stock na farmácia e outro no caso de não haver.

Na **figura 7** está apresentado o Diagrama de Comunicação para a situação de haver stock.

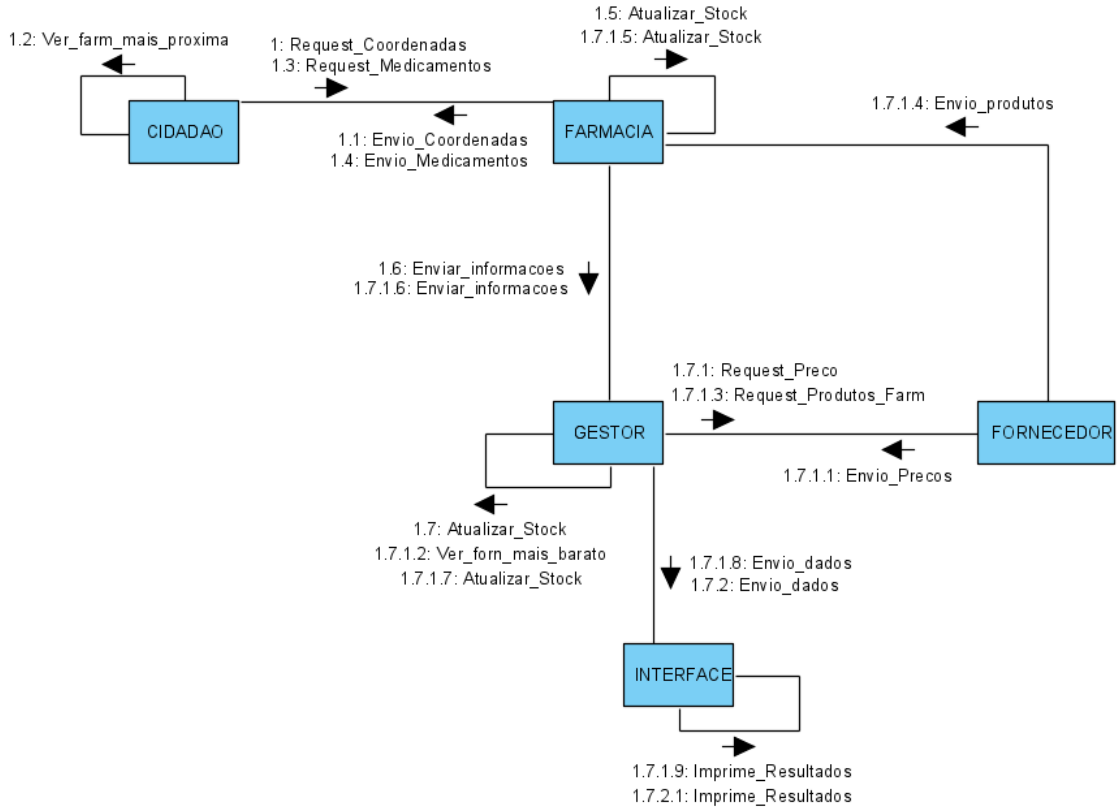

Figura 7: Diagrama de Comunicação - Caso particular I: Com stock

Nesta situação, o cidadão começa por enviar uma mensagem a pedir as coordenadas das farmácias (1 Request_Coordenadas) e por sua vez recebe essa informação de todas as farmácias (1.1 Envio_Coordenadas). De seguida, calcula a farmácia que está mais perto de si (1.2 Ver_farm_mais_proxima) e pede a essa farmácia o(s) medicamento(s) de que necessita (1.3 Request_Medicamentos) e, por sua vez, a farmácia envia esse medicamento ao cidadão (1.4 Envio_Medicamentos). A seguir, a farmácia atualiza o seu stock (1.5 Atualizar_Stock), envia essa informação ao gestor (1.6 Enviar_informacoes) e o gestor também atualiza o stock para essa farmácia (1.7 Atualizar_Stock).

É, neste momento, que o gestor irá ver se o stock da farmácia é ou não superior ao *threshold* (5). Caso seja igual ou superior, então é enviado essa informação ao agente interface (1.7.2 Envio_dados) e o agente interface apresenta as informações já anteriormente mencionadas (1.7.2.1 Imprime_Resultados). Pelo contrário, se o stock for inferior ao valor de threshold, então o gestor pede aos fornecedores, os preços para esse medicamento em causa (1.7.1 Request_Preco) e o fornecedores enviam (1.7.1.1 Envio_Precos). Depois, o gestor vê o fornecedor com o preço mais barato (1.7.1.2 Ver_forn_mais_barato) e pede a esse fornecedor para enviar o(s) medicamento(s) em questão à farmácia (1.7.1.3 Request_Produtos_Farm). Por fim, o fornecedor envia à farmácia (1.7.1.4 Envio_produtos), e esta atualiza o seu stock (1.7.1.5 Atualizar_Stock), envia essa informação ao gestor (1.7.1.1.6 Enviar_informacoes) e o gestor também atualiza o stock para essa farmácia (1.7.1.7 Atualizar_Stock). Por fim, tal como anteriormente, o gestor envia a informação ao agente interface (1.7.1.8 Envio_dados) e o agente interface imprime as informações (1.7.1.9 Imprime_Resultados).

Na **figura 8** está apresentado o Diagrama de Comunicação para a situação de não haver stock na farmácia escolhida pelo cidadão.

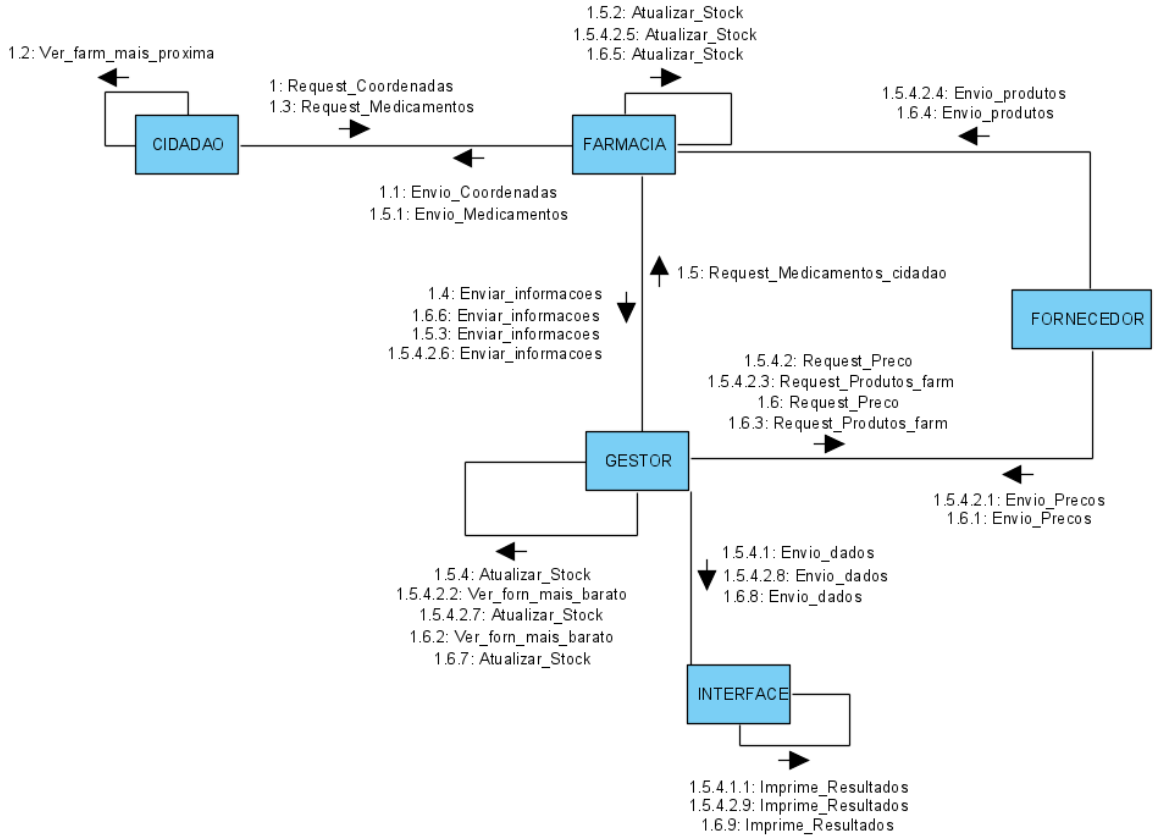

Figura 8: Diagrama de Comunicação - Caso particular II: Sem stock

Nesta situação, a parte inicial é semelhante. O cidadão começa por enviar uma mensagem a pedir as coordenadas das farmácias (1 Request_Coordenadas) e, por sua vez, recebe essa informação por todas as farmácias (1.1 Envio_Coordenadas). De seguida, calcula a farmácia que está mais perto de si (1.2 Ver_farm_mais_proxima) e pede a essa farmácia o(s) medicamento(s) de que necessita (1.3 Request_Medicamentos). A farmácia, como não tem stock para fazer a entrega ao cidadão, envia ao gestor a informação de que não tem o medicamento pretendido pelo cidadão (1.4 Enviar_Informacoes).

A seguir, o gestor vê a farmácia com stock mais próxima do cidadão e pede a essa farmácia para enviar o(s) medicamento(s) ao cidadão (1.5 Request_Medicamentos_cidadao) e essa farmácia envia, então, o medicamento ao cidadão (1.5.1 Envio_Medicamentos), atualizando o seu stock (1.5.2 Atualizar _Stock) e envia essa informação ao gestor (1.5.3 Enviar_informacoes). O gestor também atualiza o stock para essa segunda farmácia (1.5.4 Atualizar_Stock). Se nessa segunda farmácia o valor de stock for igual ou superior ao threshold então envia a informação ao agente interface (1.5.4.1 Envio_dados) e o agente interface imprime os resultados (1.5.4.1.1 Imprime_Resultados). Caso seja inferior ao valor de threshold, então o gestor pede aos fornecedores, os preços para esse medicamento em causa (1.5.4.2 Request_Preco)

e o fornecedores enviam (1.5.4.2.1 Envio_Precos). Depois, o gestor vê o fornecedor com o preço mais baixo (1.5.4.2.2 Ver_forn_mais_barato) e pede a esse fornecedor para enviar esse(s) medicamento(s) à farmácia (1.5.4.2.3 Request_Produtos_Farm). Por fim, o fornecedor envia à farmácia (1.5.4.2.4 Envio_produtos), a farmácia atualiza o seu stock (1.5.4.2.5 Atualizar_Stock), envia essa informação ao gestor (1.5.4.2.6 Enviar_informacoes) e o gestor também atualiza o stock para essa farmácia (1.5.4.2.7 Atualizar_Stock). O gestor envia a informação ao agente interface (1.5.4.2.8 Envio_dados) e o agente interface imprime as informações já anteriormente mencionadas (1.5.4.2.9 Imprime_Resultados).

Paralelamente à situação "1.5 Request_Medicamentos_cidadao", o gestor para a primeira farmácia que não possuía o medicamento, pede aos fornecedores, os preços para esse(s) medicamento(s) em causa (1.6 Request_Preco) e o fornecedores enviam (1.6.1 Envio_Precos). Depois, o gestor vê o fornecedor mais barato (1.6.2 Ver_forn_mais_barato) e pede a esse fornecedor para enviar o(s) medicamento(s) pretendido(s) à farmácia (1.6.3 Request_Produtos_Farm). Por fim, o processo é o mesmo. O fornecedor envia à farmácia (1.6.4 Envio_produtos), a farmácia atualiza o seu stock (1.6.5 Atualizar_Stock), envia essa informação ao gestor (1.6.6 Enviar_informacoes) para este poder atualizar o stock para essa farmácia (1.6.7 Atualizar_Stock) e enviar a informação ao agente interface (1.6.8 Envio_dados). O agente interface apresenta as informações atualizadas (1.6.9 Imprime_Resultados).

4.4 Diagrama de Sequência

O Diagrama de Sequência possibilita demonstrar toda a comunicação entre os diferentes agentes, dando ênfase ao nível temporal/sequencial, no qual decorrem estas interações. Além disso, permite que, no mesmo esquema, se possam representar duas situações distintas, que neste caso são a situação de haver, ou não, stock na farmácia.

Na **figura 9** está apresentado o Diagrama de Sequência deste projeto.

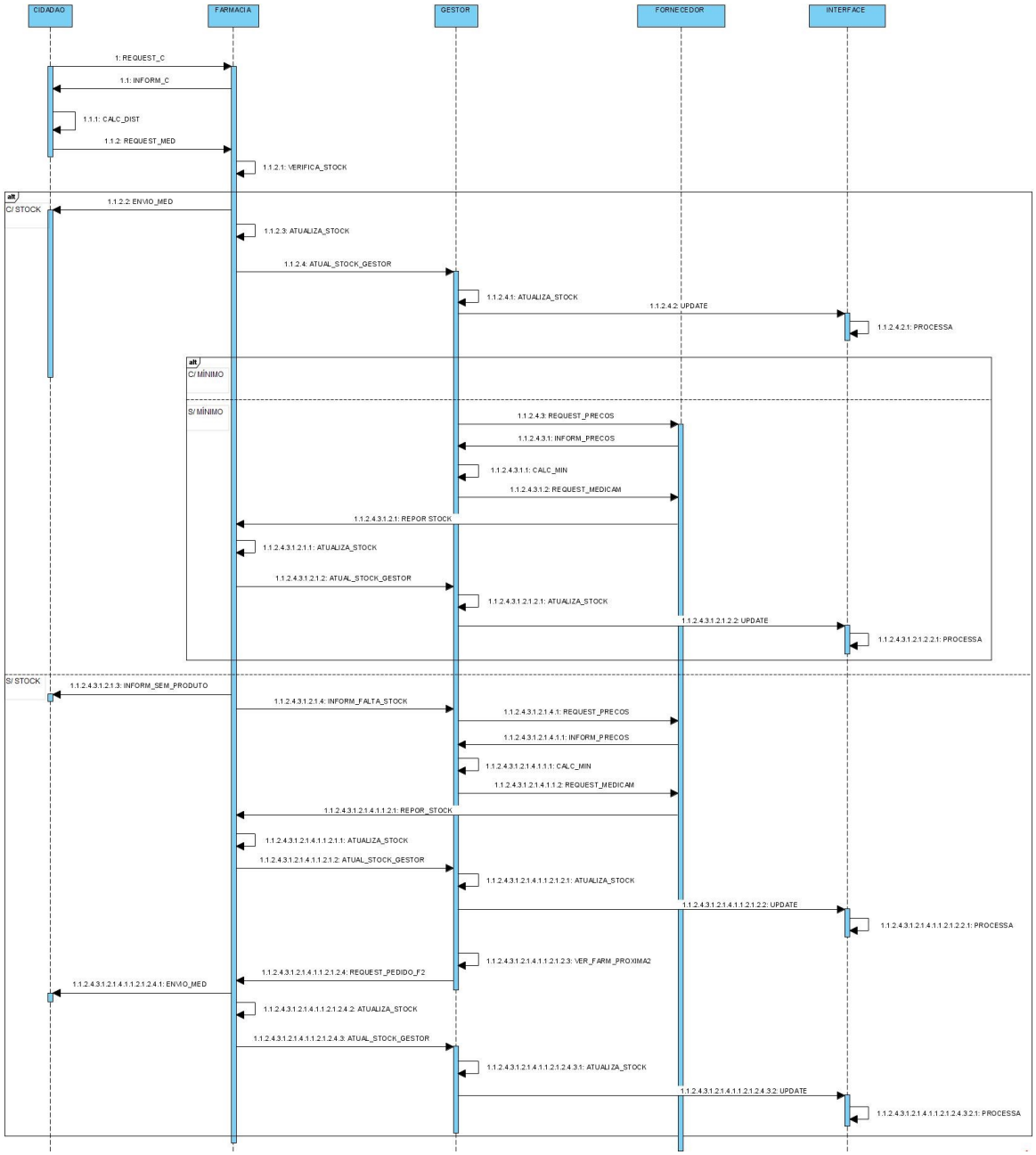

Figura 9: Diagrama de Sequência

Em primeiro lugar, o agente Cidadão pede as coordenadas de todas as farmácias (REQUEST_C) e estas enviam essas mesmas coordenadas (INFORM_C). De seguida, o cidadão calcula a farmácia mais próxima de si (CALC_DIST) e envia o pedido de medicamento(s) a essa farmácia (REQUEST_MED). Após receber o pedido, a farmácia irá verificar se possui stock para esse pedido (VERIFICA_STOCK). Caso possua stock, então essa farmácia envia o medicamento ao cidadão (ENVIO_MED) e atualiza o seu stock (ATUALIZA_STOCK). Depois, envia mensagem ao agente Gestor para que este possa também atualizar o stock da farmácia (ATUAL_STOCK_GESTOR). Por sua vez, o gestor atualiza o stock (ATUA-

LIZA_STOCK) e, a seguir, podem decorrer duas situações. Se o stock da farmácia for maior ou igual que o threshold (no nosso projeto, 5), então o gestor envia mensagem ao agente interface (UPDATE) e este apresenta as informações (PROCESSA). Se o stock for menor ao threshold, então o gestor pede os preços aos diferentes fornecedores (REQUEST_PRECOS), estes enviam os seus preços (INFORM_PRECOS) e o gestor vê qual deles é o mais barato (CALC_MIN). De seguida, pede ao fornecedor que envie esse medicamento à farmácia (REQUEST_MEDICAM) e o fornecedor envia esse medicamento à correspondente farmácia (REPOR STOCK). A farmácia atualiza o seu stock, tal como gestor e a interface acaba, também por imprimir as informações atualizadas.

