Universidad Carlos III de Madrid
Escuela Politécnica Superior

“Buscador de Información de Restaurantes mediante un Sistema Multiagente”

Proyecto Fin de Carrera

Ingeniería Técnica en Informática de Gestión

Autor Cristina Casado Corral
Director D. Daniel Borrajo Millán
Año 2015

Índice

Capítulo 1. Introducción

El proyecto desarrollado es un ejemplo de funcionamiento de un Sistema Multiagente (SMA), el cual ha sido implementado utilizando la herramienta JADE.

El SMA implementado es un buscador de información de restaurantes, es decir, tiene como objeto atender las peticiones que recibe de los usuarios, en las cuales, solicitan información de un restaurante. Para ello, todos los agentes del SMA trabajarán de forma colaborativa para dar una respuesta que satisfaga la demanda del usuario. El SMA consultará en sitios WEB o en un fichero con datos almacenados de anteriores búsquedas.

Hay que mencionar, que este proyecto, en su origen formaba parte de un proyecto común a realizar con varios compañeros. Por ello, el *wrapper* utilizado para extraer la información de la WEB, no ha sido desarrollado por mí, sino que es código reutilizado de este proyecto que no finalizó.

Por otra parte, la ontología utilizada vino dada, ya que al comienzo de este proyecto se estaba trabajando en un proyecto más amplio, cuyo objetivo era la planificación turística en ciudades.

Capítulo 2. Estado de la Cuestión

Este capítulo trata de acercarnos a los conceptos de Agente y Sistemas Multiagente. Una vez aclarados estos conceptos, se explican los lenguajes de comunicación entre agentes más relevantes.

2.1. Agentes

¿Qué es un Agente? Los siguientes apartados tratan de dar una respuesta a esta pregunta.

2.1.1. *¿Qué es un Agente?*

No hay una definición formal unificada de lo que es un agente software. En lo que sí parece haber cierto consenso, es en aceptar la autonomía como aspecto clave. Una selección de las posibles definiciones desde el punto de vista de la Inteligencia Artificial, podría ser la siguiente:

- [Maes, 1995] Los agentes autónomos son sistemas computacionales que habitan en un entorno complejo y dinámico, perciben los cambios y actúan autónomamente en dicho entorno, realizando un conjunto de metas o tareas para las cuales fueron diseñados.

- [Wooldridge, 2002] Un agente es un sistema informático situado en un entorno que es capaz de realizar acciones de forma autónoma para conseguir sus objetivos de diseño.

- [Russell & Norvig, 1995] Un agente es cualquier entidad que percibe su entorno a través de sensores y actúa sobre ese entorno mediante efectores.

- [Franklin, 1996] Un agente autónomo es un sistema situado en, y parte de un entorno, que siente ese entorno y actúa sobre él, a través del

tiempo, persiguiendo sus propios objetivos de forma que afecte lo que siente en el futuro.

2.1.2. *Características de los Agentes*

Un agente debe presentar como mínimo un conjunto de características. Un agente debe ser [Bellifemine, 2007]:

- **Autónomo**, ya que funciona sin la intervención directa de los seres humanos u otras personas y tiene control sobre sus acciones y estado interno.
- **Social**, porque coopera con los seres humanos u otros agentes con el fin de lograr sus tareas.
- **Reactivo**, responde de manera oportuna a los cambios que se producen en el medio ambiente.
- **Proactivo**, ya que no actúa simplemente en respuesta a su entorno, sino que es capaz de mostrar un comportamiento orientado a cumplir los objetivos.

Por otra parte, si es necesario un agente puede ser [Bellifemine, 2007]:

- **Móvil**, con la capacidad de viajar entre distintos nodos de una red informática.
- **Veraz**, proporcionando la certeza de que no comunica deliberadamente información falsa.
- **Benevolente**, siempre tratando de llevar a cabo lo que se pide de él.
- **Racional**, actuando siempre con el fin de alcanzar sus metas y nunca para evitar que sus objetivos se cumplan.
- **Capaz de aprender**, para adaptarse a su entorno y a los deseos de sus usuarios.

2.1.3. *Clasificación de Agentes*

Al igual que ocurre con la definición de agente, no hay una única clasificación posible de los mismos. Atendiendo a diferentes criterios se pueden establecer diversas clasificaciones, entre ellas las siguientes:

- En función de la capacidad de resolver problemas se clasifican en:
  - **Agentes Reactivos**. Realizan tareas sencillas. Reaccionan ante cambios producidos en su entorno o a mensajes provenientes de otros agentes. No son capaces de razonar acerca de sus intenciones. Sus acciones se realizan como resultado de reglas establecidas.
  - **Agentes Cognitivos**. Realizan tareas complejas. Utilizan algún tipo de representación explícita (simbólica) del conocimiento. Para realizar las tareas necesitan llevar a cabo procesos de razonamiento y otros procesos cognitivos como la planificación y el aprendizaje.
- Según su movilidad se clasifican en:
  - **Agentes Fijos**. Permanecen en el lugar donde fueron creados. Los agentes cognitivos suelen ser agentes fijos debido a su complejidad interna.
  - **Agentes Móviles**. Aquellos que poseen la capacidad de desplazarse por los nodos de una red con la finalidad de realizar las tareas necesarias para cumplir con sus objetivos de diseño.
- En función de la tarea que desempeñan pueden ser:
  - **Agentes Colaborativos**. Enfatizan su autonomía y cooperación con otros agentes para llevar a cabo tareas para sus propietarios.
  - **Agentes de Interfaz**. Asistentes personales que colaboran con el usuario en un ambiente dado. Proporcionan asistencia al usuario sugiriéndole otras formas de realizar las acciones.
  - **Agentes de Entretenimiento**. Agentes que proveen al usuario de algún tipo de entretenimiento.
  - **Agentes de Información**. Los agentes de Información, los podemos clasificar a su vez según la función que realizan [Carrascosa, 2001]:
    - Agentes de Búsqueda. Buscan, recuperan y proporcionan la información como si fueran auténticos gestores de información y documentación.
    - Agentes de Filtrado. Mediante el borrado de datos no deseado, se usan para reducir la sobreabundancia de información.
    - Agentes de Monitorización. Proporcionan al usuario la información cuando sucede un determinado acontecimiento;

por ejemplo, cuando la información se actualiza, se traslada de lugar o se borra.