Na situação em que a farmácia não possui o stock suficiente para o medicamento que o cidadão requisitou, esta envia ao gestor que não possui stock (INFORM_FALTA_STOCK). De seguida, o gestor pede os preços aos diferentes fornecedores (REQUEST_PRECOS), este enviam os seus preços (INFORM_PRECOS) e o gestor calcula qual deles é o mais barato (CALC_MIN). De seguida, pede ao fornecedor que envie esse medicamento à farmácia (REQUEST_MEDICAM) e o fornecedor envia esse medicamento à correspondente farmácia (REPOR STOCK). A farmácia atualiza o seu stock tal como gestor e a interface acaba, também por imprimir informações na consola. De seguida, o gestor também calcula a segunda farmácia mais próxima (VER_FARM_PROXIMA2) e envia mensagem a essa farmácia (REQUEST_PEDIDO_F2) para que envie o medicamento ao cidadão (ENVIO_MED). Por fim, essa farmácia atualiza o stock, envia estes dados para o gestor para este atualizar os seus (da respetiva farmácia) e o agente interface apresenta os resultados atualizados.

5 Framework JADE

JADE (*Java Agent DEvelopment Framework*) - **figura 10** - é uma estrutura de software, em código aberto, totalmente implementada em Java, que permite a construção de aplicações baseadas em agentes e que simplifica a implementação dos SMA por meio de um *middleware* que atende às especificações FIPA [5].

Um sistema baseado em JADE pode ser distribuído entre máquinas e a configuração pode ser controlada por meio de uma GUI remota. A configuração pode ser alterada em tempo de execução, movendo os agentes de uma máquina para outra, quando e da forma que for necessária. Além da abstração do agente, o JADE fornece um modelo simples, mas poderoso, de execução de tarefas, comunicação de agentes *peer to peer* com base no paradigma de passagem de mensagens assíncronas, um serviço de páginas amarelas que suporta o mecanismo de descoberta de assinaturas e muitos outros recursos avançados que facilitam o desenvolvimento de um sistema distribuído [6].

As principais vantagens do JADE estão presentes a nível da interoperabilidade, do uso de um standard bem fundamentado, da portabilidade do código e do conceito de agente.

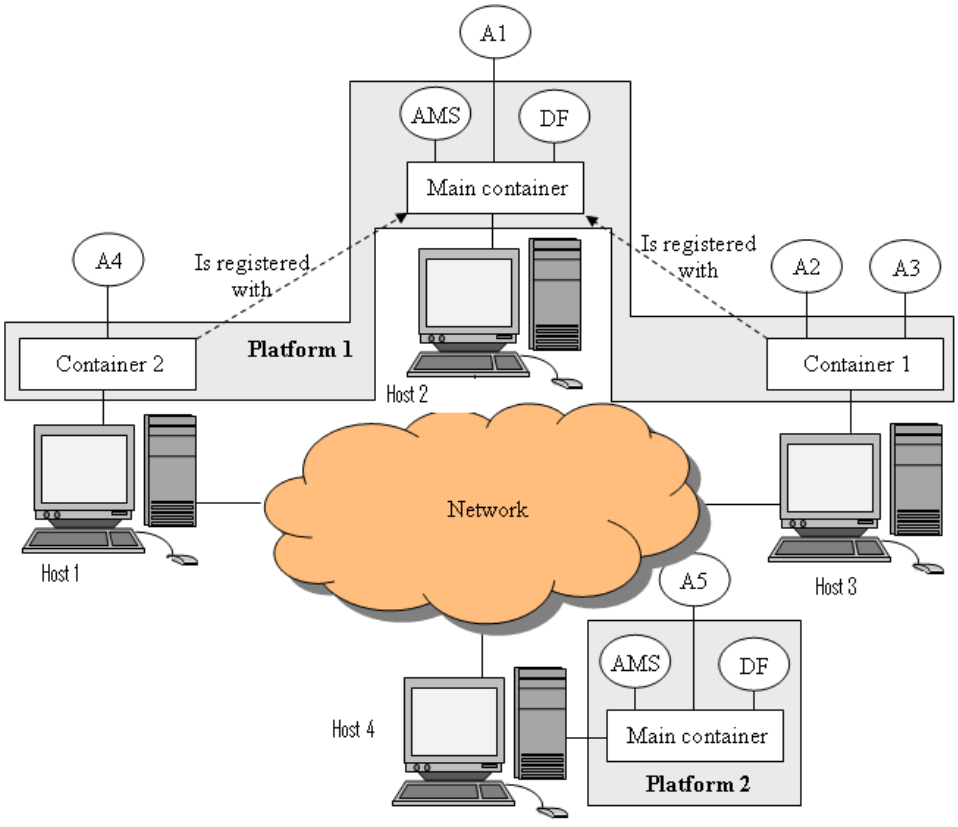

Figura 10: Arquitetura JADE [6]

5.1 Containers

Em JADE, um *container* corresponde a cada instância que é colocada a correr no ambiente de execução do JADE, onde este pode conter vários agentes. Deste modo, o conjunto de *containers* ativos designa-se por Plataforma.

Em primeiro lugar, o primeiro *container* a ser inicializado numa plataforma deve ser um *Main-Container* e posteriormente, os restantes *containers* (não principais) devem ser informados sobre o "host" e a "port" do *Main-Container* para se poderem registar. Se eventualmente outro *Main-Container* for ativado, o JADE assume que este constitui uma nova plataforma, para a qual, novos *containers* normais se podem registar.

Posto isto, um *Main-Container* difere dos *containers* normais pois contém 2 tipos de agentes fundamentais [7]:

- AMS (*Agent Management System*) - Este agente fornece o serviço de identificação, isto é, permite que cada agente da plataforma tenha um nome único. Além disto, também representa a entidade na plataforma capaz de criar ou "matar"agentes;
- DF (*Director Facilitator*) - fornece um serviço *Yellow Pages* - **figura 11** - permitindo que os agentes encontrem outros agentes necessários para atingir os seus objetivos.

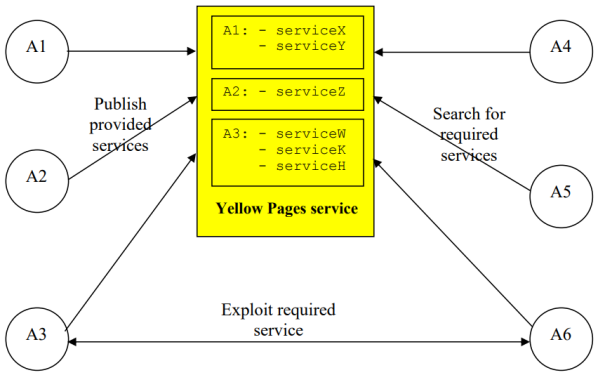

Figura 11: Serviço de Páginas Amarelas [6]

5.2 Comunicação entre Agentes

Os Agentes são criados através da extensão da classe "Agent" no Eclipse.

Contudo, é necessário programar o seu *setup()* de modo a adicionar-lhe *Behaviours*, que consistem em comportamentos que o agente irá realizar assim que se encontrar ativo. Cada agente tem ainda um identificador próprio que faz com que possa ser utilizado por outros [7].

Os agentes podem comunicar de forma transparente, independentemente de estarem no mesmo *container*, em *containers* diferentes (no mesmo ou em hosts diferentes) pertencentes à mesma plataforma ou em plataformas diferentes.

A comunicação baseia-se num paradigma de transmissão de mensagens assíncronas. O formato da mensagem é definido pela linguagem ACL, definida pelo FIPA. Uma mensagem ACL contém vários campos, incluindo [7]:

- o remetente;
- o(s) recetor(es);
- o ato comunicativo (também denominado performativo) que representa a intenção do emissor da mensagem. Por exemplo, quando um agente envia uma mensagem "INFORM", este deseja que o(s) receptor(es) tomem conhecimento de um facto. Quando um agente envia uma mensagem "REQUEST", deseja que o(s) receptor(es) realizem uma ação. O FIPA definiu 22 atos comunicativos, cada um com uma semântica bem definida;
- o conteúdo, ou seja, a informação real transmitida pela mensagem.

5.3 Behaviours

Como mencionado anteriormente, os behaviours representam o comportamento dos agentes, como eles interagem após a sua criação.

Os behaviours usados na implementação do sistema multiagente enunciado são [6]:

- *OneShotBehviours*: fazem com que a ação do agente seja executada apenas uma vez e que, de seguida, seja "morto". Contudo, esta última ação pode não ser realizada por completo, apresentando assim o agente como um agente residual que enviou ou recebeu apenas uma mensagem.
- *Cyclic-Behaviours*: fazem parte dos comportamentos mais usados. A sua implementação acontece enquanto o agente estiver ativo e for chamado repetidamente após cada evento.

6 Implementação

Na implementação do projeto, pequenas alterações foram realizadas relativamente ao planeamento. Uma delas foi diminuir o valor de threshold de 5 para 2, para permitir que a situação de chamar uma segunda farmácia mais próxima pudesse acontecer com mais frequência. Outra alteração foi descartar a variável urgente, a qual representava se o pedido do cidadão fosse urgente ou não, uma vez que se acabou por considerar que não fosse de enorme relevância para a implementação do Sistema Multiagente.

6.1 Arquitetura

Nesta secção é apresentada uma descrição sobre os diferentes agentes e classes do Sistema Multiagente criado.

Desta forma, este SMA apresenta os seguintes agentes/classes:

- Agente Cidadão;
- Agente Farmácia;
- Agente Gestor;
- Agente Fornecedor;
- Agente Interface;
- Classe Farmácia;
- Classe Fornecedor;
- Classe Posição.

6.1.1 Agente Cidadão

O agente Cidadão apresenta três classes: a classe Cidadao, na qual estão presentes as características e comportamento do cidadão, nomeadamente o seu nome, nif, data de nascimento, morada e código postal. Além disso, no método setup(), cada cidadão apresenta uma posição através da classe "Position", onde são geradas aleatoriamente duas coordenadas x e y, e são apresentados os diferentes medicamentos que o cidadão pode escolher. Também se encontra presentes três classes correspondentes a três *behaviours* diferentes. O primeiro designa-se por *Apresentação*, que corresponde a um *OneShotBehaviour*, em que quando um

determinado cidadão se inicializa, este apresenta-se com as suas características definidas anteriormente. De seguida, também se tem outro *OneShotBehaviour*, no qual cada cidadão contacta as diferentes farmácias, através das páginas amarelas, de modo a conseguir calcular qual delas se encontra mais próxima. Para tal, envia uma mensagem do tipo *SUBSCRIBE* para cada uma das farmácias, de forma a conseguir obter as suas coordenadas. O último behaviour é do tipo *Cyclicbehaviour*. Neste behaviour, o cidadão recebe constantemente mensagens até receber o produto que requisitou, sendo, que nesse momento, acaba por morrer. Neste *behaviour* o cidadão pode receber mensagens do tipo *INFORM*, onde recebe as coordenadas de cada farmácia que pediu anteriormente e verifica qual das farmácias se encontra mais perto. De seguida, escolhe aleatoriamente um produto e uma quantidade de 1 a 9 e envia esse pedido à farmácia mais próxima calculada no momento anterior. Por fim, quando o cidadão receber uma mensagem com a *performative* do tipo *AGREE*, este constata que recebeu o pedido que requisitou e acaba por se auto-destruir e outro cidadão acaba por se inicializar, repetindo o processo novamente.

6.1.2 Agente Farmácia

No agente Farmácia existem duas classes: a classe Farmacia, na qual cada farmácia apresenta as suas características como a sua posição e duas variáveis que correspondem a dois *hashmaps*. Um deles representa o stock de medicamentos em que a chave é uma string que corresponde ao nome do medicamento e o valor corresponde a um valor inteiro que será o respetivo stock. O outro *hashmap* representa o preço de cada medicamento, em que a chave é uma string que corresponde ao nome do medicamento e o valor corresponde a um valor inteiro que será o respetivo preço. No seu setup(), o agente farmacia regista-se nas suas páginas amarelas.

O agente farmácia também apresenta uma classe que corresponde a um behaviour, sendo que neste caso é um *Cyclicbehaviour*, no qual a farmácia estará constantemente à espera para receber mensagens. Nesta classe, a farmácia utiliza 6 *performatives*. A farmácia ao receber uma mensagem com a *performative* do tipo *SUBSCRIBE* significa que recebeu o pedido do cidadão para que lhe envie as suas coordenadas. Se a *performative* for do tipo *CFP* significa que recebeu o pedido do gestor para que lhe envie as suas coordenadas. Além disso, se a performative for do tipo *PROPOSE* significa que recebeu o pedido da interface para que lhe envie as suas coordenadas. Se a mensagem possuir a performative do tipo *REQUEST* mostra que a farmácia recebeu o pedido relativamente à quantidade e ao medicamento requisitado pelo cidadão. Neste momento, a farmácia verifica se o stock para esse medicamento é superior

ou igual à quantidade dita pelo cidadão. Se assim for, então manda uma mensagem do tipo *AGREE* ao respetivo cidadão a dizer que o seu pedido foi, de facto, entregue e altera o seu stock e a sua receita, já que vendeu um determinado produto e envia essa informação ao gestor. Se o stock do medicamento for inferior à quantidade requisitada pelo cidadão, então envia uma mensagem do tipo *REQUEST* ao gestor a dizer que não possui stock para o pedido do cidadão e, por isso, o gestor necessita de calcular a segunda farmácia mais próxima ao cidadão que tenha esse medicamento e essa quantidade e necessita, também, de calcular o fornecedor mais barato para que possa restabelecer o stock à primeira farmácia. Se a mensagem for do tipo *INFORM*, tal significa que a segunda farmácia recebeu a mensagem do gestor para enviar o produto ao cidadão. Além disso, esta farmácia também reestabelece o seu stock e a sua receita e envia essa informação ao gestor. Por fim, se a mensagem tiver a *performative* de *INFORM_IF*, então significa que a farmácia que inicialmente não tinha quantidade suficiente para o pedido do cidadão, recebeu o lote que foi enviado pelo fornecedor para que possa reestabelecer o seu próprio stock para 9 no respetivo medicamento, que é o máximo que cada farmácia tem para cada medicamento que possui.

6.1.3 Agente Gestor

No caso do agente gestor existem, tal como na farmácia, 3 classes. A primeira designa-se por classe Gestor, na qual estão presentes as variáveis como um *HashMap* chamado farm_gestor, em que cada chave corresponde a uma string, nome da Farmácia, e um valor que corresponde a uma instância da classe Farm, onde estão presentes características de cada farmácia como: stock de medicamentos, preço dos seus medicamentos, produtos vendidos, receita, despesa, a sua posição e o seu nome de agente. Além disso, também está presente um *HashMap* designado por forn_gestor, onde cada chave corresponde a uma string, nome do fornecedor, e um valor que corresponde a uma instância da classe Forn, onde estão presentes características de cada fornecedor como: o preço de cada medicamento que vende e o seu nome de agente. A segunda classe corresponde a um *behaviour* do tipo *OneShotBehaviour*, em que o gestor procura nas páginas amarelas cada farmácia e envia-lhe um pedido para receber as suas informações para poder colocar no *HashMap* farm_gestor. Também, ainda, neste *behaviour* o gestor contacta cada fornecedor através das respetivas páginas amarelas, para receber as suas informações e colocar no *HashMap* forn_gestor.

A última classe é um *behaviour* do tipo *CyclicBehaviour*, responsável, pelo facto, do gestor estar constantemente a receber mensagens. Se a mensagem tiver a *performative* do tipo *SUBSCRIBE* significa que recebeu mensagens de uma farmácia para se registar e, portanto,

coloca essas informações no seu *HashMap* farm_gestor. Por outro lado, se a mensagem tiver a *performative* do tipo *CFP* significa que recebeu mensagens de um fornecedor para se registar e, dessa forma, coloca essas informações no seu *HashMap* forn_gestor. Se a mensagem tiver a *performative* do tipo *REQUEST*, então o gestor está a receber informação de uma farmácia, constatando que esta não possui stock suficiente para tratar do pedido de um cidadão. Dessa forma, o gestor calcula a segunda farmácia mais próxima do cidadão e avisa-a, através de uma mensagem com uma *performative* do tipo *INFORM*, que necessita de enviar o respetivo pedido ao cidadão em questão. Ainda nesta *performative*, o gestor calcula o fornecedor mais barato para o respetivo medicamento em causa e manda-lhe uma mensagem do tipo *INFORM* a dizer que necessita de reestabelecer o stock para 9 na primeira farmácia.

Além disto, se o gestor receber uma mensagem do tipo *INFORM_REF* significa que recebeu informação de uma farmácia a dizer que enviou o pedido ao cidadão e, assim, o gestor atualiza o seu stock, atualiza a sua receita e atualiza os seus produtos vendidos. De seguida, verifica se a quantidade atual de stock é inferior ou não ao valor de *threshold* (2) e se não for, então envia uma mensagem do tipo *INFORM* ao agente interface para imprimir resultados. Caso seja, então calcula o fornecedor mais barato para o respetivo medicamento e envia-lhe uma mensagem do tipo *INFORM* para reestabelecer o stock a essa mesma farmácia.

Por fim, se o gestor receber uma mensagem com a *performative* do tipo *INFORM*, tal significa que recebeu informação de que uma farmácia reestabeleceu o seu stock graças ao fornecedor e, por esse motivo, o gestor atualiza o seu stock, atualiza a sua despesa e envia mensagem à interface do tipo *INFORM* a dizer para imprimir resultados.

6.1.4 Agente Fornecedor

No agente fornecedor existem duas classes. A classe fornecedor, onde está presente a variável med_prec, em que cada chave é uma string que corresponde a um nome de um medicamento e o seu valor é o preço corresponde a esse mesmo medicamento. No seu setup(), o agente fornecedor regista-se nas suas páginas amarelas.

Para além disto, também existe uma classe que corresponde a um *behaviour* do tipo *CyclicBehaviour*, responsável pelo fornecedor estar constantemente a receber mensagens. Se a mensagem tiver a *performative* do tipo *CFP*, significa que recebeu uma mensagem do gestor a pedir que lhe envie as suas informações e, portanto, o fornecedor envia um objeto do tipo Forn, que corresponde a uma instância da classe Forn, onde estão presentes as características de cada fornecedor como: o preço dos medicamentos que vende e o seu nome de agente. Para tal, então envia uma mensagem ao gestor do tipo *CFP*.

Além disso, se a mensagem for do tipo *INFORM*, então significa que o fornecedor recebeu um pedido do gestor a constatar que necessita de reestabelecer o stock para uma determinada farmácia.

6.1.5 Agente Interface

No agente interface, que é responsável por imprimir resultados, existem três classes. A primeira classe designada por Interface apresenta três variáveis. Uma do tipo *HashMap* chamado Farmacias, em que cada chave é uma string com o nome de cada farmácia e o valor corresponde à respetiva instância da farmácia da classe Farm. As restantes duas variáveis são do tipo string: "farmaciaComMaisPedidos"e "farmacia_maior_lucro". No seu setup(), estão presentes os seus *behaviours*: ContactarFarmacias() e Receiver().

A segunda classe (Contactar_Farmacias()) corresponde a um *behaviour* do tipo *OneShotBehaviour*, no qual o agente interface contacta cada farmácia presente nas páginas amarelas e envia-lhes uma mensagem do tipo *PROPOSE*, de forma a poder receber as suas informações.

A terceira classe, designada por Receiver corresponde a um *behaviour* do tipo *CyclicBehaviour*, o qual é responsável pela interface estar constantemente a receber mensagens. Se a mensagem tiver a *performative* do tipo *SUBSCRIBE* significa que a farmácia enviou as suas informações e, portanto, o agente interface coloca no HashMap Farmacias essas informações.

Por fim, se a mensagem tiver uma *performative* do tipo *INFORM*, então significa que chegou o momento do agente interface imprimir os resultados. Para tal recebe um *HashMap* do gestor como objeto contido na mensagem, que contém todas as informações atualizadas de cada farmácia. Desta forma, o agente interface percorre o *HashMap* e mostra 4 gráficos: um que contém o stock atual de cada farmácia, outro onde mostra os produtos vendidos e total de vendas de cada farmácia, outro onde se encontra o lucro por farmácia e outro onde se encontra o total de pedidos em cada farmácia. Além disto, também aparecem, na consola, duas frases a mostrar qual a farmácia que apresenta maior lucro e qual a farmácia que apresenta maior número de pedidos.

6.1.6 Classe Farmácia

A classe Farmácia, designada por "Class Forn", é a classe onde se encontram as informações relevantes de cada farmácia. Nesta classe está presente qual o agente respetivo que representa uma farmácia, o seu stock de medicamento, tal como o preço por cada medicamento e o número de produtos vendidos por cada medicamento. Além disso, tem-se a posição (x e y),

receita e despesa para a farmácia. O seu construtor é definido pelo agente, pela posição e, além disto, também nesta classe estão presentes os métodos get() e set() de cada variável.

6.1.7 Classe Fornecedor

A classe Fornecedor, designada por "Class Farm", é a classe onde se encontram as informações relevantes de cada fornecedor. O seu construtor é definido pelo agente que representa cada fornecedor e, também nesta classe está presente a variável "med_preço"que é um *HashMap*, no qual cada chave é uma string que corresponde ao nome de cada medicamento e o seu valor corresponde ao respetivo preço. Além disso, estão presentes os métodos get() e set() para a variável "AID agent" e o *HashMap* "med_preço".

6.1.8 Classe Posição

A classe Posição, designada por "Class Position", serve para fornecer uma coordenada x e y a cada cidadão e a cada farmácia. O seu construtor é definido por um valor de x e y, que são variáveis do tipo "int"e, também nesta classe estão presentes os métodos get() e set() para as variáveis x e y.

6.2 Demonstrações

Nesta secção serão demonstrados os outputs da consola, bem como alguns resultados estatísticos que o Sistema Multiagente, nomeadamente o agente interface, imprime.

Primeiramente, foram inicializadas seis farmácias com um intervalo de tempo entre elas de 500 milissegundos, bem como dois fornecedores com um intervalo de tempo entre eles de 500 milissegundos. De seguida, inicializou-se o gestor e 500 milissegundos depois foi inicializado o agente interface. Além disso, quando o agente gestor é inicializado, este procura nas páginas amarelas e contacta cada farmácia e cada fornecedor para que estes lhe possam enviar as suas coordenadas e informações. O agente interface também procura nas páginas amarelas e contacta cada farmácia para que lhe possam enviar as suas coordenadas - **figura 12**.

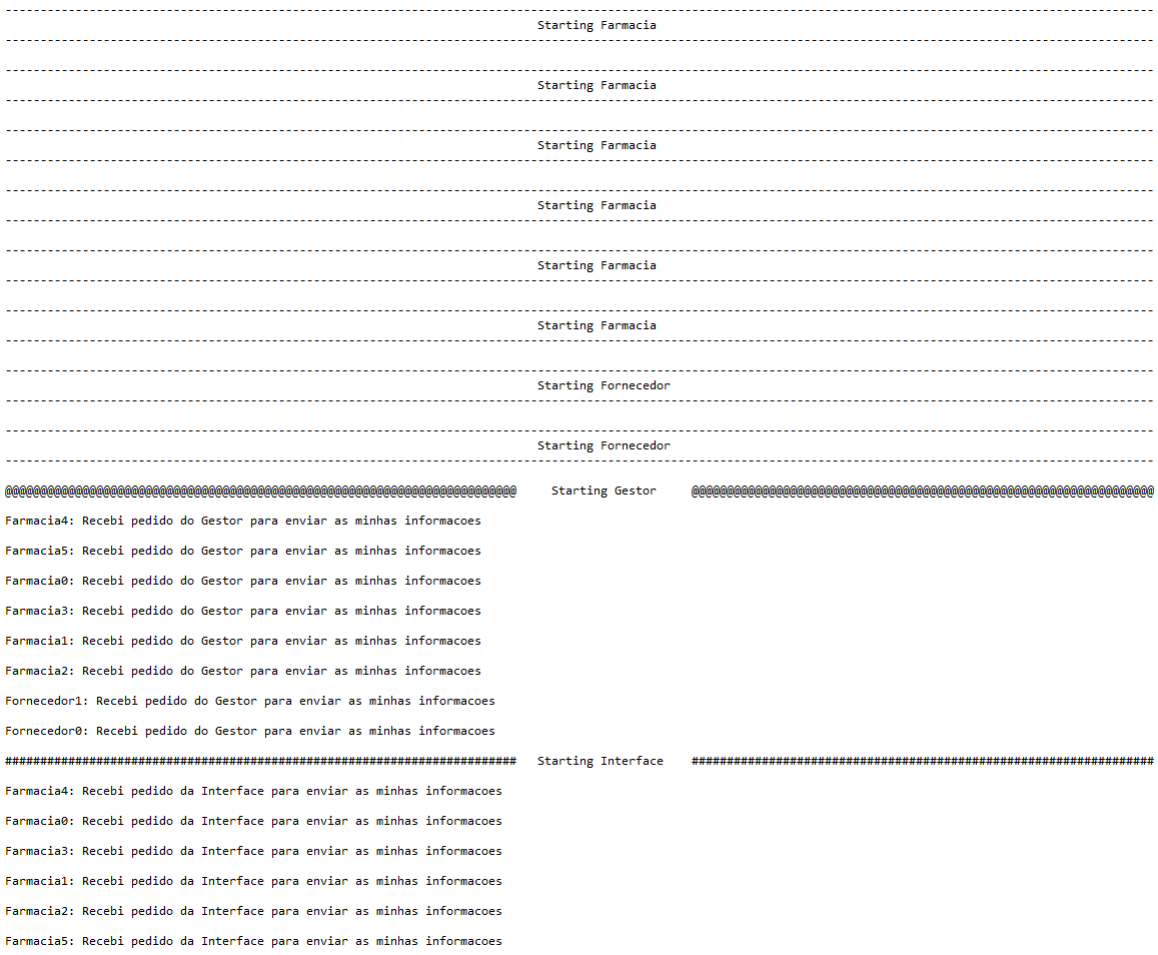

Figura 12: Inicialização dos agentes

Quando um agente cidadão se inicializa, **figura 13**, este mostra as suas informações como: nome, NIF, data de nascimento, morada e código postal. Depois pede e recebe as coordenadas de cada farmácia para poder calcular a farmácia que está mais próxima.

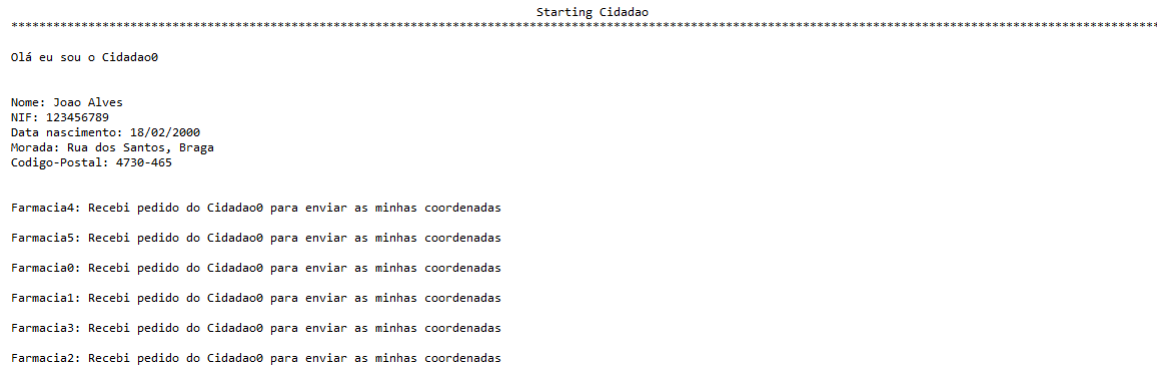

Figura 13: Inicialização do agente cidadão e pedido das coordenadas das farmácias

De seguida, o cidadão escolhe a farmácia mais próxima e envia o seu pedido. Se a farmácia possui o stock necessário para o pedido do cidadão, confirma e envia esse pedido ao cidadão e também envia a informação de que o agente gestor necessita de atualizar o seu stock. Esta situação está presente na figura abaixo.

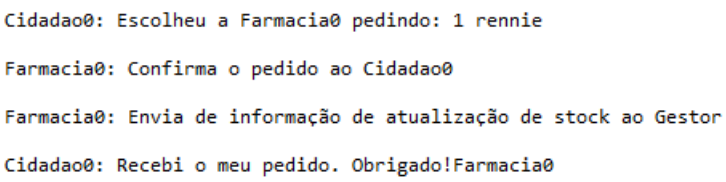

Figura 14: Escolha da farmácia mais próxima, envio e recebimento do pedido

Se, eventualmente, a farmácia escolhida pelo cidadão não possuir stock suficiente para responder ao pedido do cidadão, então pede ao gestor para calcular a segunda farmácia mais próxima do cidadão e envia também o pedido do cidadão para que o gestor possa enviar esse pedido à segunda farmácia - **figura 15**.

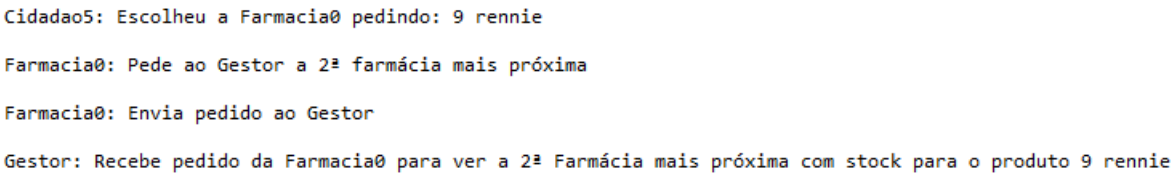

Figura 15: 1ª farmácia pede ao gestor para calcular a 2ª farmácia mais próxima

Depois de o gestor calcular a 2ª farmácia mais próxima, envia o pedido do cidadão a essa farmácia para essa poder enviar esse mesmo pedido ao cidadão correspondente - **figura 16**.

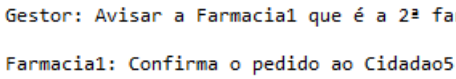

Figura 16: 2ª farmácia recebe o pedido e envia ao cidadão

De seguida, na figura **17**, o gestor, no momento em que envia a informação à 2ª farmácia mais próxima, calcula qual o fornecedor mais barato para o medicamento em questão e envia mensagem a esse fornecedor para que este reponha stock à primeira farmácia. Essa farmácia pede ao gestor que atualize o seu stock e que diga ao agente interface para imprimir resultados.

```
Fornecedor1: Recebe pedido do GESTOR para reestabelecer o stock na Farmacia0 para o produto rennie

Farmacia0: Recebi o lote que precisava. Obrigado! Fornecedor1

Farmacia0: Envia de informação de atualização de stock ao Gestor

Gestor: Avisar a Interface para imprimir resultados!
```

Figura 17: Reestabelecimento de stock na primeira farmácia

Quando a 2ª farmácia envia o pedido ao cidadão, esta pede ao gestor que atualize o seu stock e que diga ao agente interface para imprimir resultados. O cidadão acaba por confirmar que recebeu o pedido. Além disso, uma vez que neste caso o valor de stock naquele momento da 2ª farmácia era inferior ao valor de threshold - 2 - então envia uma mensagem ao gestor a avisar que necessita de repor o stock para o medicamento requisitado pelo cidadão. Após receber o lote, pede ao gestor que volte a atualizar o seu stock e que diga ao agente interface para imprimir resultados - **figura 18**.

```
Farmacia1: Envia de informação de atualização de stock ao Gestor

Cidadao5: Recebi o meu pedido. Obrigado!Farmacia1

Fornecedor1: Recebe pedido do GESTOR para reestabelecer o stock na Farmacia1 para o produto rennie

Farmacia1: Recebi o lote que precisava. Obrigado! Fornecedor1

Farmacia1: Envia de informação de atualização de stock ao Gestor

Gestor: Avisar a Interface para imprimir resultados!
```

Figura 18: Atualização de stock na 2ª farmácia e envio à interface

Quando o agente interface imprime resultados, são mostradas várias tabelas. A primeira, **figura 19**, corresponde ao stock das farmácias para todos os medicamentos.

STOCK DAS FARMÁCIAS

FARMÁCIA	Brufen	Ben-u-ron	Aspirina	Xanax	Valium	Fenistil	Voltaren	Buscopan	Leite NAN	Kompensan	Rennie	Bissolvon	Strepfen
FARMACIA4	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA5	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA0	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA1	9	9	9	9	9	9	9	9	9	9	0	9	9
FARMACIA2	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA3	9	9	9	9	9	9	9	9	9	9	9	9	9

Figura 19: Stock das farmácias

A segunda, **figura 20**, corresponde ao número de produtos vendidos para cada medicamento em cada farmácia, bem como o total de vendas.

Nº DE VENDAS

FARMÁCIA	Brufen	Ben-u-ron	Aspirina	Xanax	Valium	Fenistil	Voltaren	Buscopan	Leite NAN	Kompensan	Rennie	Bissolvon	Strepfen	Total de vendas
FARMACIA4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FARMACIA5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FARMACIA0	0	0	0	0	0	0	0	0	0	0	9	0	0	9
FARMACIA1	0	0	0	0	0	0	0	0	0	0	18	0	0	18
FARMACIA2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FARMACIA3	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 20: Número de produtos vendidos

A terceira tabela, **figura 21**, corresponde ao histórico de cada paciente para cada farmácia, mostrando quantos medicamentos foram vendidos a cada paciente em cada farmácia.

Histórico de Pacientes por Farmácia

FARMÁCIA	Cidadao0	Cidadao1	Cidadao2	Cidadao3	Cidadao4	Cidadao5	Cidadao6	Cidadao7	Cidadao8	Cidadao9
FARMACIA4	0	0	0	0	0	0	0	0	0	0
FARMACIA5	0	0	0	0	0	0	0	0	0	0
FARMACIA0	1	1	2	0	5	0	0	0	0	0
FARMACIA1	0	0	0	9	0	9	0	0	0	0
FARMACIA2	0	0	0	0	0	0	0	0	0	0
FARMACIA3	0	0	0	0	0	0	0	0	0	0

Figura 21: Histórico de pacientes por farmácia

A última tabela na consola, representada na **figura 22**, é aquela que apresenta o lucro e o número de pedidos em cada farmácia. Além disso, mostra duas frases, evidenciando qual a farmácia com maior lucro e farmácia que teve mais pedidos, ou seja, que foi mais requisitada pelos cidadãos.

FARMÁCIA	Lucro	Pedidos
FARMACIA4	0	0
FARMACIA5	0	0
FARMACIA0	72	4
FARMACIA1	207	2
FARMACIA2	0	0
FARMACIA3	0	0

A farmacia com maior lucro é: Farmacia1 com 207 euros!
A farmacia com maior numero de pedidos é: Farmacia0 com 4 pedido(s)!

Figura 22: Lucro e pedidos por farmácia

Antes de apresentar uma demonstração completa, mostrando todas as mensagens de um exemplo, é necessário salientar que a situação mais demorada do nosso Sistema Multiagente é quando a primeira farmácia não possui stock suficiente e, por isso, uma segunda farmácia é localizada pelo gestor para enviar o pedido ao cidadão. Nesta situação, a primeira farmácia receberá um lote para reestabelecer stock de um fornecedor e a segunda pelo facto de enviar o pedido ao cidadão e, no caso de ficar com um stock inferior ao valor de threshold - 2, então também necessitará de receber um lote pelo fornecedor mais barato para reestabelecer, também, o seu stock. Para tal, e só para efeitos de demonstração, forçou-se a que o pedido seja feito sempre à mesma farmácia e que seja pedido sempre o mesmo medicamento.