- **Agentes Híbridos**. son aquellos que en su funcionamiento poseen la combinación de dos o más de las capacidades de los tipos mencionados.

2.2. Arquitecturas de Agentes

La arquitectura de un agente es esencialmente un mapa de la parte interna de un agente: sus estructuras de datos, las operaciones que pueden realizarse en estas estructuras de datos, y el flujo de control entre estas estructuras de datos [Wooldridge, 2000]

Atendiendo al tipo de procesamiento empleado, se pueden distinguir tres tipos de arquitecturas:

- **Arquitecturas deliberativas**. Utilizan modelos de representación simbólica del conocimiento. Los agentes parten de un estado inicial y son capaces de generar planes para lograr sus objetivos. Las decisiones se toman usando mecanismos de razonamiento lógico. Un ejemplo de arquitectura deliberativa es la arquitectura BDI (Belief-Desire-Intention). En esta arquitectura los agentes se caracterizan por estar dotados de los siguientes estados mentales:
  - Creencias: Conocimiento que el agente tiene sobre sí mismo y su entorno.
  - Deseos: Los objetivos que posee el agente a largo plazo
  - Intenciones: Expresan los objetivos que en cada momento intenta cumplir el agente. Se puede introducir también el concepto de plan, que permite definir las intenciones como los planes que un agente está realizando en un momento dado.
- **Arquitecturas reactivas**. Se caracterizan por no tener un modelo simbólico como elemento de razonamiento. Ofrecen respuestas inmediatas a los estímulos que perciben en el entorno. A partir de una percepción concreta se disparan las acciones pertinentes Este tipo de agentes puede implementarse usando una infraestructura de comunicación a través de mensajes ACL (Agent Communication Language).

- **Arquitecturas híbridas**. Combinan aspectos de los dos modelos mencionados anteriormente. Por una parte toman la capacidad de realizar razonamientos y tomar decisiones de las arquitecturas deliberativas y por otra parte la capacidad de reaccionar de forma inmediata ante estímulos de las arquitecturas reactivas.

2.3. Sistema Multiagente

Un Sistema Multiagente (SMA) es un sistema que consiste en un conjunto de agentes en el que interactúan los unos con los otros, normalmente mediante el intercambio de mensajes a través de alguna infraestructura de red. En el caso más general, los agentes en un SMA estarán representando o actuando en nombre de los usuarios o los propietarios con muy diferentes objetivos y motivaciones. Con el fin de interactuar con éxito, estos agentes requerirán por lo tanto la capacidad de cooperar, coordinar y negociar con los demás, de la misma manera que cooperamos, coordinamos y negociamos con otras personas en nuestra vida cotidiana. [Wooldridge, 2002].

Los SMA permiten la integración de agentes existentes dentro de un gran sistema. Esto nos ofrece la posibilidad de que para solucionar un problema no sea necesario definir un nuevo agente especializado, sino que mediante la combinación de diferentes agentes dentro del sistema y el trabajo común de dichos agentes se logre alcanzar el objetivo deseado.

En un SMA un agente debe ser capaz de intuir las acciones de los otros agentes involucrados en el sistema que afectan a su propia tarea de planificación y saber influir en las acciones de los otros agentes en beneficio de sus propios objetivos. En algunos casos, los efectos de las acciones de los otros agentes sobre el propio agente pueden suponer un obstáculo para que el agente alcance su objetivo. Para influir sobre lo que hacen los demás agentes, se necesitan métodos para que los agentes puedan comunicarse entre ellos.

La comunicación entre los agentes se puede producir de forma indirecta mediante el sistema de Pizarra o de forma directa utilizando el sistema de mensajes o diálogos:

- **Sistema de Pizarra**. Es la arquitectura básica. La arquitectura de pizarra es muy utilizada en Inteligencia Artificial. La pizarra es una estructura de datos que es usada como mecanismo general de

comunicación entre las múltiples fuentes de conocimiento y es gestionada y arbitrada por un controlador. Es decir, es un área de trabajo común a todos los agentes pertenecientes al SMA, donde los agentes intercambian información, datos y conocimiento. El intercambio de información, datos y conocimiento no se realiza mediante una comunicación directa entre los agentes, sino que cada agente escribe en la pizarra la información que desea y a su vez puede acceder a toda la información contenida.

- **Sistema de Mensajes/Dialogo**. Este sistema es el más utilizado actualmente en la comunicación entre agentes. En la comunicación mediante mensajes la comunicación se realiza directamente entre dos agentes. Un agente emisor envía un mensaje específico a un agente receptor. Para que la comunicación mediante mensajes ayude en la implementación de estrategias de cooperación, es necesario definir un protocolo de comunicaciones que especifique el proceso de comunicación, el formato de los mensajes y el lenguaje de comunicación, por lo que todos los agentes que se comuniquen entre sí deben conocer la semántica del lenguaje de comunicación.

2.4. Lenguajes de Comunicación

Los dos lenguajes de comunicación entre agentes más relevantes son KQML (Knowledge Query and Manipulation Language), el cuál es la base de un ACL, y FIPA ACL (FIPA Agent Communication Language) el más utilizado actualmente. Ambos lenguajes de comunicación están basados en la Teoría de los Actos del Habla que asigna una intención a cada acto comunicativo, es decir, con cada acto comunicativo se quiere transmitir una creencia, una intención o un deseo con la intención de provocar ciertos efectos sobre el conocimiento del oyente.

2.4.1. *KQML*

El External Interfaces Group fue fundado en 1990 como parte de DARPA (Defense Advanced Research Projects Agency) Knowledge Sharing Effort con el objetivo de desarrollar un protocolo para el intercambio de conocimiento entre

sistemas de información autónomos. El principal resultado de este esfuerzo es KQML, Knowledge Query and Manipulation Language [KQML 1993].

KQML nos proporciona una sintaxis estándar y unas recomendaciones de cómo los agentes pueden comunicarse entre sí.