Então, no exemplo da figura abaixo, a situação mais demorada aconteceu com o cidadão 2. Para contextualizar o exemplo, antes do cidadão 2, o cidadão 0 pediu 2 medicamentos "Rennie" à farmácia 0 e recebeu-os, ficando a farmácia 0 com 7 medicamentos "Rennie". De seguida, o cidadão 1 pediu 2 medicamentos "Rennie", e recebeu-os, ficando esta farmácia com 5 "Rennie"no seu stock.

O cidadão 2 pede 8 medicamentos "Rennie" à farmácia 0, contudo como só existem 5, então esta farmácia pede ao gestor para calcular a 2ª farmácia mais próxima, com stock, que corresponde à farmácia 4. De seguida, esta farmácia confirma o pedido ao cidadão. O fornecedor mais barato para, neste caso, o medicamento "Rennie", é chamado pelo gestor para que reestabeleça o stock à farmácia 0. O cidadão 2 recebe o seu pedido e a farmácia 4, como fica apenas com 1 medicamento "Rennie", que é abaixo do threshold, pede ao gestor que atualize o seu stock. Ao mesmo tempo, a farmácia 0 acaba por receber o lote que precisava do fornecedor mais barato (fornecedor 1) e pede ao gestor que atualize o seu stock. O gestor pede à interface que imprima estes resultados. Na mesma ordem de ideias, o gestor pede também ao fornecedor 1 (neste caso, o mais barato para o medicamento pretendido) que reestabeleça o stock a essa farmácia. A farmácia 4 também recebe o lote que precisa e volta a mandar mensagem ao gestor que atualize o seu stock. Por fim, o gestor pede à interface que volte a imprimir resultados.

```
                                                                                    Starting Cidadao
************************************************************************************************************************************************************************

Olá eu sou o Cidadao2


Nome: Rodrigo Cardoso
NIF: 222226789
Data nascimento: 18/09/1998
Morada: Rua dos Anjos, Coimbra
Codigo-Postal: 4900-333


Farmacia4: Recebi pedido do Cidadao2 para enviar as minhas coordenadas

Farmacia5: Recebi pedido do Cidadao2 para enviar as minhas coordenadas

Farmacia0: Recebi pedido do Cidadao2 para enviar as minhas coordenadas

Farmacia3: Recebi pedido do Cidadao2 para enviar as minhas coordenadas

Farmacia1: Recebi pedido do Cidadao2 para enviar as minhas coordenadas

Farmacia2: Recebi pedido do Cidadao2 para enviar as minhas coordenadas

Cidadao2: Escolheu a Farmacia0 pedindo: 8 rennie

Farmacia0: Pede ao Gestor a 2ª farmácia mais próxima

Farmacia0: Envia pedido ao Gestor

Gestor: Recebe pedido da Farmacia0 para ver a 2ª Farmácia mais próxima com stock para o produto 8 rennie

Gestor: Avisar a Farmacia4 que é a 2ª farmácia mais próxima

Farmacia4: Confirma o pedido ao Cidadao2


Fornecedor1: Recebe pedido do GESTOR para reestabelecer o stock na Farmacia0 para o produto rennie

Farmacia4: Envia de informação de atualização de stock ao Gestor

Cidadao2: Recebi o meu pedido. Obrigado!Farmacia4

Farmacia0: Recebi o lote que precisava. Obrigado! Fornecedor1

Farmacia0: Envia de informação de atualização de stock ao Gestor

Fornecedor1: Recebe pedido do GESTOR para reestabelecer o stock na Farmacia4 para o produto rennie

Gestor: Avisar a Interface para imprimir resultados!
```

Figura 23: Envio e recebimento de mensagens no sistema multiagente

Quando o agente interface imprime resultados, são mostrados várias tabelas. A primeira corresponde ao stock das farmácias para todos os medicamentos e a segunda corresponde ao número de vendas por farmácia. Estes resultados correspondem quando a primeira farmácia pede ao gestor que atualize o seu stock e que envie à interface para imprimir resultados

```
************************************************************************************************************************************************************************
                                                                                    STOCK DAS FARMÁCIAS
************************************************************************************************************************************************************************
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+
| FARMÁCIA    | Brufen  | Ben-u-ron   | Aspirina | Xanax   | Valium  | Fenistil  | Voltaren | Buscopan  | Leite NAN   | Kompensan | Rennie    | Bissolvon | Strepfen |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+
| FARMACIA4   |    9    |      9      |    9     |    9    |    9    |     9     |    9     |     9     |      9      |    9    |     1     |    9     |    9     |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+
| FARMACIA5   |    9    |      9      |    9     |    9    |    9    |     9     |    9     |     9     |      9      |    9    |     9     |    9     |    9     |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+
| FARMACIA0   |    9    |      9      |    9     |    9    |    9    |     9     |    9     |     9     |      9      |    9    |     9     |    9     |    9     |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+
| FARMACIA1   |    9    |      9      |    9     |    9    |    9    |     9     |    9     |     9     |      9      |    9    |     9     |    9     |    9     |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+
| FARMACIA2   |    9    |      9      |    9     |    9    |    9    |     9     |    9     |     9     |      9      |    9    |     9     |    9     |    9     |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+
| FARMACIA3   |    9    |      9      |    9     |    9    |    9    |     9     |    9     |     9     |      9      |    9    |     9     |    9     |    9     |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+


************************************************************************************************************************************************************************
                                                                                    Nº DE VENDAS
************************************************************************************************************************************************************************
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
| FARMÁCIA    | Brufen  | Ben-u-ron   | Aspirina | Xanax   | Valium  | Fenistil  | Voltaren | Buscopan  | Leite NAN   | Kompensan | Rennie    | Bissolvon | Strepfen |   Total de vendas   |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
| FARMACIA4   |    0    |      0      |    0     |    0    |    0    |     0     |    0     |     0     |      0      |    0    |     8     |    0     |    0     |        8        |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
| FARMACIA5   |    0    |      0      |    0     |    0    |    0    |     0     |    0     |     0     |      0      |    0    |     0     |    0     |    0     |        0        |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
| FARMACIA0   |    0    |      0      |    0     |    0    |    0    |     0     |    0     |     0     |      0      |    0    |     4     |    0     |    0     |        4        |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
| FARMACIA1   |    0    |      0      |    0     |    0    |    0    |     0     |    0     |     0     |      0      |    0    |     0     |    0     |    0     |        0        |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
| FARMACIA2   |    0    |      0      |    0     |    0    |    0    |     0     |    0     |     0     |      0      |    0    |     0     |    0     |    0     |        0        |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
| FARMACIA3   |    0    |      0      |    0     |    0    |    0    |     0     |    0     |     0     |      0      |    0    |     0     |    0     |    0     |        0        |
+-------------+---------+-------------+----------+---------+---------+-----------+----------+-----------+-------------+---------+-----------+----------+----------+-----------------+
```

Figura 24: Stock das farmácias e número de vendas por farmácia

A terceira e quarta tabelas dizem respeito ao histórico de pacientes e ao lucro/pedidos por farmácia, respetivamente. Também se apresenta qual a farmácia com maior lucro e com mais pedidos, ou seja, a que foi mais requisitada pelos cidadãos.

Histórico de Pacientes por Farmácia

FARMÁCIA	Cidadao0	Cidadao1	Cidadao2	Cidadao3	Cidadao4	Cidadao5	Cidadao6	Cidadao7	Cidadao8	Cidadao9
FARMACIA4	0	0	8	0	0	0	0	0	0	0
FARMACIA5	0	0	0	0	0	0	0	0	0	0
FARMACIA0	2	2	0	0	0	0	0	0	0	0
FARMACIA1	0	0	0	0	0	0	0	0	0	0
FARMACIA2	0	0	0	0	0	0	0	0	0	0
FARMACIA3	0	0	0	0	0	0	0	0	0	0

FARMÁCIA	Lucro	Pedidos
FARMACIA4	120	1
FARMACIA5	0	0
FARMACIA0	32	2
FARMACIA1	0	0
FARMACIA2	0	0
FARMACIA3	0	0

```
A farmacia com maior lucro é: Farmacia4 com 120 euros!
A farmacia com maior numero de pedidos é: Farmacia0 com 2 pedido(s)!
Farmacia4: Recebi o lote que precisava. Obrigado! Fornecedor1

Farmacia4: Envia de informação de atualização de stock ao Gestor

Gestor: Avisar a Interface para imprimir resultados!
```

Figura 25: Histórico de pacientes e lucro/pedidos por farmácia

De seguida, a interface volta a imprimir resultados pelo facto da segunda farmácia também receber o lote que necessitava para reestabelecer stock, pois apresentava um stock inferior ao valor de threshold após enviar o pedido ao cidadão.

STOCK DAS FARMÁCIAS

FARMÁCIA	Brufen	Ben-u-ron	Aspirina	Xanax	Valium	Fenistil	Voltaren	Buscopan	Leite NAN	Kompensan	Rennie	Bissolvon	Strepfen
FARMACIA4	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA5	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA0	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA1	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA2	9	9	9	9	9	9	9	9	9	9	9	9	9
FARMACIA3	9	9	9	9	9	9	9	9	9	9	9	9	9

Nº DE VENDAS

FARMÁCIA	Brufen	Ben-u-ron	Aspirina	Xanax	Valium	Fenistil	Voltaren	Buscopan	Leite NAN	Kompensan	Rennie	Bissolvon	Strepfen	Total de vendas
FARMACIA4	0	0	0	0	0	0	0	0	0	0	8	0	0	8
FARMACIA5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FARMACIA0	0	0	0	0	0	0	0	0	0	0	4	0	0	4
FARMACIA1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FARMACIA2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FARMACIA3	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 26: Stock das farmácias e número de vendas por farmácia

```
**********************************************************************************************************************************
                                                Histórico de Pacientes por Farmácia
**********************************************************************************************************************************
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  FARMÁCIA |  Cidadao0 | Cidadao1  |  Cidadao2 |  Cidadao3 |  Cidadao4 | Cidadao5  | Cidadao6  |  Cidadao7 | Cidadao8  | Cidadao9  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| FARMACIA4 |     0     |     0     |     8     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| FARMACIA5 |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| FARMACIA0 |     2     |     2     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| FARMACIA1 |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| FARMACIA2 |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| FARMACIA3 |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |     0     |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+


| FARMÁCIA  |   Lucro   |  Pedidos  |
+-----------+-----------+-----------+
| FARMACIA4 |    64     |     1     |
| FARMACIA5 |     0     |     0     |
| FARMACIA0 |    32     |     2     |
| FARMACIA1 |     0     |     0     |
| FARMACIA2 |     0     |     0     |
| FARMACIA3 |     0     |     0     |
+-----------+-----------+-----------+


A farmacia com maior lucro é: Farmacia4 com 64 euros!
A farmacia com maior numero de pedidos é: Farmacia0 com 2 pedido(s)!
```

Figura 27: Histórico de pacientes e lucro/pedidos por farmácia

Além destas tabelas que são impressas na consola, também se recorreu ao software "Jfreechart"para elaborar gráficos de barras e gráficos circulares. Os gráficos que são apresentados são: número de vendas por farmácia (**figura 28**), lucro por farmácia (**figura 29**) e, por fim, qual foi o cidadão que mais pedidos fez em cada uma das farmácias (**figura 30**).

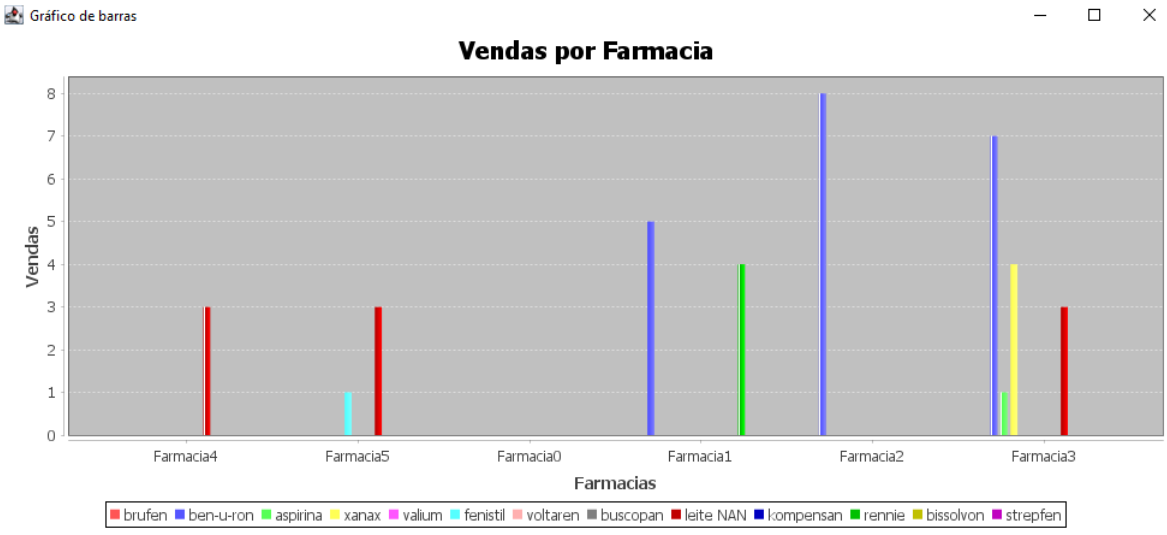

Figura 28: Número de vendas por farmácia

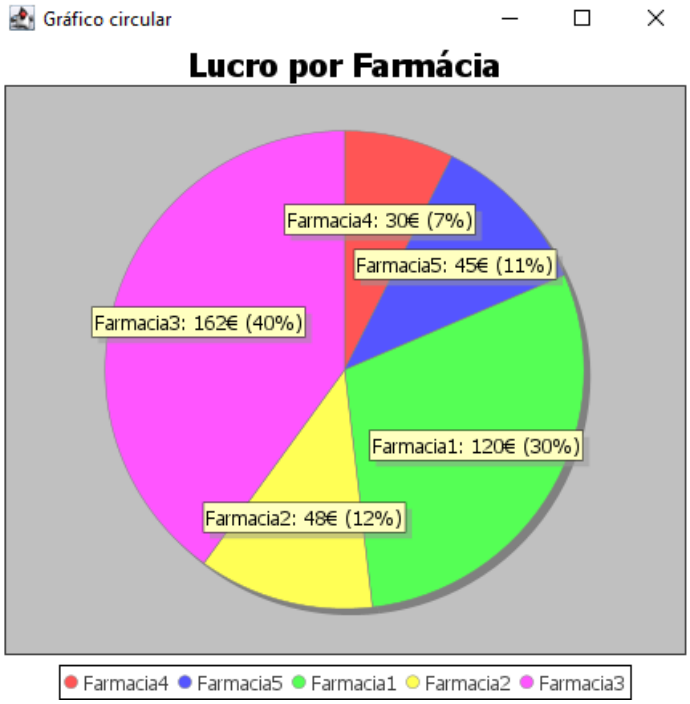

Figura 29: Lucro por farmácia

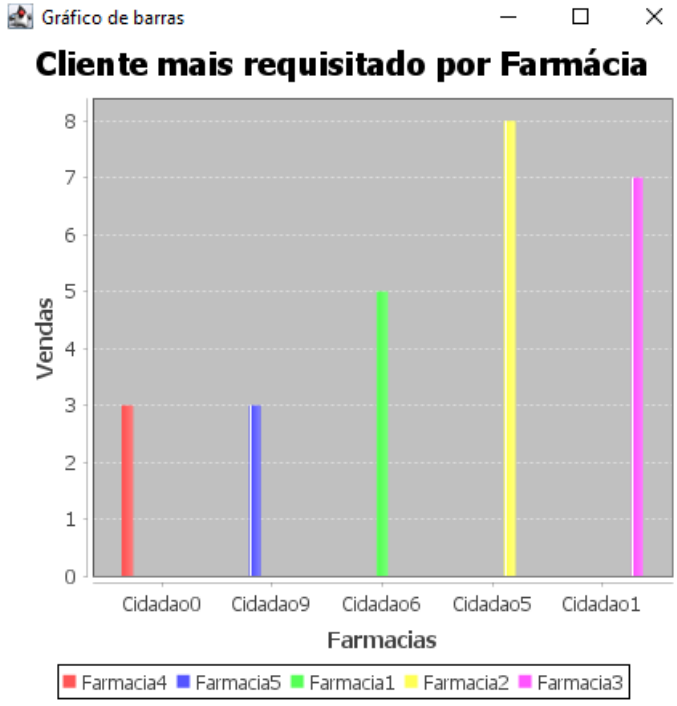

Figura 30: Cliente mais requisitado em cada farmácia

7 Conclusão

O presente trabalho prende-se com o desenvolvimento de um Sistema Multiagente que, de forma automática, tem a capacidade de gerir a gestão e a manutenção de farmácias, permitindo, assim, uma constante atualização de stocks das correspondentes farmácias e também uma visualização de informações relativas ao lucro, stock e vendas dessas mesmas. O SMA implementado permite uma melhoria na gestão de recursos e simplifica o processo de compra de medicamentos e produtos de uma farmácia.

Como referido ao longo do relatório, com este SMA, o cidadão, de uma forma simples, consegue adquirir medicamentos das farmácias mais próximas recorrendo ao cálculo da distância entre a sua posição e a localização das mesmas. Permite também, uma gestão eficiente do stock nas farmácias, prevendo os stocks necessários e pedindo, se necessário, encomendas a fornecedores. O sistema é gerido por um agente gestor que é responsável por manter os stocks das farmácias, calculando para esse efeito quando é que é necessário comprar mais medicamentos. Para tal, este escolhe qual o fornecedor mais barato para levar o medicamento para a farmácia em questão. Para destacar o desempenho das farmácias o agente gestor envia os dados para o agente interface, que imprime os mesmos, de modo que este possa verificar o lucro, as vendas e o stock de cada farmácia ao longo do tempo.

É importante referir, que apesar de dificuldades encontradas durante a fase transição do planeamento para implementação, todas foram superadas, já que se conseguiu efetuar um Sistema Multiagente que corresponde às expectativas iniciais do projeto.