Un mensaje KQML tiene una performativa que nos indica que intención se quiere comunicar y una serie de parámetros asociados a dicha performativa definidos como pares de atributos-valor.

```
(KMQL-performative
        :sender <valor>
        :receiver <valor>
        :languaje <valor>
        :ontology <valor>
        :content <valor>
        :reply-with <valor>
        :in reply-to<valor>
        .....
)
```

A continuación se muestran algunas de las performativas más relevantes, clasificadas según la intención que pretenden comunicar:

- **de información genérica**: tell, deny, untell.
- **de base de datos**: insert, delete, delete-one, delete-all.
- **de respuestas básicas**: error, sorry.
- **de preguntas básicas**: evaluate, reply, ask-if, ask-about, ask-one, ask-all.
- **de múltiple respuestas**: stream-about, stream-all, eos.
- **de peticiones de realización**: achieve, unachieve.
- **de control de flujo**: standby, ready, next, rest, discard, generator.
- **de definición de capacidades**: advertise.
- **de notificación**: subscribe, monitor.
- **de red**: register, unregister, forward, broadcast, pipe, break, transport-address.

- **de facilitación**: broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all.

Los parámetros, como ya se ha indicado anteriormente, están compuestos por una pareja de atributo-valor. Existe un conjunto de parámetros reservados, en el sentido de que cuando una performativa haga uso de ellos, el uso de ese atributo debe ser consistente con la definición que tiene asociada. El conjunto de parámetros reservados se detalla en la siguiente tabla.

PARÁMETRO	**DEFINICIÓN**
:sender	Nombre del emisor del mensaje
:receiver	El receptor del mensaje
:language	Lenguaje utilizado en el parámetro *:content*
:ontology	El nombre de la ontología utilizada en el parámetro *:content*
:content	Información asociada a la actitud expresada por la performativa
:in-reply-to	Etiqueta para la respuesta a un mensaje anterior
:reply-with	Si el emisor espera respuesta, la etiqueta para dicha respuesta

Tabla 2.1 Parámetros mensaje KQML

La asimilación de KQML por la comunidad de sistemas multiagente fue significativa, y varias implementaciones basadas en KQML fueron desarrolladas y distribuidas. A pesar de este éxito, KQML fue posteriormente criticado por una serie de razones [Wooldridge 2000]. Por no contar con un conjunto de performativas bien restringido, agentes de diferentes sistemas pueden adoptar dialectos diferentes, impidiendo la interoperabilidad. Otro problema es que no posee una semántica definida formalmente, de manera que no garantiza que las performativas sean correctamente interpretadas por todos los agentes. Además, el lenguaje no tiene performativas que permitan establecer compromisos entre los agentes, lo cual puede ser importante para la coordinación de sus acciones. Además, el lenguaje carece de performativas que permitan establecer compromisos entre los agentes, lo cual es importante si los agentes deben coordinar las acciones entre sí. Estas críticas, entre otros,

condujeron al desarrollo de un lenguaje nuevo, pero más bien estrechamente relacionado por el consorcio de la FIPA (Foundation for Intelligent Physical Agents)

2.4.2. *FIPA*

La FIPA fue formada originalmente en 1996 en Suiza como una organización para establecer los estándares de especificaciones software para generar agentes heterogéneos y la interacción entre agentes y sistemas basados en agentes [FIPA].

A lo largo de este punto se harán referencias a los documentos de especificaciones FIPA que apliquen.

2.4.2.1 Mensaje FIPA-ACL

La sintaxis de un mensaje FIPA-ACL es muy parecida a la de un mensaje KQML.

```
(FIPA-ACL-performative
        :sender <valor>
        :receiver <valor>
        :languaje <valor>
        :ontology <valor>
        :content <valor>
        :reply-with <valor>
        :in reply-to<valor>
        :conversation-id<valor>
        .....
)
```

Un mensaje de FIPA ACL contiene un conjunto de uno o más parámetros de mensaje. Los parámetros necesarios para la comunicación eficaz del agente variará según la situación; el único parámetro que es obligatorio en todos los mensajes ACL es *performative*, aunque se espera que la mayoría de los mensajes ACL también contengan los parámetros *sender*, *receiver* y *content* [FIPA00061].

PARÁMETRO	DEFINICIÓN	
Performative	Tipo de acto comunicativo	
sender	Identificador del emisor del mensaje	Participantes en la comunicación
Receiver	Identificador del receptor del mensaje	
reply-to	A qué agente hay que enviar los mensajes posteriores dentro de una conversación	
content	Contenido del mensaje	
language	Lenguaje utilizado en el contenido del mensaje	Descripción del contenido
encoding	Codificación específica del contenido del mensaje	
ontology	Referencia a una ontología para dar significado a los símbolos en el contenido del mensaje	
protocol	Protocolo de interacción utilizado para estructurar una conversación	Control de la conversación
conversation-id	Identificador único del hilo de una conversación	
reply-with	Expresión que debe ser utilizada en el parámetro *in-reply-to del mensaje de respuesta*	
in-reply-to	Expresión indicada en el mensaje previo en el parámetro reply-with	
reply-by	Tiempo en el que la respuesta debe ser recibida	

Tabla 2.2 Parámetros mensaje FIPA-ACL

2.4.2.2 Actos Comunicativos

FIPA ha definido y estandarizado una librería de actos comunicativos (communicative acts o abreviadamente CAs). El nombre asignado a un acto comunicativo debe de identificarse de manera inequívoca dentro del mensaje FIPA-ACL, debe ser una palabra o abreviatura en ingles que sugiera la semántica del acto comunicativo [FIPA00037].

La lista de los actos comunicativos definidos por FIPA-ACL se detalla a continuación:

- **Accept Proposal**. La acción de aceptar una propuesta recibida previamente para realizar una acción.
- **Agree**. La acción de ponerse de acuerdo para llevar a cabo algún tipo de acción, posiblemente en el futuro.
- **Cancel**. La acción de un agente de informar a otro agente de que ya no tiene la intención de que el segundo agente realice alguna acción.
- **Call for Proposal (*cfp*)**. Es una acción de propósito general para iniciar un proceso de negociación haciendo una llamada a la participación para llevar a cabo una acción dada.
- **Confirm.** El emisor informa al receptor de que una proposición dada es verdadera, aun sabiendo que el receptor duda sobre dicha proposición.
- **Disconfirm**. El emisor informa al receptor de que una proposición dada es falsa, aun sabiendo que el receptor cree probable que la proposición sea verdadera.
- **Failure**. La acción de decirle a otro agente que se intentó realizar una acción, pero el intento fue fallido.
- **Inform**. El emisor informa al receptor de que una proposición dada es verdadera.
- **Inform If**. Es una acción en la que el emisor solicita al receptor que le informe si una proposición es verdadera.
- **Inform Ref**. Una acción del emisor para informar al receptor sobre el objeto que corresponde a un descriptor, por ejemplo, un nombre.
- **Not Understood**. El emisor informa al receptor de que percibió que realizó alguna acción, pero que no comprendía lo que acababa de hacer. Un caso muy frecuente es que el emisor le diga al receptor que no entendió el mensaje que recibió.
- **Propagate**. El emisor tiene la intención de que el receptor trate el mensaje embebido cómo si se lo hubiera enviado directamente a él, y que además identifique a los agentes indicados en el descriptor dado y les reenvíe el mensaje a ellos.