Também de salientar, que para trabalho futuro, poder-se-ia, eventualmente, melhorar ainda mais o SMA. Para tal, seria necessário, por exemplo, associar um tipo de urgência a cada cidadão e, assim, a farmácia ia respondendo aos pedidos de maior para menor urgência. Também se poderia colocar outra funcionalidade, nomeadamente a possibilidade do cidadão conseguir pedir mais do que um medicamento a uma determinada farmácia.

Referências

[1] Ordem dos Farmacêuticos. "Industria Farmacêutica". Acedido em: 26 abril, 2021. [Online]. Available: https://www.ordemfarmaceuticos.pt/pt/areas-profissionais/industria-farmaceutica/

[2] FABERNOVEL. "As inovações que estão a impactar a Indústria Farmacêutica". Acedido em: 26 abril, 2021. [Online]. Available: https://supertoast.pt/2016/09/29/o-futuro-da-industria-farmaceutica/

[3] Farmácias Portuguesas. "Conheça a nova App das Farmácias Portuguesas". Acedido em: 26 abril, 2021. [Online]. Available: https://www.farmaciasportuguesas.pt/app

[4] FARMÁCIA ENTREGA. "A Farmácia Online que entrega onde estiver". Acedido em: 26 abril, 2021. [Online]. Available: https://farmaciaentrega.pt/

[5] JADE. "JAVA Agent DEvelopment Framework". Acedido em: 23 maio, 2021. [Online]. Available: https://jade.tilab.com/

[6] G. Caire, "Jade tutorial: Jade programming for beginners," 2009. [Online]. Available: https://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf

[7] JADE. "Tutorial 1: JADE Architecture Overview". Acedido em: 23 maio, 2021. [Online]. Available: https://jade.tilab.com/doc/tutorials/JADEAdmin/jadeArchitecture.html

Anexos

A. Main Container

```
1  import jade.core.Profile;
2  import jade.core.ProfileImpl;
3  import jade.core.Runtime;
4  import jade.wrapper.AgentController;
5  import jade.wrapper.ContainerController;
6
7  public class MainContainer {
8
9    Runtime rt;
10   ContainerController container;
11
12   public static void main(String[] args) {
13     MainContainer a = new MainContainer();
14
15     a.initMainContainerInPlatform("localhost", "9888", "MainContainer");
16
17     int n;
18     int limit_farmacias = 6;
19     int limit_cidadaos = 10;
20     int limit_fornecedores = 2;
21
22     // Start Agents Farmacias!
23     for (n = 0; n < limit_farmacias; n++) {
24       try {
25         a.startAgentInPlatform("Farmacia" + n, "Agents.Farmacia");
26         Thread.sleep(500);
27
28       } catch (InterruptedException e) {
29         e.printStackTrace();
30       }
31     }
32
33       try {
34         Thread.sleep(500);
35       } catch (InterruptedException e1) {
36         // TODO Auto-generated catch block
37         e1.printStackTrace();
38       }
39
40     // Start Agents Fornecedores!
41     for (n = 0; n < limit_fornecedores; n++) {
42       try {
43         a.startAgentInPlatform("Fornecedor" + n, "Agents.Fornecedor");
44         Thread.sleep(500);
```

```
45
46        } catch (InterruptedException e) {
47          e.printStackTrace();
48        }
49      }
50
51        try {
52          Thread.sleep(500);
53        } catch (InterruptedException e1) {
54          // TODO Auto-generated catch block
55          e1.printStackTrace();
56        }
57
58      try {
59        a.startAgentInPlatform("Gestor", "Agents.Gestor");
60        Thread.sleep(500);
61      } catch (InterruptedException e1) {
62        // TODO Auto-generated catch block
63        e1.printStackTrace();
64      }
65
66      try {
67        a.startAgentInPlatform("Interface", "Agents.Interface");
68        Thread.sleep(500);
69      }
70        catch (InterruptedException e1) {
71          // TODO Auto-generated catch block
72          e1.printStackTrace();
73        }
74
75      // Start Agents Cidadaos!
76      for (n = 0; n < limit_cidadaos; n++) {
77        try {
78          a.startAgentInPlatform("Cidadao" + n, "Agents.Cidadao");
79          Thread.sleep(5000);
80        }
81        catch (InterruptedException e) {
82        e.printStackTrace();
83        }
84      }
85    }
86
87    public ContainerController initContainerInPlatform(String host, String port, String
        containerName) {
88      // Get the JADE runtime interface (singleton)
89      this.rt = Runtime.instance();
90
91
```

```
92      // Create a Profile, where the launch arguments are stored
93      Profile profile = new ProfileImpl();
94      profile.setParameter(Profile.CONTAINER_NAME, containerName);
95      profile.setParameter(Profile.MAIN_HOST, host);
96      profile.setParameter(Profile.MAIN_PORT, port);
97      // create a non-main agent container
98      ContainerController container = rt.createAgentContainer(profile);
99      return container;
100   }
101
102   public void initMainContainerInPlatform(String host, String port, String
          containerName) {
103
104     // Get the JADE runtime interface (singleton)
105     this.rt = Runtime.instance();
106
107     // Create a Profile, where the launch arguments are stored
108     Profile prof = new ProfileImpl();
109     prof.setParameter(Profile.CONTAINER_NAME, containerName);
110     prof.setParameter(Profile.MAIN_HOST, host);
111     prof.setParameter(Profile.MAIN_PORT, port);
112     prof.setParameter(Profile.MAIN, "true");
113     prof.setParameter(Profile.GUI, "true");
114
115     // create a main agent container
116     this.container = rt.createMainContainer(prof);
117     rt.setCloseVM(true);
118   }
119
120   public void startAgentInPlatform(String name, String classpath) {
121     try {
122       AgentController ac = container.createNewAgent(name, classpath, new Object[0]);
123       ac.start();
124     } catch (Exception e) {
125       e.printStackTrace();
126     }
127   }
128 }
```

B. Agente Cidadao

```
129 package Agents;
130
131 import jade.core.Agent;
132 import java.util.ArrayList;
133 import java.util.Arrays;
134 import java.util.HashMap;
```

```
135 import java.util.Calendar;
136 import java.util.Random;
137
138 import Classes.Position;
139 import jade.core.AID;
140 import jade.core.Agent;
141 import jade.core.behaviours.CyclicBehaviour;
142 import jade.core.behaviours.OneShotBehaviour;
143 import jade.domain.DFService;
144 import jade.domain.FIPAException;
145 import jade.domain.FIPAAgentManagement.DFAgentDescription;
146 import jade.domain.FIPAAgentManagement.ServiceDescription;
147 import jade.lang.acl.ACLMessage;
148
149 public class Cidadao extends Agent {
150   private String name;
151   int xCidadao, yCidadao;
152   int quantidade;
153   int numFarmacias;
154   private ArrayList<String> products;
155   private String produtoRequisitado;
156   private String nome;
157   private String nif;
158   private String data_nascimento;
159   private String morada;
160   private String cod_postal;
161
162   protected void setup() {
163     super.setup();
164
165     System.out.print(
166         "************************************************************\n");
167     System.out.print(
168           "                    Starting Cidadao                      \n");
169     System.out.print(
170         "************************************************************\n");
171     System.out.print("\n");
172
173     // As coordenadas do cidadao sao escolhidas de forma aleatoria
174     Random rand = new Random();
175     Position init = new Position(rand.nextInt(100), rand.nextInt(100));
176     xCidadao= init.getX();
177     yCidadao = init.getY();
178     products = new ArrayList<String>();
179
180     products.add("brufen");
181     products.add("ben-u-ron");
182     products.add("aspirina");
```

```
183     products.add("xanax");
184     products.add("valium");
185     products.add("fenistil");
186     products.add("voltaren");
187     products.add("buscopan");
188     products.add("leite NAN");
189     products.add("kompensan");
190     products.add("rennie");
191     products.add("bissolvon");
192     products.add("strepfen");
193
194     produtoRequisitado = "";
195     this.addBehaviour(new Apresentacao());
196     this.addBehaviour(new ContactarFarmacias());
197     this.addBehaviour(new Receiver());
198   }
199   private class Apresentacao extends OneShotBehaviour {
200     public void action() {
201       char aChar = myAgent.getLocalName().charAt(myAgent.getLocalName().length() - 1);
202
203       if (aChar=='0') {
204         nome = "Joao Alves";
205         nif = "123456789";
206         data_nascimento = "18/02/2000";
207         morada = "Rua dos Santos, Braga";
208         cod_postal = "4730-465";
209       }
210       if (aChar=='1') {
211         nome = "Paulo Alves";
212         nif = "126435689";
213         data_nascimento = "26/07/1980";
214         morada = "Rua dos Barbaros, Porto";
215         cod_postal = "4820-465";
216       }
217       if (aChar=='2') {
218         nome = "Rodrigo Cardoso";
219         nif = "222226789";
220         data_nascimento = "18/09/1998";
221         morada = "Rua dos Anjos, Coimbra";
222         cod_postal = "4900-333";
223       }
224       if (aChar=='3') {
225         nome = "Marta Pereira";
226         nif = "123453421";
227         data_nascimento = "20/02/1990";
228         morada = "Rua dos Pepinos, Lisboa";
229         cod_postal = "4875-455";
230       }
```

```
231         if (aChar=='4') {
232           nome = "Jose Marco";
233           nif = "132323789";
234           data_nascimento = "21/05/2000";
235           morada = "Rua dos Morcegos, Braga";
236           cod_postal = "4500-440";
237         }
238         if (aChar=='5') {
239           nome = "Antonio dos Anjos";
240           nif = "987654321";
241           data_nascimento = "23/07/2010";
242           morada = "Rua das Santas, Porto";
243           cod_postal = "4900-444";
244         }
245         if (aChar=='6') {
246           nome = "Manuel Rogerio";
247           nif = "465738291";
248           data_nascimento = "10/01/1970";
249           morada = "Rua dos Macacos, Braga";
250           cod_postal = "4910-432";
251         }
252         if (aChar=='7') {
253           nome = "Jorge Tiago";
254           nif = "965496786";
255           data_nascimento = "29/10/2005";
256           morada = "Rua dos Raios, Porto";
257           cod_postal = "4730-478";
258         }
259         if (aChar=='8') {
260           nome = "Ana Goncalves";
261           nif = "254722777";
262           data_nascimento = "30/04/2001";
263           morada = "Rua dos Felizes, Lisboa";
264           cod_postal = "4810-465";
265         }
266         if (aChar=='9') {
267           nome = "Sofia Marbella";
268           nif = "123876987";
269           data_nascimento = "12/02/1988";
270           morada = "Rua dos Tristes, Lisboa";
271           cod_postal = "4720-323";
272         }
273 
274         System.out.println("Ola eu sou o " + myAgent.getLocalName());
275         System.out.println("\n");
276 
277         System.out.println("Nome: " + nome);
278         System.out.println("NIF: " + nif);
```

```
279       System.out.println("Data nascimento: " + data_nascimento);
280       System.out.println("Morada: " + morada);
281       System.out.println("Codigo-Postal: " + cod_postal);
282
283       System.out.println("\n");
284     }
285   }
286
287   private class ContactarFarmacias extends OneShotBehaviour {
288     public void action() {
289
290       try {
291         // Contactar todas as farmacias
292         DFAgentDescription dfd = new DFAgentDescription();
293         ServiceDescription sd = new ServiceDescription();
294         sd.setType("Farmacia");
295         dfd.addServices(sd);
296
297         DFAgentDescription[] result = DFService.search(this.myAgent, dfd);
298
299         if (result.length > 0) {
300           for (int i = 0; i < result.length; ++i) {
301             // Agent Found
302             DFAgentDescription dfd1 = result[i];
303             AID provider = new AID();
304             provider.setLocalName(dfd1.getName().getLocalName());
305             numFarmacias = result.length;
306
307             //Envia msg a cada uma para pedir as suas coordenadas
308             ACLMessage mensagem = new ACLMessage(ACLMessage.SUBSCRIBE);
309             mensagem.addReceiver(provider);
310             myAgent.send(mensagem);
311           }
312         }
313       }
314         catch (FIPAException fe) {
315           fe.printStackTrace();
316         }
317       }
318     }
319
320   private class Receiver extends CyclicBehaviour {
321     private int xOrigin, yOrigin;
322     private int minDistance = 1000;
323     private String closestFarmacia; //farmacia mais proxima
324     private int FarmaciasProcessadas = 0;
325
326
```

```
327     @Override
328     public void action() {
329       ACLMessage msg = receive();
330       name = myAgent.getLocalName();
331       if (msg != null) {
332         //Recebe msg da farmacia
333         if (msg.getPerformative() == ACLMessage.INFORM) { // RECEBE COORDENADAS
334           String[] coordinates = msg.getContent().split(","); //Separa as coordenadas
                  e atribui o valor as variaveis definidas inicialmente
335           FarmaciasProcessadas++;
336           xOrigin = Integer.parseInt(coordinates[0]);
337           yOrigin = Integer.parseInt(coordinates[1]);
338
339           // Calcula a distancia entre o cidadao e cada farmacia
340           int distance = (int) Math
341               .sqrt(((Math.pow((xCidadao - xOrigin), 2)) + (Math.pow((yCidadao -
                      yOrigin), 2))));
342
343           if (distance < minDistance) {
344             minDistance = distance; //altera a distancia minima
345             closestFarmacia = msg.getSender().getLocalName(); //altera a farmacia +
                    proxima
346           }
347           if (FarmaciasProcessadas == numFarmacias) { //se ja processei todas as
                  farmacias
348
349             Random randomizer = new Random();
350             String random = products.get(randomizer.nextInt(products.size())); //
                    Escolhe um produto random da lista
351             produtoRequisitado = random;
352             Random rand = new Random();
353             while (true) {
354               quantidade = rand.nextInt(10);
355               if (quantidade !=0) {
356                 break;
357               }
358             }
359             closestFarmacia="Farmacia0";
360             produtoRequisitado="rennie";
361
362             System.out.println(name + ": Escolheu a " + closestFarmacia + " pedindo: "
                     + quantidade + " " + produtoRequisitado);
363             System.out.print("\n");
364
365             ACLMessage mensagem = new ACLMessage(ACLMessage.REQUEST); //envia msg a
                    farmacia para fazer o pedido
366             AID receiver = new AID();
367             receiver.setLocalName(closestFarmacia);
```

```
368             mensagem.addReceiver(receiver);
369             mensagem.setContent("Cidadao:" + "," + xCidadao + "," + yCidadao + "," +
                    produtoRequisitado + ","+ quantidade );
370             myAgent.send(mensagem);
371             FarmaciasProcessadas = 0; //reinicia as variaveis
372             minDistance = 1000;
373             closestFarmacia = null;
374             }
375         }
376
377         else if (msg.getPerformative() == ACLMessage.AGREE) {
378           name = myAgent.getLocalName();
379           System.out.println(name + ": Recebi o meu pedido. Obrigado!" + msg.getSender
                ().getLocalName());
380           System.out.print("\n");
381           }
382       }
383     }
384   }
385   protected void takeDown() {
386     System.out.println("Ending Customer");
387     super.takeDown();
388   }
389 }
```