- **Propose**. La acción de presentar una propuesta para llevar a cabo una determinada acción, dadas ciertas condiciones previas.
- **Proxy**. El emisor quiere que el receptor identifique a los agentes que cumplan con el descriptor dado, y les reenvíe el mensaje embebido.
- **Query If**. La acción de preguntar a otro agente si una proposición dada es verdadera.
- **Query Ref**. La acción de preguntar a otro agente por un objeto referenciado por una expresión.
- **Refuse**. La acción de negarse a realizar una acción determinada, explicando la razón de la negativa.
- **Reject Proposal**. La acción de rechazar una propuesta para llevar a cabo algún tipo de acción durante una negociación.
- **Request**. El emisor solicita al receptor que lleve a cabo alguna acción. Un uso importante de esta acción es solicitar al receptor que lleve a cabo otro acto comunicativo.
- **Request When**. El emisor quiere que el receptor lleve a cabo una acción, cuando una proposición dada se cumpla.
- **Request Whenever**. El emisor quiere que el receptor lleve a cabo una acción tan pronto como se cumpla la proposición dada y, posteriormente, cada vez que la proposición se vuelva a cumplir.
- **Subscribe**. La acción de solicitar una intención persistente para que el receptor informe al emisor del valor de una referencia, y lo notifique de nuevo cada vez que el objeto identificado por la referencia cambie.

2.4.2.3 Lenguajes de Contenido

FIPA facilita también una librería de Lenguajes de Contenido que se detalla a continuación:

- **FIPA-SL** (Semantic Language). Se utiliza para definir la semántica de la intención de los CAs cómo una lógica de actitudes y acciones mentales, formalizadas en un lenguaje modal. Las expresiones de contenido se definen en términos representados como fórmulas bien formadas que consisten en términos (constante, conjunto, secuencia, término funcional, expresión y acción) y constantes (constantes numéricas, de

cadena, de fecha y hora). Es el lenguaje más general de todos (ver *FIPA SL Content Language Specification - FIPA00008*).

- **FIPA-CCL** (Constrant Choice Language). Define una semántica que permite especificar predicados con restricciones (ver *FIPA CCL Content Language Specification - FIPA00009*).
- **FIPA-KIF** (Knowledge Interchange Format). Permite expresar objetos como términos y proposiciones como oraciones. Utiliza una semántica declarativa, es completamente lógico y asegura la representación del conocimiento sobre el conocimiento (ver *FIPA KIF Content Language Specification - FIPA00010*).
- **FIPA-RDF** (Resource Description Framework). Expresa los objetos, proposiciones y las acciones que realizan los objetos mediante la definición de esquemas (ver *FIPA RDF Content Language Specification - FIPA00011*).

2.4.2.4 Protocolos de Comunicación

Las conversaciones en curso entre los agentes a menudo siguen pautas típicas. En tales casos, ciertas secuencias de mensajes son esperadas, y en cualquier punto de la conversación se espera que otros mensajes prosigan. A estos patrones típicos de intercambio de mensajes se les denomina protocolos de comunicación o interacción (*Interaction Protocols* o abreviadamente *IPs*). FIPA ha definido en sus especificaciones una serie de protocolos que recogen las conversaciones más comunes que se realizan entre agentes:

- **Request**. A un agente se le pide que realice cierta acción (ver *FIPA Request Interaction Protocol Specification - FIPA 00026*).
- **Request when**. A un agente se le pide que realice cierta acción siempre que se cumpla la precondición (ver *FIPA Request When Interaction Protocol Specification - FIPA 00028*).
- **Query**. A un agente se le pide que informe sobre algo (ver *FIPA Query Interaction Protocol Specification - FIPA 00027*).
- **Contract net**. Un agente pide la realización de cierta tarea a un conjunto de agentes. Estos dan su propuesta basada en unos costes y el iniciador elige quién la realiza finalmente (ver *FIPA Contract Net Interaction Protocol Specification - FIPA 00029*).

- **Iterated Contract Net**. Es una ampliación del protocolo anterior. Si rechaza una oferta, puede volver a lanzar la petición (ver *FIPA Iterated Contract Net Interaction Protocol Specification - FIPA 00030*).
- **Brokering**. Un agente (broker) ofrece las funcionalidades de otros agentes o reenvía las peticiones al agente apropiado (ver *FIPA Brokering Interaction Protocol Specification - FIPA 00033*).
- **Recruiting**. Es como el brokering, pero las respuestas sobre el servicio van directamente al agente que lo necesita (no a través del broker) (ver *FIPA Recruiting Interaction Protocol Specification - FIPA 00034).*
- **Propose**. El iniciador propone a una serie de agentes la realización de una tarea y estos aceptan o no (ver *FIPA Propose Interaction Protocol Specification - FIPA 00036*).
- **Subscribe**. Un agente pide ser notificado si cierta condición se vuelve verdadera (ver *FIPA Subscribe Interaction Protocol Specification - FIPA 00035*).

2.4.2.5 Plataforma de Agentes

Además de todas las especificaciones indicadas anteriormente, FIPA se ha preocupado también de definir las características que deben cumplir las plataformas de gestión de Sistemas Multiagentes. FIPA especifica solo el comportamiento externo, es decir la interfaz, y deja a cargo del desarrollador las decisiones de diseño. FIPA establece también el modelo de referencia lógico para la creación, el registro, la localización, la comunicación, la migración y la destrucción de los agentes.

La figura 2.1 muestra el conjunto de componentes de una Plataforma de Agentes (AP) FIPA [FIPA00023].

Las entidades que componen este modelo de referencia son:

- **Agente (Agent)**: Un agente es el actor fundamental en una plataforma de agentes y ofrece uno o varios servicios computacionales que se pueden publicar como una descripción del servicio. Un agente debe tener al menos un dueño y un Identificador de Agente (AID) de modo que se le pueda distinguir inequívocamente dentro del universo de agentes. Un agente se puede registrar en varias direcciones de transporte en las cuales puede ser contactado.

- **Facilitador de Directorio (Directory Facilitator, DF):** El DF ofrece servicios de Páginas Amarillas a otros agentes. Los agentes pueden registrar sus servicios en el DF o consultar el DF para averiguar qué servicios son ofrecidos por otros agentes. Pueden existir múltiples DFs dentro de un punto de acceso y puede estar federado. El DF es una reificación del Directorio de Servicio del Agente.

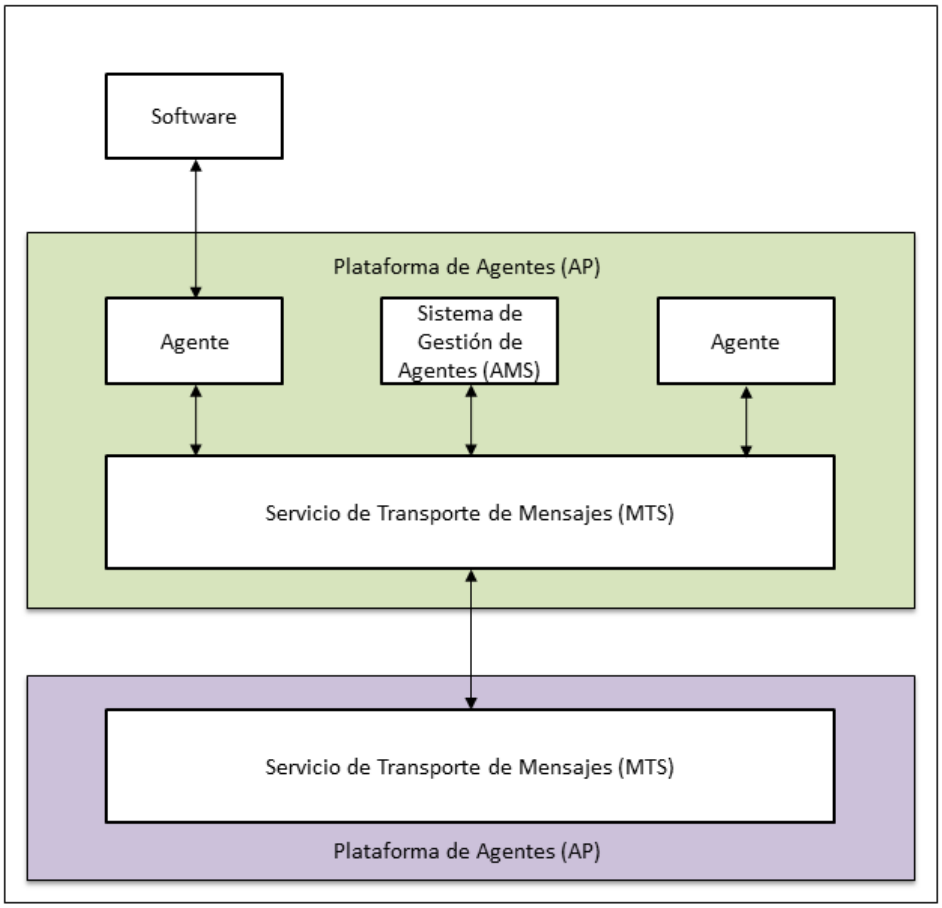


Figura 2.1. Modelo de referencia de Gestión de Agentes de FIPA

- **Sistema de Gestión de Agentes (Agent Management System, AMS):** Es un componente obligatorio de la AP. El AMS controla el acceso y uso de la AP. En una AP solo puede existir un AMS. El AMS permite controlar el ciclo de vida del agente, ofreciendo servicios para la creación, destrucción y control del cambio de estado de los agentes. Además, el AMS proporciona un servicio de nombres, también conocido como Páginas Blancas, que asocia cada AID con la dirección de transporte real en la que se encuentra, proporcionando de este modo un método básico para buscar agentes dentro de la AP.

- **Servicio de Transporte de Mensajes (Message Transport Service, MTS):** El MTS proporciona un mecanismo para el envío de mensajes ACL entre agentes, ya sea dentro de una AP o entre diferentes AP. El proceso consiste en enrutar los mensajes ACL desde el agente origen al agente destino. El MTS está provisto de un ACC (Agent Communication Channel). En el caso en el que el agente destino se encuentre en un AP distinto, el ACC local envía el mensaje al ACC remoto utilizando el conveniente MTP (Message Transport Protocol). El ACC remoto finalmente entregará el mensaje. El modelo de comunicación entre agentes es asíncrono, lo que implica que el MTS no se queda bloqueado ante el envío o recepción de mensajes, ya que existen colas de envío y recepción, para las que deben definirse políticas de gestión de colas.
- **Plataforma de Agentes (Agent Platform, AP):** Proporciona la infraestructura física en la que los agentes pueden implementarse. La AP se compone de la/s máquina/s, sistema operativo, software de soporte de agentes, componentes de administración de agentes FIPA (DF, AMS y MTS) y agentes. Cabe señalar que el concepto de plataforma no implica que todos los agentes residentes deban estar ubicados en la misma máquina. Con respecto a la comunicación entre los agentes, FIPA está implicada únicamente en cómo se lleva a cabo la comunicación entre los que son nativos de la AP y externos a la AP. Los agentes son libres de intercambiar mensajes directamente por cualquier medio que puedan soportar.
- **Software:** Describe colecciones de instrucciones ejecutables accesibles a través de un agente. Los agentes pueden acceder al software, por ejemplo, para agregar nuevos servicios, adquirir nuevos protocolos de comunicación, nuevos protocolos de seguridad / algoritmos, nuevos protocolos de negociación, etc.