C. Agente Farmacia

```
390 package Agents;
391
392 import java.io.IOException;
393 import java.io.Serializable;
394 import java.util.HashMap;
395 import java.util.Random;
396
397 import Classes.Farm;
398 import Classes.Position;
399 import jade.core.AID;
400 import jade.core.Agent;
401 import jade.core.behaviours.CyclicBehaviour;
402 import jade.domain.DFService;
403 import jade.domain.FIPAException;
404 import jade.domain.FIPAAgentManagement.DFAgentDescription;
405 import jade.domain.FIPAAgentManagement.ServiceDescription;
406 import jade.lang.acl.ACLMessage;
407
408 public class Farmacia extends Agent {
409
```

```
410   int xFarmacia, yFarmacia;
411
412   private HashMap<String, Integer> stock_medicamentos = new HashMap<>();
413   private HashMap<String, Integer> preco_medicamentos = new HashMap<>();
414
415   private Farm farmacia;
416
417   protected void setup() {
418     super.setup();
419
420     System.out.print(
421         "-----------------------------------------------------------------\n");
422
423     System.out.print(
424         "                    Starting Farmacia                            \n");
425
426     System.out.print(
427         "-----------------------------------------------------------------\n");
428
429     System.out.print("\n");
430
431     // As coordenadas da farmacia sao escolhidas de forma aleatoria
432     Random rand = new Random();
433
434     xFarmacia = rand.nextInt(100);
435     yFarmacia = rand.nextInt(100);
436
437     Position init = new Position(xFarmacia, yFarmacia);
438
439     stock_medicamentos.put("brufen", 9); stock_medicamentos.put("ben-u-ron", 9);
          stock_medicamentos.put("aspirina", 9); stock_medicamentos.put("xanax", 9);
440     stock_medicamentos.put("valium", 9); stock_medicamentos.put("fenistil", 9);
          stock_medicamentos.put("voltaren", 9); stock_medicamentos.put("buscopan", 9);
441     stock_medicamentos.put("leite NAN", 9);stock_medicamentos.put("kompensan", 9);
          stock_medicamentos.put("rennie", 9);stock_medicamentos.put("bissolvon", 9);
442     stock_medicamentos.put("strepfen", 9);
443
444     preco_medicamentos.put("brufen", 10); preco_medicamentos.put("ben-u-ron", 12);
          preco_medicamentos.put("aspirina", 8); preco_medicamentos.put("xanax", 10);
445     preco_medicamentos.put("valium", 11); preco_medicamentos.put("fenistil", 15);
          preco_medicamentos.put("voltaren", 13); preco_medicamentos.put("buscopan", 8);
446     preco_medicamentos.put("leite NAN", 10);preco_medicamentos.put("kompensan", 10);
          preco_medicamentos.put("rennie", 15);preco_medicamentos.put("bissolvon", 10);
447     preco_medicamentos.put("strepfen", 12);
448
449     // Cada farmacia regista-se nas paginas amarelas
450     DFAgentDescription dfd = new DFAgentDescription();
451     dfd.setName(getAID());
```

```
452     ServiceDescription sd = new ServiceDescription();
453     sd.setType("Farmacia");
454     sd.setName(getLocalName());
455     dfd.addServices(sd);
456 
457     try {
458       DFService.register(this, dfd);
459     } catch (FIPAException fe) {
460       fe.printStackTrace();
461     }
462 
463     //Comportamento
464     addBehaviour(new Receiver());
465   }
466 
467 private class Receiver extends CyclicBehaviour {
468 
469     private int xCidadao, yCidadao;
470     private String Pedido;
471     private int quantidade;
472     private String customerName;
473 
474     public void action() {
475       ACLMessage msg = receive();
476       if (msg != null) {
477         //Recebe msg do cidadao e envia-lhe as suas coordenadas
478         if (msg.getPerformative() == ACLMessage.SUBSCRIBE) {
479 
480           System.out.println(myAgent.getLocalName() + ": Recebi pedido do " + msg.
                 getSender().getLocalName() +
481               " para enviar as minhas coordenadas");
482           System.out.print("\n");
483 
484           AID provider = msg.getSender();
485           ACLMessage response = new ACLMessage(ACLMessage.INFORM); //responde a msg do
                  cidadao mas agora no tipo INFORM
486           response.addReceiver(provider);
487           response.setContent(xFarmacia + "," + yFarmacia); //envia as suas
                 coordenadas
488           myAgent.send(response);
489         }
490         //Recebe msg do gestor e envia-lhe as suas coordenadas
491         else if (msg.getPerformative() == ACLMessage.CFP) {
492 
493           try {
494             System.out.println(myAgent.getLocalName() + ": Recebi pedido do Gestor
                    para enviar as minhas informacoes");
495             System.out.print("\n");
```

```
496             Random rand = new Random();
497             farmacia = new Farm(myAgent.getAID(),
498                 new Position(xFarmacia, yFarmacia));
499
500             ACLMessage mensagem = new ACLMessage(ACLMessage.SUBSCRIBE);
501             mensagem.setContentObject(farmacia);
502
503             mensagem.addReceiver(msg.getSender());
504
505             myAgent.send(mensagem);
506
507             }
508
509             catch (IOException e) {
510               // TODO Auto-generated catch block
511               e.printStackTrace();
512             }
513         }
514
515         //Recebe msg da interface e envia-lhe as suas coordenadas
516         else if (msg.getPerformative() == ACLMessage.PROPOSE) {
517           try {
518             System.out.println(myAgent.getLocalName() + ": Recebi pedido da Interface
                  para enviar as minhas informacoes");
519             System.out.print("\n");
520
521             Random rand = new Random();
522             farmacia = new Farm(myAgent.getAID(),
523                 new Position(xFarmacia, yFarmacia));
524
525             ACLMessage mensagem = new ACLMessage(ACLMessage.SUBSCRIBE);
526             mensagem.setContentObject(farmacia);
527
528             mensagem.addReceiver(msg.getSender());
529
530             myAgent.send(mensagem);
531           }
532           catch (IOException e) {
533             // TODO Auto-generated catch block
534             e.printStackTrace();
535           }
536         }
537         else if (msg.getPerformative() == ACLMessage.REQUEST) {
538           String[] Pedido_Coordenadas = msg.getContent().split(","); //Separa as
                coordenadas do cidadao e atribui o valor as variaveis definidas
                inicialmente
539           xCidadao = Integer.parseInt(Pedido_Coordenadas[1]);
540           yCidadao = Integer.parseInt(Pedido_Coordenadas[2]);
```

```
541             Pedido = Pedido_Coordenadas[3];
542             quantidade = Integer.parseInt(Pedido_Coordenadas[4]);
543             customerName = msg.getSender().getLocalName();
544
545             if (stock_medicamentos.get(Pedido) >=quantidade) {
546               //Confirma o pedido ao cidadao
547
548               System.out.println(myAgent.getLocalName() + ": Confirma o pedido ao " +
                      customerName);
549               System.out.print("\n");
550
551               //Adormece durante 0.5 segundos para simular o tempo de entrega
552               try {
553                 Thread.sleep(500);
554               } catch (InterruptedException e) {
555                 // TODO Auto-generated catch block
556                 e.printStackTrace();
557               }
558
559               //E avisa o cidadao que o seu pedido foi entregue
560               AID receiver = new AID();
561               receiver.setLocalName(customerName);
562               ACLMessage mensagem = new ACLMessage(ACLMessage.AGREE);
563               mensagem.addReceiver(receiver);
564               mensagem.setContent("O seu pedido foi entregue!");
565               myAgent.send(mensagem);
566
567               int stock = stock_medicamentos.get(Pedido);
568
569               int left = stock - quantidade;
570
571               stock_medicamentos.put(Pedido, left);
572
573               int preco = preco_medicamentos.get(Pedido);
574
575               int receita = preco * quantidade;
576
577               AID receiver2 = new AID();
578               receiver2.setLocalName("Gestor");
579               ACLMessage mensagem2 = new ACLMessage(ACLMessage.INFORM_REF);
580               mensagem2.setContent(myAgent.getLocalName() + "," + "aumentar 1 unidade
                      pedidos e aumentar receita para"+ "," + receita + "," + "atualizei
                      stock:" + "," + Pedido +"," + "para" + "," + left + "," + customerName
                       +"," + quantidade);
581               mensagem2.addReceiver(receiver2);
582
583               System.out.println(myAgent.getLocalName() + ": Envia de informacao de
                      atualizacao de stock ao Gestor");
```

```
584             System.out.print("\n");
585
586             myAgent.send(mensagem2);
587           }
588           else {
589             System.out.println(myAgent.getLocalName() + ": Pede ao Gestor a 2 farmacia
                    mais proxima");
590             System.out.print("\n");
591             //Envia pedido ao Gestor para ele verificar o stock
592             AID receiver = new AID();
593             receiver.setLocalName("Gestor");
594             ACLMessage mensagem = new ACLMessage(ACLMessage.REQUEST);
595             mensagem.setContent(myAgent.getLocalName() + " nao tenho stock para este
                  pedido:" + ","+  customerName + "," + xCidadao + "," + yCidadao + ","
                  + xFarmacia + "," + yFarmacia + "," + Pedido + "," + quantidade);
596             mensagem.addReceiver(receiver);
597
598             System.out.println(myAgent.getLocalName() + ": Envia pedido ao Gestor");
599             System.out.print("\n");
600
601             myAgent.send(mensagem);
602           }
603         }
604         else if (msg.getPerformative() == ACLMessage.INFORM) {
605           String[] informacao = msg.getContent().split(",");
606
607           String pedido = informacao[1];
608           int quantidade = Integer.parseInt(informacao[2]);
609           String cidadao_name = informacao[4];
610
611           System.out.println(myAgent.getLocalName() + ": Confirma o pedido ao " +
                  cidadao_name);
612           System.out.print("\n");
613
614           ACLMessage resp = msg.createReply();
615           resp.setPerformative(ACLMessage.CONFIRM);
616           resp.setContent("FARMACIA: Confirma ao cidadao o seu pedido para o produto"
                  + "," + pedido + quantidade);
617           System.out.print("\n");
618
619           //Adormece durante 0.5 segundos para simular o tempo de entrega
620           try {
621             Thread.sleep(500);
622           } catch (InterruptedException e) {
623             // TODO Auto-generated catch block
624             e.printStackTrace();
625           }
626
```

```
627             //E avisa o cidadao que o seu pedido foi entregue
628             AID receiver = new AID();
629             receiver.setLocalName(cidadao_name);
630             ACLMessage mensagem = new ACLMessage(ACLMessage.AGREE);
631             mensagem.addReceiver(receiver);
632             mensagem.setContent("O seu pedido foi entregue!");
633             myAgent.send(mensagem);
634
635             int stock = stock_medicamentos.get(pedido);
636
637             int left = stock - quantidade;
638
639             stock_medicamentos.put(pedido, left);
640
641             int preco = preco_medicamentos.get(pedido);
642
643             int preco_total = preco * quantidade;
644
645             AID receiver2 = new AID();
646             receiver2.setLocalName("Gestor");
647             ACLMessage mensagem2 = new ACLMessage(ACLMessage.INFORM_REF);
648             mensagem2.setContent(myAgent.getLocalName() + "," + "aumentar 1 unidade
                    pedidos e aumentar lucro para"+ "," + preco_total + "," + "atualizei
                    stock:" + "," + pedido +"," + "para" + "," + left + "," + cidadao_name +
                     "," + quantidade);
649             mensagem2.addReceiver(receiver2);
650
651             System.out.println(myAgent.getLocalName() + ": Envia de informacao de
                    atualizacao de stock ao Gestor");
652             System.out.print("\n");
653
654             myAgent.send(mensagem2);
655         }
656
657         else if (msg.getPerformative() == ACLMessage.INFORM_IF) {
658
659             System.out.println(myAgent.getLocalName() + ": Recebi o lote que precisava.
                    Obrigado! " + msg.getSender().getLocalName());
660             System.out.print("\n");
661
662             String[] informacao = msg.getContent().split(",");
663
664             String med = informacao[1];
665             String ID_Farmacia = informacao[3];
666             int despesa = Integer.parseInt(informacao[5]);
667
668             stock_medicamentos.put(med, 9);
669             int quantidade = 9;
```

```
670             AID receiver = new AID();
671             receiver.setLocalName("Gestor");
672             ACLMessage mensagem = new ACLMessage(ACLMessage.INFORM);
673             mensagem.setContent(myAgent.getLocalName() + "," + "atualizei stock:" + ","
                    + med +"," + "para " + "," + quantidade + "," + despesa);
674             mensagem.addReceiver(receiver);
675
676             System.out.println(myAgent.getLocalName() + ": Envia a informacao de
                    atualizacao de stock ao Gestor");
677             System.out.print("\n");
678
679             myAgent.send(mensagem);
680           }
681         }
682         else {
683           block();
684         }
685       }
686     }
687 }
```

D. Agente Fornecedor

```
690 package Agents;
691
692 import java.io.IOException;
693 import java.util.HashMap;
694 import java.util.Random;
695
696 import Classes.Forn;
697 import Classes.Position;
698 import jade.core.AID;
699 import jade.core.Agent;
700 import jade.core.behaviours.CyclicBehaviour;
701 import jade.domain.DFService;
702 import jade.domain.FIPAException;
703 import jade.domain.FIPAAgentManagement.DFAgentDescription;
704 import jade.domain.FIPAAgentManagement.ServiceDescription;
705 import jade.lang.acl.ACLMessage;
706
707 public class Fornecedor extends Agent {
708   private HashMap<String, Integer> med_prec = new HashMap<>();
709
710   protected void setup() {
711     super.setup();
712
713
```

```
714     System.out.print(
715         "---------------------------------------------------------------\n");
716 
717     System.out.print(
718         "                    Starting Fornecedor                        \n");
719 
720     System.out.print(
721         "---------------------------------------------------------------\n");
722 
723     System.out.print("\n");
724 
725     // Cada fornecedor regista-se nas paginas amarelas
726     DFAgentDescription dfd = new DFAgentDescription();
727     dfd.setName(getAID());
728     ServiceDescription sd = new ServiceDescription();
729     sd.setType("Fornecedor");
730     sd.setName(getLocalName());
731     dfd.addServices(sd);
732 
733     try {
734       DFService.register(this, dfd);
735     } catch (FIPAException fe) {
736       fe.printStackTrace();
737     }
738 
739     //Comportamento
740     addBehaviour(new Receiver());
741   }
742 
743 private class Receiver extends CyclicBehaviour {
744 
745     public void action() {
746       ACLMessage msg = receive();
747       if (msg != null) {
748         //Recebe msg do gestor e envia-lhe as suas coordenadas
749         if (msg.getPerformative() == ACLMessage.CFP) {
750 
751           try {
752             System.out.println(myAgent.getLocalName() + ": Recebi pedido do Gestor
                    para enviar as minhas informacoes");
753             System.out.print("\n");
754 
755             char aChar = myAgent.getLocalName().charAt(10);
756             //System.out.println(aChar=='0');
757 
758             if (aChar=='0') {
759               med_prec.put("brufen", 5); med_prec.put("ben-u-ron", 6); med_prec.put("
                    aspirina", 7); med_prec.put("xanax", 8);
```

```
760             med_prec.put("valium", 9); med_prec.put("fenistil", 10); med_prec.put("
                    voltaren", 5); med_prec.put("buscopan", 6);
761             med_prec.put("leite NAN", 7);med_prec.put("kompensan", 8);med_prec.put("
                    rennie", 11);med_prec.put("bissolvon", 7);
762             med_prec.put("strepfen", 9);
763           }
764
765           if (aChar=='1') {
766             med_prec.put("brufen", 9); med_prec.put("ben-u-ron", 10); med_prec.put("
                    aspirina", 3); med_prec.put("xanax", 5);
767             med_prec.put("valium", 7); med_prec.put("fenistil", 6); med_prec.put("
                    voltaren", 9); med_prec.put("buscopan", 5);
768             med_prec.put("leite NAN", 8);med_prec.put("kompensan", 7);med_prec.put("
                    rennie", 7);med_prec.put("bissolvon", 9);
769             med_prec.put("strepfen", 10);
770           }
771
772           Forn fornecedor = new Forn(myAgent.getAID());
773           fornecedor.setSMed_preco(med_prec);
774           ACLMessage mensagem = new ACLMessage(ACLMessage.CFP);
775           mensagem.setContentObject(fornecedor);
776
777           mensagem.addReceiver(msg.getSender());
778
779           myAgent.send(mensagem);
780           }
781
782           catch (IOException e) {
783             // TODO Auto-generated catch block
784             e.printStackTrace();
785           }
786       }
787       if (msg.getPerformative() == ACLMessage.INFORM) {
788
789         String[] pedido = msg.getContent().split(",");
790         String ID_Farmacia = pedido[5];
791         String Pedido = pedido[1];
792         String quantidade = pedido[3];
793         int depesa = Integer.parseInt(pedido[6]);
794
795         System.out.println(myAgent.getLocalName() + ": Recebe pedido do GESTOR para
                reestabelecer o stock na " +
796         ID_Farmacia + " para o produto " + Pedido);
797         System.out.print("\n");
798
799         //Adormece durante 0.5 segundos para simular o tempo de entrega
800         try {
801           Thread.sleep(500);
```

```
802             } catch (InterruptedException e) {
803               // TODO Auto-generated catch block
804               e.printStackTrace();
805             }
806             //Avisa farmacia para reestabelecer o stock
807             AID receiver_2 = new AID();
808             receiver_2.setLocalName(ID_Farmacia);
809             ACLMessage mensagem_2 = new ACLMessage(ACLMessage.INFORM_IF);
810             mensagem_2.addReceiver(receiver_2);
811             mensagem_2.setContent("O stock para o medicamento" + "," + Pedido + "," +
812             "para a farmacia" + "," + ID_Farmacia + "," + "tem que ser reestabelecido
                    para 9" + "," + depesa);
813             myAgent.send(mensagem_2);
814           }
815         }
816       }
817     }
818 }
```