Capítulo 3. JADE

Para el desarrollo de este proyecto se ha utilizado JADE. Este capítulo explica qué es JADE y lo qué ofrece para el desarrollo de SMA. Para la elaboración de este capítulo se han consultado principalmente las guías proporcionadas por la propia distribución de JADE: JADE Administrator's GUIDE [Bellifemine, 2010], JADE Programmer's GUIDE [Caire, 2010] y JADE TUTORIAL JADE PROGRAMMING FOR BEGINNERS [Caire, 2009].

3.1. ¿Qué es JADE?

JADE (Java Agent Development Environment) es un software que ofrece un marco de trabajo para facilitar el desarrollo de aplicaciones de agentes cumpliendo con las especificaciones FIPA para la interoperabilidad de Sistemas Multiagentes Inteligentes. El objetivo de JADE es facilitar el desarrollo al tiempo que garantiza el cumplimiento estándar a través de un conjunto integral de servicios de sistemas y agentes. Para lograr tal objetivo, alguna de las características que ofrece JADE son:

- AP compatible con FIPA que incluye el AMS, el DF y el ACC. Todos estos tres agentes se activan automáticamente en el arranque de la plataforma.
- Plataforma de Agentes distribuida. La AP se puede dividir en varios hosts pero solo se ejecuta una máquina virtual de Java en cada nodo. Los Agentes se implementan como hilos de Java y se elige un transporte adecuado para la entrega de mensajes, dependiendo de la ubicación relativa de los agentes emisor y receptor.
- Entorno de ejecución multiproceso con la programación de dos niveles. Cada agente JADE se ejecuta dentro de su propio hilo de control, pero también es capaz de ejecutar varias conductas simultáneamente. Una programación preventiva se realiza entre todos los agentes dentro de

una única máquina virtual de Java, mientras que la programación cooperativa se utiliza para las diversas tareas de un solo agente.

- Programación Orientada a Objetos. La mayoría de los conceptos presentes en las especificaciones FIPA se representan con clases de Java, de modo que se presenta una interfaz de programación uniforme a los usuarios.
- Biblioteca de protocolos de interacción.
- GUI (Interfaz Gráfica de Usuario) de Administración. Proporciona una herramienta gráfica que permite gestionar tanto local como remotamente el ciclo de vida del agente, es decir, crear, suspender, reanudar, migrar, clonar y matar. También proporciona herramientas que ayudan al programador a depurar y monitorizar, cómo el *Sniffer Agent* y el *Introspector Agent*.

3.2. Arquitectura JADE

Una plataforma JADE se compone [Bellifemine, 2007] de contenedores (Container) de agentes que pueden estar distribuidos por la red. Los agentes viven en contenedores que son el proceso Java que proporciona el entorno de ejecución de JADE (JADE run-time) y todos los servicios necesarios para alojar y ejecutar agentes. En cada plataforma debe existir un contenedor principal (Main-Container) que será el primero en ejecutarse, y en el que se registrarán el resto de contenedores al comenzar su ejecución. En este contenedor se alojan dos agentes especiales según especifica FIPA, el AMS y el DF. Estos dos agentes se inician automáticamente al arrancar la plataforma proporcionando la gestión del agente y el servicio de Páginas Blancas, y el servicio de Páginas Amarillas de la plataforma, respectivamente. En el host donde reside el contenedor principal, también se lanza el registro de RMI (Remote Method Invocation), que es usado internamente por JADE.

A cada plataforma se le asigna un nombre-identificador, el cual se construye por defecto según la siguiente estructura:

<hostname>:<port>/JADE

Donde:

- *hostname*: corresponde a la dirección IP o nombre DNS del host en el cual corre la plataforma, es decir, en donde se ubica el contenedor principal.
- *puerto*: puerto TCP (Transmission Control Protocol) asociado a al RMI, por defecto 1099.

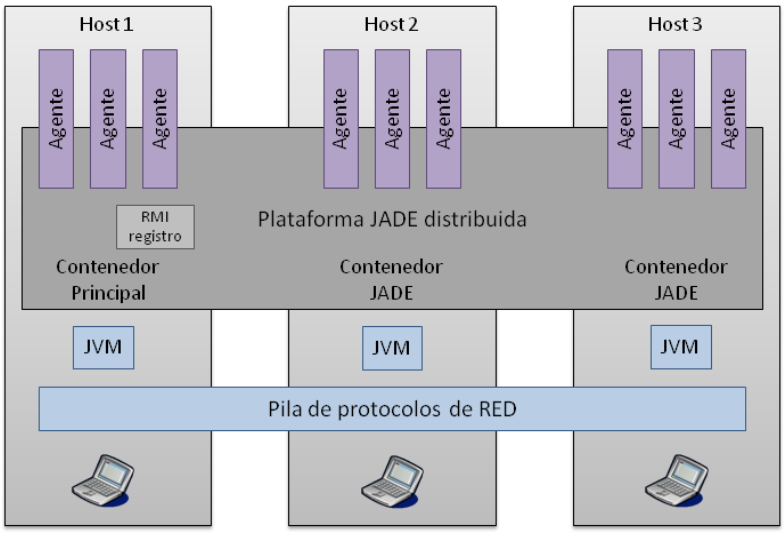


Figura 3.1 Plataforma de Agentes JADE distribuida en varios host

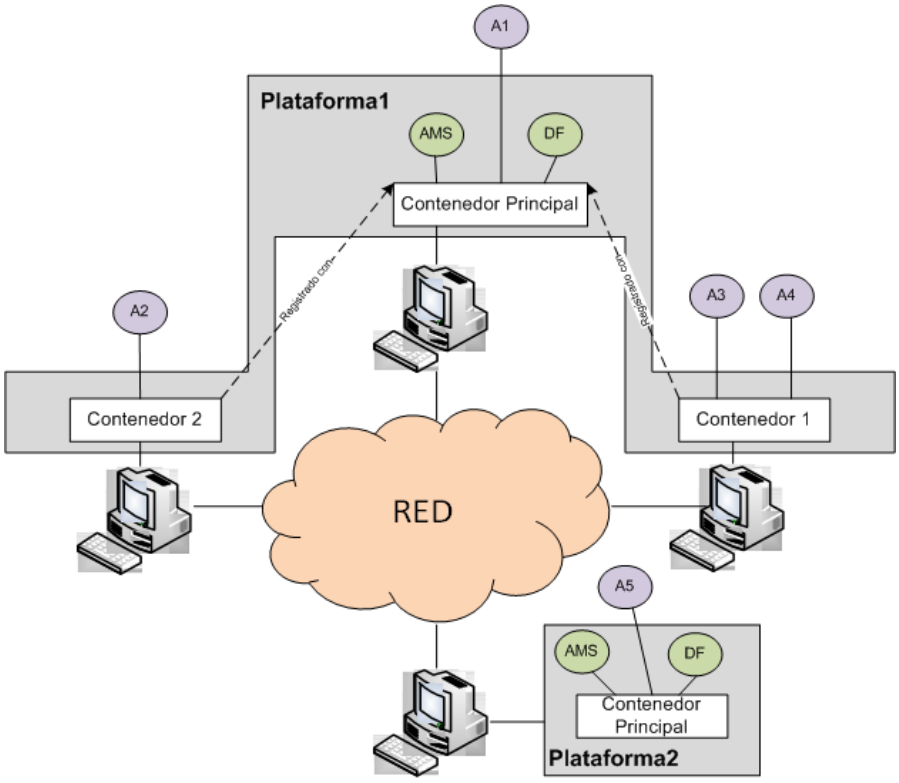


Figura 3.2 Plataforma de Agentes distribuida en una RED

La figura 3.1 muestra un ejemplo de una plataforma JADE distribuida en diferentes host.

En la figura 3.2 se representa un ejemplo de la arquitectura de la plataforma distribuida JADE. Se puede observar cómo cada plataforma aloja dentro de su contenedor principal a su AMS y DF propios.

3.3. Agente JADE

Un agente JADE es autónomo y se implementa como un hilo de ejecución independiente que decide por sí mismo cuándo y qué mensajes leer. Puede mantener varias conversaciones de forma simultánea y ejecutar varias tareas de forma concurrente.

De acuerdo con las especificaciones FIPA, cada instancia de agente se identifica por un AID. En JADE un AID se representa como una instancia de la clase `jade.core.AID`. El AID tiene los siguientes atributos:

- Identificador Global Unico (GUID): Por defecto JADE compone este nombre de la siguiente forma:

  <local-name>@<agent-platform-identifier>

  Donde:

  - *local-name*: nombre del agente, dado por el programador o desde línea de comandos al lanzar el agente en su plataforma. Dentro de una plataforma solo puede existir un agente con este nombre.
  - *agent-platform-identifier*: nombre-identificador de la plataforma origen del agente.
- Un conjunto de direcciones: Cada agente hereda las direcciones de transporte de su plataforma origen, las cuales solo son usadas cuando un agente necesita comunicarse con otro agente que vive en otra plataforma FIPA compatible.

En JADE un agente se implementa con la clase `Agent` (paquete `jade.core`), que actúa como una superclase para la creación de agentes definidos por el programador. De este modo, el agente creado, hereda todas las capacidades de la clase `Agent`, tales como mecanismos básicos de interacción con la plataforma de agentes (registro, configuración, etc). Al agente se le

pueden incorporar, a través de un conjunto básico de métodos, comportamientos (*behaviours*) específicos, en los cuales profundizaremos más adelante, para cumplir con las necesidades y objetivos del agente.

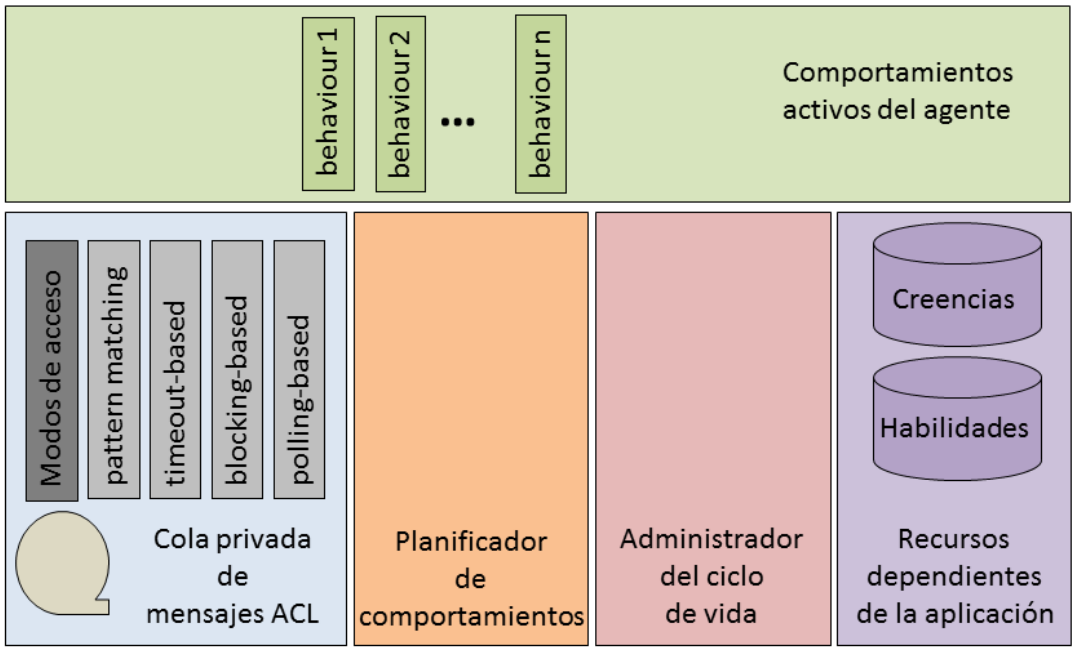


Figura 3.3 Arquitectura interna de un Agente JADE

La figura 3.3 ilustra la arquitectura interna de un agente JADE genérico. En la parte superior se encuentran los comportamientos activos del agente, que representarían sus intenciones o tareas. Cada servicio que el agente provee debe ser implementado como uno o más comportamientos, los cuales pueden ser ejecutados de forma simultánea. En la parte inferior izquierda de la figura se representa la cola privada de mensajes del agente. En el centro del dibujo se ubican: el planificador de comportamientos, el cual es responsable del orden en que se ejecutarán los mismos, y el administrador del ciclo de vida, encargado de controlar el estado actual del agente. Finalmente, en la parte derecha de la figura se encuentran los recursos del agente dependientes de la aplicación. En ese lugar serán almacenadas, por ejemplo, las creencias y habilidades que el agente vaya adquiriendo durante la ejecución de la aplicación.