E. Agente Gestor

```
820 package Agents;
821
822 import java.io.IOException;
823 import java.util.HashMap;
824
825 import jade.core.AID;
826 import jade.core.Agent;
827 import jade.core.behaviours.CyclicBehaviour;
828 import jade.core.behaviours.OneShotBehaviour;
829 import jade.domain.DFService;
830 import jade.domain.FIPAException;
831 import jade.domain.FIPAAgentManagement.DFAgentDescription;
832 import jade.domain.FIPAAgentManagement.ServiceDescription;
833 import jade.lang.acl.ACLMessage;
834 import jade.lang.acl.UnreadableException;
835 import Classes.Farm;
836 import Classes.Forn;
837 import Classes.Position;
838
839 public class Gestor extends Agent {
840
841   private HashMap<String, Farm> farm_gestor = new HashMap<>();
842   private HashMap<String, Forn> forn_gestor = new HashMap<>();
843   private int threshold = 2;
844
845
```

```
846   protected void setup() {
847     super.setup();
848     System.out.print(
849         "@@@@@@@@@@@@@@@@@@@@     Starting Gestor    @@@@@@@@@@@@@@@@@@@@\n");
850     System.out.print("\n");
851 
852     addBehaviour(new ContactarFarmacias());
853     addBehaviour(new Receiver());
854   }
855 
856   private class ContactarFarmacias extends OneShotBehaviour {
857     private static final long serialVersionUID = 1L;
858     private int numFarmacias;
859     private int numFornecedores;
860 
861     public void action() {
862 
863       try {
864         // Contactar todas as farmacias
865         DFAgentDescription dfd = new DFAgentDescription();
866         ServiceDescription sd = new ServiceDescription();
867         sd.setType("Farmacia");
868         dfd.addServices(sd);
869 
870         DFAgentDescription[] result = DFService.search(this.myAgent, dfd);
871         String[] farmacias; //array de AID das farmacias
872         farmacias = new String[result.length]; //array de AID das farmacias com
              tamanho do array result
873         numFarmacias = result.length;
874 
875         for (int i = 0; i < numFarmacias; ++i) {
876           farmacias[i] = result[i].getName().getLocalName(); //introduz no array
                farmacia o nome de cada farmacia
877 
878           //envia msg a cada farmacia para ela enviar as suas coordenadas
879           ACLMessage mensagem = new ACLMessage(ACLMessage.CFP);
880           AID receiver = new AID();
881           receiver.setLocalName(farmacias[i]);
882           mensagem.addReceiver(receiver);
883           myAgent.send(mensagem);
884           }
885 
886         // Contactar todos os fornecedores
887         DFAgentDescription dfd2 = new DFAgentDescription();
888         ServiceDescription sd2 = new ServiceDescription();
889         sd2.setType("Fornecedor");
890         dfd2.addServices(sd2);
891 
```

```
892         DFAgentDescription[] result2 = DFService.search(this.myAgent, dfd2);
893         String[] fornecedores; //array de AID das fornecedores
894         fornecedores = new String[result2.length]; //array de AID dos fornecedores com
                tamanho do array result2
895         numFornecedores = result2.length;
896
897         for (int i = 0; i < numFornecedores; ++i) {
898           fornecedores[i] = result2[i].getName().getLocalName(); //introduz no array
                fornecedor o nome de cada fornecedor
899
900           //envia msg a cada fornecedor para ela enviar as suas coordenadas
901           ACLMessage mensagem2 = new ACLMessage(ACLMessage.CFP);
902           AID receiver2 = new AID();
903           receiver2.setLocalName(fornecedores[i]);
904           mensagem2.addReceiver(receiver2);
905           myAgent.send(mensagem2);
906           }
907
908         } catch (FIPAException e) {
909         e.printStackTrace();
910         }
911       }
912     }
913
914   private class Receiver extends CyclicBehaviour {
915     private static final long serialVersionUID = 1L;
916     private String Farmacia;
917     private String Pedido;
918     private double tempo;
919     private String[] med,med1,med2,med3;
920     private HashMap<String, Integer> stock;
921     private int xFarmacia, yFarmacia, xCidadao, yCidadao;
922     private int minDistance = 1000;
923     private String cidadaoName;
924     private int quantidade;
925     AID secondFarmacia = null;
926     AID fornecedor_barato = null;
927     Forn Forn_mais_barato = null;
928     Forn Forn_mais_barato2 = null;
929     private String Fornecedor;
930     AID Farmacia2 = null;
931     int preco = 1000;
932     int preco2 = 1000;
933
934     public void action() {
935
936       ACLMessage msg = receive();
937       if (msg != null) {
```

```
938         //Recebe coordenadas de cada farmacia
939         if (msg.getPerformative() == ACLMessage.SUBSCRIBE) {
940           AID Farmacia2 = msg.getSender();
941           Farmacia= msg.getSender().getLocalName();
942
943           //Cria uma nova Class_Farmacia guardando no hashmap Farmacias
944           try {
945             Farm a = (Farm) msg.getContentObject();
946             farm_gestor.put(Farmacia,a);
947
948           } catch (UnreadableException e) {
949             // TODO Auto-generated catch block
950             e.printStackTrace();
951           }
952         }
953
954         //Recebe coordenadas de cada fornecedor
955         else if (msg.getPerformative() == ACLMessage.CFP) {
956           try {
957             Forn fornecedor = (Forn) msg.getContentObject();
958             Fornecedor = msg.getSender().getLocalName();
959             forn_gestor.put(Fornecedor,fornecedor);
960           } catch (UnreadableException e) {
961             // TODO Auto-generated catch block
962             e.printStackTrace();
963           }
964         }
965
966         else if (msg.getPerformative() == ACLMessage.REQUEST) {
967           int stock_pedido = 0;
968
969           String[] pedido = msg.getContent().split(",");
970           Farmacia =msg.getSender().getLocalName();
971           Pedido = pedido[6];
972           xCidadao = Integer.parseInt(pedido[2]);
973           yCidadao = Integer.parseInt(pedido[3]);
974           xFarmacia=Integer.parseInt(pedido[4]);
975           yFarmacia=Integer.parseInt(pedido[5]);
976           cidadaoName = pedido[1];
977           quantidade = Integer.parseInt(pedido[7]);
978
979           System.out.println(myAgent.getLocalName() + ": Recebe pedido da " + Farmacia
                    + " para ver a 2 Farmacia mais proxima com stock para "
980               + "o produto " + quantidade + " " + Pedido);
981           System.out.print("\n");
982
983           //Percorre as farmacias e escolhe todas a excecao da farmacia sem stock
984           for (HashMap.Entry<String, Farm> pair  : farm_gestor.entrySet()) {
```

```
985             Farm f=pair.getValue();
986             if (!f.getAgent().getLocalName().equals(Farmacia)) {
987               stock=f.getStock_medicamentos(); //obtem a lista de medicamentos
988               xFarmacia = f.getPosition().getX();
989                 yFarmacia = f.getPosition().getY();
990
991                 //Percorre med2 ate encontrar o produto pretendido
992               for(HashMap.Entry<String, Integer> med  : stock.entrySet()) {
993                 if(med.getKey().equals(Pedido)) {
994                   stock_pedido = med.getValue();//Guarda a quantidade em stock desse
                          mesmo produto
995
996                   //So calcula a distancia se houver stock, caso contrario, avanca
                          para a proxima farmacia
997                   if (stock_pedido>=quantidade) {
998                     int distance = (int) Math.sqrt(((Math.pow((xCidadao - xFarmacia),
                            2)) +
999                         (Math.pow((yCidadao - yFarmacia), 2))));
1000                    if (distance < minDistance) {
1001                      minDistance = distance; //altera a distancia minima
1002                      secondFarmacia = f.getAgent();
1003                    }
1004                  }
1005                }
1006              }
1007            }
1008          }
1009          //depois de percorrer todas as farmacias, contactar a 2 mais proxima
1010          //Avisa o GESTOR da 2 farmacia mais proxima
1011          ACLMessage mensagem = new ACLMessage(ACLMessage.INFORM);
1012          AID receiver = new AID();
1013
1014          mensagem.addReceiver(secondFarmacia);
1015
1016          System.out.println(myAgent.getLocalName() + ": Avisar a " + secondFarmacia.
                 getLocalName() + " que e a 2 farmacia mais proxima");
1017          System.out.print("\n");
1018
1019          mensagem.setContent("Tera de enviar este produto" + "," + Pedido + "," +
                 quantidade + "," + "ao cidadao" + "," + cidadaoName);
1020          myAgent.send(mensagem);
1021
1022          minDistance = 1000; //REINICIA A VARIAVEL
1023          secondFarmacia = null;
1024
1025          //Percorre todos os fornecedores e escolhe o mais barato
1026          for (HashMap.Entry<String, Forn> pair  : forn_gestor.entrySet()) {
1027            Forn f=pair.getValue();
```

```
1028            int preco_forn=f.buscar_preco_med(Pedido); //obtem a lista de medicamentos
1029            if (preco_forn < preco) {
1030              preco = preco_forn;
1031              Forn_mais_barato = f;
1032            }
1033          }
1034
1035          Farm farma = farm_gestor.get(Farmacia);
1036          int stock_1farmacia = farma.getStock_medicamentos().get(Pedido);
1037          int despesa = (9-stock_1farmacia) * Forn_mais_barato.buscar_preco_med(Pedido
                  );
1038
1039          ACLMessage mensagem2 = new ACLMessage(ACLMessage.INFORM);
1040          AID receiver2 = new AID();
1041          receiver2 = Forn_mais_barato.getAgent();
1042          mensagem2.addReceiver(receiver2);
1043
1044          mensagem2.setContent("Tera de enviar este produto" + "," + Pedido + "," + "
                  quantidade" + "," + quantidade + "," + " a farmacia" + "," + Farmacia +
                  "," + despesa);
1045          myAgent.send(mensagem2);
1046
1047          preco=1000;
1048          Forn_mais_barato=null;
1049        }
1050
1051        else if (msg.getPerformative() == ACLMessage.INFORM_REF) {
1052          //int preco2 = 1000;
1053          String[] informacao = msg.getContent().split(",");
1054
1055          String med = informacao[4];
1056          String ID_Farmacia = informacao[0];
1057          int quantidade = Integer.parseInt(informacao[6]);
1058          int quantidade2 = Integer.parseInt(informacao[8]);
1059          int receita = Integer.parseInt(informacao[2]);
1060          String customerName= informacao[7];
1061          Farm Farmacia = farm_gestor.get(ID_Farmacia);
1062          int vendeu = Farmacia.getStock_medicamentos().get(med) - quantidade;
1063          //System.out.println(med);
1064          Farmacia.alterarStock(quantidade, med);
1065          Farmacia.setReceita(receita);
1066          Farmacia.setPedidos();
1067          HashMap<String, Integer> produtos_vendidos= Farmacia.getProdutos_vendidos();
1068
1069          int vendidos = produtos_vendidos.get(med) + vendeu;
1070          produtos_vendidos.put(med, vendidos);
1071          HashMap<String, Integer> farm_cidadao= Farmacia.getCidadaos();
1072          farm_cidadao.put(customerName, quantidade2);
```

```
1073            if (quantidade < threshold) {
1074            //Percorre as farmacias e escolhe todas a excecao da farmacia sem stock
1075            for (HashMap.Entry<String, Forn> pair  : forn_gestor.entrySet()) {
1076              Forn f=pair.getValue();
1077
1078              int preco_forn=f.buscar_preco_med(med); //obtem a lista de medicamentos
1079
1080              if (preco_forn < preco2) {
1081                preco2 = preco_forn;
1082                Forn_mais_barato2 = f;
1083              }
1084            }
1085            int despesa = (9-quantidade) * Forn_mais_barato2.buscar_preco_med(med);
1086
1087            ACLMessage mensagem2 = new ACLMessage(ACLMessage.INFORM);
1088            AID receiver2 = new AID();
1089            receiver2 = Forn_mais_barato2.getAgent();
1090            mensagem2.addReceiver(receiver2);
1091
1092            mensagem2.setContent("Tera de enviar este produto" + "," + med + "," + "
                    quantidade" + "," + quantidade + "," + " a farmacia" + "," +
                    ID_Farmacia + "," + despesa);
1093            myAgent.send(mensagem2);
1094            preco2=1000;
1095            Forn_mais_barato2=null;
1096          }
1097          else {
1098             try {
1099              ACLMessage mensagem = new ACLMessage(ACLMessage.INFORM);
1100              AID receiver = new AID();
1101              //int valor = 1;
1102              receiver.setLocalName("Interface");
1103
1104              System.out.println(myAgent.getLocalName() + ": Avisar a Interface para
                      imprimir resultados!");
1105              System.out.print("\n");
1106
1107              mensagem.setContentObject(farm_gestor);
1108              mensagem.addReceiver(receiver);
1109              myAgent.send(mensagem);
1110             }
1111             catch (IOException e) {
1112                // TODO Auto-generated catch block
1113                e.printStackTrace();
1114             }
1115          }
1116        }
1117
```

```
1118          else if (msg.getPerformative() == ACLMessage.INFORM) {
1119            String[] informacao = msg.getContent().split(",");
1120
1121            String med = informacao[2];
1122            String ID_Farmacia = informacao[0];
1123            int quantidade = Integer.parseInt(informacao[4]);
1124            int despesa = Integer.parseInt(informacao[5]);
1125
1126            Farm Farmacia = farm_gestor.get(ID_Farmacia);
1127
1128            HashMap<String, Integer> stock_med = Farmacia.getStock_medicamentos();
1129            int stock = 10;
1130
1131            Farmacia.alterarStock(9, med);
1132            Farmacia.setDespesa(despesa);
1133
1134            try {
1135              ACLMessage mensagem = new ACLMessage(ACLMessage.INFORM);
1136              AID receiver = new AID();
1137
1138              receiver.setLocalName("Interface");
1139
1140              System.out.println(myAgent.getLocalName() + ": Avisar a Interface para
                      imprimir resultados!");
1141              System.out.print("\n");
1142
1143              mensagem.setContentObject(farm_gestor);
1144              mensagem.addReceiver(receiver);
1145              myAgent.send(mensagem);
1146            }
1147            catch (IOException e) {
1148              // TODO Auto-generated catch block
1149              e.printStackTrace();
1150            }
1151          }
1152        }
1153        else {
1154          block();
1155        }
1156      }
1157    }
1158  }
```

F. Agente Interface

```
1162 package Agents;
1163
1164 import java.awt.Color;
1165 import java.awt.Dimension;
1166 import java.io.Serializable;
1167 import java.text.DecimalFormat;
1168 import java.util.ArrayList;
1169 import java.util.HashMap;
1170 import java.util.Map;
1171
1172 import Classes.Farm;
1173 import Classes.Position;
1174 import jade.core.AID;
1175 import jade.core.Agent;
1176 import jade.core.behaviours.CyclicBehaviour;
1177 import jade.core.behaviours.OneShotBehaviour;
1178 import jade.domain.DFService;
1179 import jade.domain.FIPAException;
1180 import jade.domain.FIPAAgentManagement.DFAgentDescription;
1181 import jade.domain.FIPAAgentManagement.ServiceDescription;
1182 import jade.lang.acl.ACLMessage;
1183 import jade.lang.acl.UnreadableException;
1184
1185 import org.jfree.chart.ChartFactory;
1186 import org.jfree.chart.ChartFrame;
1187 import org.jfree.chart.ChartPanel;
1188 import org.jfree.chart.JFreeChart;
1189 import org.jfree.chart.labels.PieSectionLabelGenerator;
1190 import org.jfree.chart.labels.StandardPieSectionLabelGenerator;
1191 import org.jfree.chart.plot.CategoryPlot;
1192 import org.jfree.chart.plot.PiePlot;
1193 import org.jfree.chart.plot.PlotOrientation;
1194 import org.jfree.chart.renderer.category.CategoryItemRenderer;
1195 import org.jfree.data.category.DefaultCategoryDataset;
1196 import org.jfree.data.general.DefaultPieDataset;
1197 import com.orsoncharts.plot.PiePlot3D;
1198
1199 public class Interface extends Agent {
1200   private Map<String,Farm> Farmacias = new HashMap<>() ;
1201   //private Map<String,Integer> dic;
1202   private String farmaciaComMaisPedidos;
1203   private String farmacia_maior_lucro;
1204
1205   private static final long serialVersionUID = 1L;
1206   protected void setup() {
1207     super.setup();
```

```
1208    System.out.print(
1209        "######################   Starting Interface    #################\n");
1210    System.out.print("\n");
1211
1212    //Behaviours
1213    addBehaviour(new ContactarFarmacias());
1214    addBehaviour(new Receiver());
1215  }
1216
1217  private class ContactarFarmacias extends OneShotBehaviour {
1218    private static final long serialVersionUID = 1L;
1219    private int numFarmacias;
1220
1221    public void action() {
1222
1223      try {
1224        // Contactar todas as farmacias
1225        DFAgentDescription dfd = new DFAgentDescription();
1226        ServiceDescription sd = new ServiceDescription();
1227        sd.setType("Farmacia");
1228        dfd.addServices(sd);
1229
1230        DFAgentDescription[] result = DFService.search(this.myAgent, dfd);
1231        String[] farmacias; //array de AID das farmacias
1232        farmacias = new String[result.length]; //array de AID das farmacias com
              tamanho do array result
1233        numFarmacias = result.length;
1234
1235        for (int i = 0; i < result.length; ++i) {
1236          farmacias[i] = result[i].getName().getLocalName(); //introduz no array
                farmacia o nome de cada farmacia
1237
1238          //envia msg a cada farmacia para ela enviar as suas coordenadas
1239          ACLMessage mensagem = new ACLMessage(ACLMessage.PROPOSE);
1240          AID receiver = new AID();
1241          receiver.setLocalName(farmacias[i]);
1242          mensagem.addReceiver(receiver);
1243          myAgent.send(mensagem);
1244          }
1245
1246        } catch (FIPAException e) {
1247        e.printStackTrace();
1248        }
1249      }
1250    }
1251
1252
1253
```

```
1254  private class Receiver extends CyclicBehaviour {
1255
1256    private static final long serialVersionUID = 1L;
1257    private String ID_Farmacia;
1258    private String Pedido;
1259    private double tempo;
1260    private String[] med,med1,med2,med3;
1261    private Integer[] stock2;
1262    private int xFarmacia, yFarmacia, xCidadao, yCidadao;
1263    private int minDistance = 1000;
1264    private String customerName;
1265
1266    @SuppressWarnings("unchecked")
1267    public void action() {
1268
1269      ACLMessage msg = receive();
1270      if (msg != null) {
1271
1272        //Recebe coordenadas de cada farmacia
1273        if (msg.getPerformative() == ACLMessage.SUBSCRIBE) {
1274          AID Farmacia2 = msg.getSender();
1275          String ID_Farmacia= msg.getSender().getLocalName();
1276
1277          //Cria uma nova Class_Farmacia guardando no arraylist Farmacias
1278          try {
1279            Farm a = (Farm) msg.getContentObject();
1280            Farmacias.put(ID_Farmacia,a);
1281
1282          } catch (UnreadableException e) {
1283            // TODO Auto-generated catch block
1284            e.printStackTrace();
1285          }
1286        }
1287
1288        if (msg.getPerformative() == ACLMessage.INFORM) {
1289
1290          try {
1291            HashMap<String, Farm> farmacias = (HashMap<String, Farm>) msg.
                  getContentObject();
1292
1293            farmacia_maior_lucro = "";
1294            int lucro = 0;
1295
1296            farmaciaComMaisPedidos = "";
1297            int pedido = 0;
1298
1299
1300            System.out.print(
```

```
1301                "*********************************************************\n");
1302            System.out.print(
1303                "                STOCK DAS FARMACIAS                      \n");
1304            System.out.print(
1305                "*********************************************************\n");
1306            System.out.print(
1307                "+-------------------------------------------------------+\n");
1308
1309            System.out.print(
1310                "|  FARMACIA  |  Brufen  | Ben-u-ron |  Aspirina  |  Xanax  |  Valium
                      | Fenistil | Voltaren |  Buscopan  | Leite NAN | Kompensan |
                    Rennie  | Bissolvon | Strepfen |\n");
1311
1312            for(HashMap.Entry<String, Farm> pair  : farmacias.entrySet()) {
1313              Farm f=pair.getValue();
1314
1315              HashMap<String, Integer> stock4 =  f.getStock_medicamentos(); //Obter a
                    lista dos seus stocks
1316
1317              System.out.print("+----------------------------------------+\n");
1318              System.out.print("|  "+pair.getKey().toUpperCase()+" |    " + stock4.get
                    ("brufen") + "     |     "
1319                  + stock4.get("ben-u-ron") + "     |     "
1320                  + stock4.get("aspirina") + "      |    " + stock4.get("xanax")
1321                  + "    |    " + stock4.get("valium") + "     |     " + stock4.get("
                        fenistil") + "    |    "
1322                  + stock4.get("voltaren")+ "     |     " + stock4.get("buscopan")
1323                  + "      |     " + stock4.get("leite NAN") + "     |     " + stock4.
                        get("kompensan") + "     |     "
1324                  + stock4.get("rennie") + "    |     " + stock4.get("bissolvon") + "
                            |     "
1325                  + stock4.get("strepfen") + "    |     "
1326                  + " \n");
1327              }
1328            System.out.print(
1329                "+-------------------------------------------------------+\n");
1330            System.out.print("\n");
1331            System.out.print("\n");
1332            System.out.print("\n");
1333
1334            System.out.print(
1335                "*********************************************************\n");
1336            System.out.print(
1337                "                N DE VENDAS                              \n");
1338            System.out.print(
1339                "*********************************************************\n");
1340
1341            System.out.print(
```

```
1342                "+-------------------------------------------------------+\n");
1343
1344            System.out.print(
1345                "|  FARMACIA  |  Brufen  | Ben-u-ron |  Aspirina  |  Xanax  |  Valium
                        | Fenistil | Voltaren |  Buscopan  | Leite NAN | Kompensan |
                       Rennie  | Bissolvon | Strepfen |    Total de vendas    |\n");
1346
1347            DefaultCategoryDataset dados = new DefaultCategoryDataset();
1348
1349            for(HashMap.Entry<String, Farm> pair  : farmacias.entrySet()) {
1350              Farm f=pair.getValue();
1351              String nome_farmacia= f.getAgent().getLocalName();
1352              dados.setValue(f.getProdutos_vendidos().get("brufen"), "brufen",
                      nome_farmacia);
1353              dados.setValue(f.getProdutos_vendidos().get("ben-u-ron"), "ben-u-ron",
                      nome_farmacia);
1354              dados.setValue(f.getProdutos_vendidos().get("aspirina"), "aspirina",
                      nome_farmacia);
1355              dados.setValue(f.getProdutos_vendidos().get("xanax"), "xanax",
                      nome_farmacia);
1356              dados.setValue(f.getProdutos_vendidos().get("valium"), "valium",
                      nome_farmacia);
1357              dados.setValue(f.getProdutos_vendidos().get("fenistil"), "fenistil",
                      nome_farmacia);
1358              dados.setValue(f.getProdutos_vendidos().get("voltaren"), "voltaren",
                      nome_farmacia);
1359              dados.setValue(f.getProdutos_vendidos().get("buscopan"), "buscopan",
                      nome_farmacia);
1360              dados.setValue(f.getProdutos_vendidos().get("leite NAN"), "leite NAN",
                      nome_farmacia);
1361              dados.setValue(f.getProdutos_vendidos().get("kompensan"), "kompensan",
                      nome_farmacia);
1362              dados.setValue(f.getProdutos_vendidos().get("rennie"), "rennie",
                      nome_farmacia);
1363              dados.setValue(f.getProdutos_vendidos().get("bissolvon"), "bissolvon",
                      nome_farmacia);
1364              dados.setValue(f.getProdutos_vendidos().get("strepfen"), "strepfen",
                      nome_farmacia);
1365
1366              HashMap<String, Integer> produtos_vendidos =  f.getProdutos_vendidos();
                      //Obter a lista dos seus stocks
1367              int soma = 0;
1368              for(HashMap.Entry<String, Integer> conjunto  : produtos_vendidos.
                      entrySet()) {
1369                soma += conjunto.getValue();
1370              }
1371
1372              System.out.print(
```

```
1373                "+-------------------------------------------------------+\n");
1374            System.out.print("|  "+pair.getKey().toUpperCase()+" |    " +
                    produtos_vendidos.get("brufen") + "     |     "
1375                + produtos_vendidos.get("ben-u-ron") + "     |     "
1376                + produtos_vendidos.get("aspirina") + "      |    " +
                        produtos_vendidos.get("xanax")
1377                + "    |    " + produtos_vendidos.get("valium") + "     |     " +
                        produtos_vendidos.get("fenistil") + "    |    "
1378                + produtos_vendidos.get("voltaren")+ "     |     " +
                        produtos_vendidos.get("buscopan")
1379                + "      |     " + produtos_vendidos.get("leite NAN") + "     |
                        " + produtos_vendidos.get("kompensan") + "     |     "
1380                + produtos_vendidos.get("rennie") + "    |     " + produtos_vendidos
                        .get("bissolvon") + "     |     "
1381                + produtos_vendidos.get("strepfen") + "    |          " + soma + "
                                |    "                                    + " \n");
1382            }
1383        JFreeChart chart = ChartFactory.createBarChart(" Vendas por Farmacia ", "
                Farmacias", "Vendas", dados, PlotOrientation.VERTICAL, true, true,
                false);
1384        ChartFrame frame1 = new ChartFrame("Grafico de barras", chart);
1385        frame1.setVisible(true);
1386        frame1.setBounds(300, 0, 1000, 450);
1387
1388        System.out.print(
1389            "+-------------------------------------------------------------+\n");
1390        System.out.print("\n");
1391        System.out.print("\n");
1392        System.out.print("\n");
1393        System.out.print(
1394            "***************************************************************\n");
1395        System.out.print(
1396            "           Historico de Pacientes por Farmacia              \n");
1397        System.out.print(
1398            "***************************************************************\n");
1399        System.out.print(
1400            "+-------------------------------------------------------------+\n");
1401        System.out.print(
1402            "|  FARMACIA  |  Cidadao0  | Cidadao1 |  Cidadao2  |  Cidadao3  |
                    Cidadao4  | Cidadao5 | Cidadao6 |  Cidadao7  | Cidadao8 | Cidadao9
                     |\n");
1403
1404        DefaultCategoryDataset dados3 = new DefaultCategoryDataset();
1405        for(HashMap.Entry<String, Farm> pair  : farmacias.entrySet()) {
1406          Farm f=pair.getValue();
1407          HashMap<String, Integer> farm_cidadao =  f.getCidadaos(); //Obter a
                    lista dos seus stocks
1408          String nome_farmacia= f.getAgent().getLocalName();
```

```
1409                int max = 0;
1410                String cidadao = "";
1411                for(HashMap.Entry<String, Integer> conjunto  : farm_cidadao.entrySet())
                        {
1412                  int cidadao_vend = conjunto.getValue();
1413                  if (cidadao_vend>max){
1414                    cidadao = conjunto.getKey();
1415                    max = cidadao_vend;
1416                  }
1417                }
1418                if (max!=0) {
1419                  dados3.setValue(max, nome_farmacia, cidadao);
1420                }
1421                System.out.print(
1422                    "+---------------------------------------------------------+\n");
1423                System.out.print("|  "+pair.getKey().toUpperCase()+" |     " +
                    farm_cidadao.get("Cidadao0") + "      |     "
1424                    + farm_cidadao.get("Cidadao1") + "    |     "
1425                    + farm_cidadao.get("Cidadao2") + "      |      " + farm_cidadao.get(
                        "Cidadao3")
1426                    + "     |      " + farm_cidadao.get("Cidadao4") + "     |     " +
                        farm_cidadao.get("Cidadao5") + "    |    "
1427                    + farm_cidadao.get("Cidadao6")+ "     |     " + farm_cidadao.get("
                        Cidadao7")
1428                    + "      |     " + farm_cidadao.get("Cidadao8") + "    |     " +
                        farm_cidadao.get("Cidadao9") + "    |     " + "\n");
1429                }
1430
1431            JFreeChart chart3 = ChartFactory.createBarChart(" Cliente mais requisitado
                     por Farmacia ", "Farmacias", "Vendas", dados3, PlotOrientation.
                    VERTICAL, true, true, false);
1432            ChartFrame frame3 = new ChartFrame("Grafico de barras", chart3);
1433            frame3.setVisible(true);
1434            frame3.setBounds(1000, 500, 450, 450);
1435
1436            System.out.print("+-----------------------------------------+\n");
1437            System.out.print("\n");
1438            System.out.print("\n");
1439            System.out.print("\n");
1440            System.out.print(
1441                "|  FARMACIA  |   Lucro   |  Pedidos  |");
1442            System.out.print(               "\n");
1443            System.out.print(
1444                "+-----------------------------------+\n");
1445
1446            DefaultPieDataset dados2 = new DefaultPieDataset();
1447
1448            for(HashMap.Entry<String, Farm> pair  : farmacias.entrySet()) {
```

```
1449                Farm f=pair.getValue();
1450                int lucro2 = f.getReceita() - f.getDespesa();
1451
1452                int pedido2= f.getPedidos();
1453
1454                String nome_farmacia= f.getAgent().getLocalName();
1455                if (lucro2>0) {
1456                  dados2.setValue(nome_farmacia, lucro2);
1457                }
1458                System.out.print("|  "+pair.getKey().toUpperCase()+" |    " + lucro2 + "
                           |      " + pedido2 + "      |");
1459                System.out.print(                                "\n");
1460                if (lucro2 == lucro) {
1461                  farmacia_maior_lucro+=" e ";
1462                  farmacia_maior_lucro+=pair.getKey();
1463                }
1464
1465                if (lucro2 > lucro) {
1466                  lucro=lucro2;
1467                  farmacia_maior_lucro=pair.getKey();
1468                }
1469
1470                if (pedido2 == pedido) {
1471                  farmaciaComMaisPedidos+=" e ";
1472                  farmaciaComMaisPedidos+=pair.getKey();
1473                }
1474
1475                if (pedido2 > pedido) {
1476                  pedido=pedido2;
1477                  farmaciaComMaisPedidos=pair.getKey();
1478                }
1479              }
1480              JFreeChart chart2 = ChartFactory.createPieChart(" Lucro por Farmacia ",
                      dados2, true, true, false);
1481              PiePlot plot = (PiePlot) chart2.getPlot();
1482
1483                  plot.setSimpleLabels(true);
1484                  PieSectionLabelGenerator gen = new StandardPieSectionLabelGenerator(
1485                          "{0}: {1}E ({2})", new DecimalFormat("0"), new DecimalFormat("
                              0%"));
1486                      plot.setLabelGenerator(gen);
1487
1488              ChartFrame frame2 = new ChartFrame("Grafico circular", chart2);
1489              frame2.setVisible(true);
1490
1491              frame2.setBounds(0, 500, 450, 450);
1492
1493              System.out.print("+-----------------------------------+\n");
```

```
1494              System.out.print("\n");
1495              System.out.print("\n");
1496              System.out.print("\n");
1497
1498              System.out.println("A farmacia com maior lucro e: " + farmacia_maior_lucro
                       + " com " + lucro + " euros!");
1499
1500              System.out.println("A farmacia com maior numero de pedidos e: " +
                      farmaciaComMaisPedidos + " com " + pedido + " pedido(s)!");
1501              System.out.print("\n");
1502            }
1503            catch (UnreadableException e) {
1504              // TODO Auto-generated catch block
1505              e.printStackTrace();
1506            }
1507          }
1508        }
1509        else {
1510          block();
1511        }
1512      }
1513    }
1514  }
```

G. Classe Farm

```
1517  package Classes;
1518
1519  import java.util.HashMap;
1520
1521  import jade.core.AID;
1522
1523  public class Farm implements java.io.Serializable{
1524
1525    private AID agent;
1526    private Position position;
1527    private HashMap<String, Integer> stock_medicamentos = new HashMap<>();
1528    private HashMap<String, Integer> preco_medicamentos = new HashMap<>();
1529    private HashMap<String, Integer> produtos_vendidos = new HashMap<>();
1530    private HashMap<String, Integer> farm_cidadao = new HashMap<>();
1531    int receita;
1532    int despesa;
1533    int pedidos;
1534
1535    public Farm(AID agent, Position position) {
1536
1537      this.agent = agent;
```

```
1538        this.position = position;
1539
1540        stock_medicamentos.put("brufen", 9); stock_medicamentos.put("ben-u-ron", 9);
                stock_medicamentos.put("aspirina", 9); stock_medicamentos.put("xanax", 9);
1541        stock_medicamentos.put("valium", 9); stock_medicamentos.put("fenistil", 9);
                stock_medicamentos.put("voltaren", 9); stock_medicamentos.put("buscopan", 9);
1542        stock_medicamentos.put("leite NAN", 9);stock_medicamentos.put("kompensan", 9);
                stock_medicamentos.put("rennie", 9);stock_medicamentos.put("bissolvon", 9);
1543        stock_medicamentos.put("strepfen", 9);
1544
1545        preco_medicamentos.put("brufen", 10); preco_medicamentos.put("ben-u-ron", 12);
                preco_medicamentos.put("aspirina", 8); preco_medicamentos.put("xanax", 10);
1546        preco_medicamentos.put("valium", 11); preco_medicamentos.put("fenistil", 15);
                preco_medicamentos.put("voltaren", 13); preco_medicamentos.put("buscopan", 8);
1547        preco_medicamentos.put("leite NAN", 10);preco_medicamentos.put("kompensan", 10);
                preco_medicamentos.put("rennie", 15);preco_medicamentos.put("bissolvon", 10);
1548        preco_medicamentos.put("strepfen", 12);
1549
1550        produtos_vendidos.put("brufen", 0); produtos_vendidos.put("ben-u-ron", 0);
                produtos_vendidos.put("aspirina", 0); produtos_vendidos.put("xanax", 0);
1551        produtos_vendidos.put("valium", 0); produtos_vendidos.put("fenistil", 0);
                produtos_vendidos.put("voltaren", 0); produtos_vendidos.put("buscopan", 0);
1552        produtos_vendidos.put("leite NAN", 0);produtos_vendidos.put("kompensan", 0);
                produtos_vendidos.put("rennie", 0);produtos_vendidos.put("bissolvon", 0);
1553        produtos_vendidos.put("strepfen", 0);
1554
1555        farm_cidadao.put("Cidadao0", 0); farm_cidadao.put("Cidadao1", 0); farm_cidadao.put
                ("Cidadao2", 0); farm_cidadao.put("Cidadao3", 0);
1556        farm_cidadao.put("Cidadao4", 0); farm_cidadao.put("Cidadao5", 0); farm_cidadao.put
                ("Cidadao6", 0); farm_cidadao.put("Cidadao7", 0);
1557        farm_cidadao.put("Cidadao8", 0);farm_cidadao.put("Cidadao9", 0);
1558
1559        receita=0;
1560        despesa=0;
1561        pedidos=0;
1562    }
1563
1564    public HashMap<String, Integer> getCidadaos() {
1565        return farm_cidadao;
1566    }
1567    public void setCidadaos(HashMap<String, Integer> farm_cidadao) {
1568        this.farm_cidadao = farm_cidadao;
1569    }
1570    public HashMap<String, Integer> getPreco_medicamentos() {
1571        return preco_medicamentos;
1572    }
1573    public void setPreco_medicamentos(HashMap<String, Integer> preco_medicamentos) {
1574        this.preco_medicamentos = preco_medicamentos;
```

```
1575    }
1576    public int getReceita() {
1577      return receita;
1578    }
1579    public void setReceita(int valor) {
1580      this.receita += valor;
1581    }
1582    public int getDespesa() {
1583      return despesa;
1584    }
1585    public void setDespesa(int valor) {
1586      this.despesa += valor;
1587    }
1588    public int getPedidos() {
1589      return pedidos;
1590    }
1591    public void setPedidos() {
1592      this.pedidos += 1;
1593    }
1594    public Position getPosition() {
1595      return position;
1596    }
1597    public void setPosition(Position position) {
1598      this.position = position;
1599    }
1600    public AID getAgent() {
1601      return agent;
1602    }
1603    public HashMap<String, Integer> getProdutos_vendidos() {
1604      return produtos_vendidos;
1605    }
1606    public void setProdutosVendidos(HashMap<String, Integer> produtos_vendidos) {
1607      this.produtos_vendidos = produtos_vendidos;
1608    }
1609    public HashMap<String, Integer> getStock_medicamentos() {
1610      return stock_medicamentos;
1611    }
1612    public void setStock_medicamentos(HashMap<String, Integer> stock_medicamentos) {
1613      this.stock_medicamentos = stock_medicamentos;
1614    }
1615    public void alterarStock(int stock, String medicamento){
1616         stock_medicamentos.put(medicamento, stock);
1617    }
1618    public void alterarVendido(int quantidade, String medicamento){
1619         produtos_vendidos.put(medicamento, quantidade);
1620    }
1621  }
```

H. Classe Forn

```
1623 package Classes;
1624
1625 import java.util.HashMap;
1626 import jade.core.AID;
1627
1628 public class Forn implements java.io.Serializable{
1629   private AID agent;
1630   private HashMap<String, Integer> med_preco = new HashMap<>();
1631
1632   public Forn(AID agent) {
1633
1634     this.agent = agent;
1635   }
1636   public AID getAgent() {
1637     return agent;
1638   }
1639   public HashMap<String, Integer> getMed_preco() {
1640     return med_preco;
1641   }
1642   public void setSMed_preco(HashMap<String, Integer> med_preco) {
1643     this.med_preco = med_preco;
1644   }
1645   public Integer buscar_preco_med(String medicamento){
1646         return med_preco.get(medicamento);
1647   }
1648 }
```

I. Classe Position

```
1650 package Classes;
1651
1652 public class Position implements java.io.Serializable {
1653
1654   private int x,y;
1655
1656   public Position(int x, int y) {
1657     super();
1658     this.x = x;
1659     this.y = y;
1660   }
1661   public int getX() {
1662     return x;
1663   }
1664   public void setX(int x) {
1665     this.x = x;
1666   }
```

```
1667    public int getY() {
1668      return y;
1669    }
1670    public void setY(int y) {
1671      this.y = y;
1672    }
1673  }
```