3.3.1. *Ciclo de vida del Agente*

Un agente puede estar en uno de varios estados de acuerdo con el ciclo de vida especificado por el estándar FIPA. En JADE, estos estados se

representan como constantes estáticas definidas en la clase `AMSAgentDescription` (paquete `jade.domain.FIPAAgentManagement`). La clase `Agent` proporciona métodos públicos que permiten realizar las transiciones entre estados. Por ejemplo, el método `doWait()` coloca en estado *Esperando* a un agente que se encuentra en estado *Activo*, el método `doSuspend()` pasa al agente desde el estado *Activo* a estado *Suspendido*, el método `doActivate()` realiza la transición contraria al método anterior, etc. En la figura 3.4 se puede observar los estados por los que puede pasar un agente en su ciclo de vida.

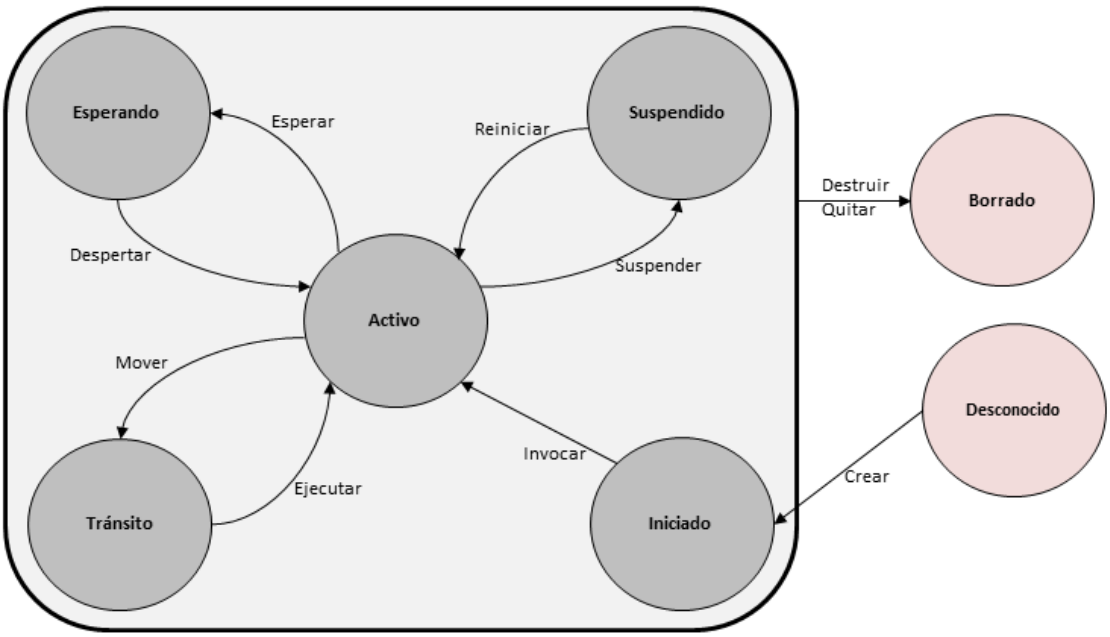


Figura 3.4 Ciclo de vida del Agente

Al crearse un agente, JADE realiza varias acciones automáticamente:

- Se llama a su constructor.
- Se crea su GUID.
- Se registra en el AMS y se le asigna el estado *Activo*.
- Se ejecuta su método `setup()`. Este método contiene las tareas de inicialización del agente, tales como registrar el agente en el DF (`DFService.register(agente,dfd)`), modificar el AMS y añadir el comportamiento del agente.

Para detener la ejecución de un agente se debe llamar al método `doDelete()` de la clase `Agent`. La invocación de este método provoca que el

agente pase a estado *Borrado*. Antes de que esto ocurra se invoca al método `takeDown()` también de la clase `Agent`, el cual puede sobrescribirse para incluir, por ejemplo, tareas de limpieza que deseen realizarse antes de que el agente sea destruido.

3.3.2. *Comunicación del Agente*

El modelo de comunicación que adopta JADE es el paso asíncrono de mensajes (figura 3.5). Cada agente tiene una especie de buzón (cola de mensajes del agente), donde JADE deposita en tiempo de ejecución los mensajes enviados por otros agentes. Siempre que un mensaje se añade en la cola de mensajes del agente receptor, este recibe una notificación. El agente decide en qué momento extrae el mensaje.

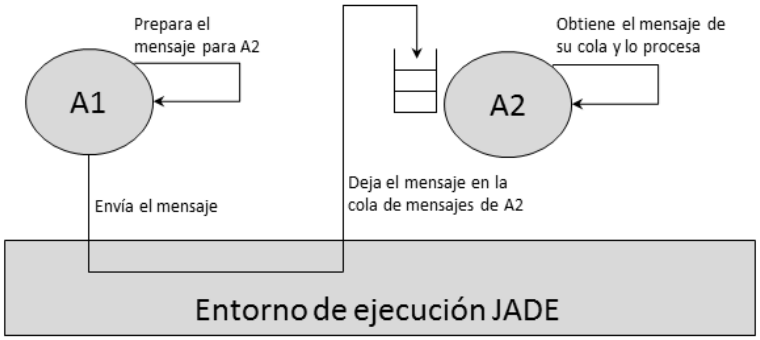


Figura 3.5 Modelo de paso asíncrono de mensaje

Para enviar un mensaje el agente debe crear una instancia de la clase `ACLMessage` (paquete `jade.lang.acl`), llenar sus atributos con los valores apropiados y, finalmente, invocar al método `send()` de la clase Agent. Para recibir un mensaje, el agente puede usar los métodos `receive()`o `blockingReceive()` también de la clase `Agent`. Con el primero, el agente accede a su cola de mensajes sin que se bloquee su ejecución; por el contrario, con el segundo método el agente se bloquea, se detiene la ejecución de todos sus comportamientos, a la espera de la llegada del mensaje requerido.

JADE utiliza el lenguaje de comunicación de agentes FIPA. Los actos comunicativos definidos por este lenguaje (ver apartado 2.4.2.2) se encuentran definidos como constantes (REQUEST, INFORM, etc.) en la clase `ACLMessage`. Cuando se crea un nuevo mensaje, una de estas constantes debe pasarse

como argumento al constructor de la clase `ACLMessage`. Esta clase contiene también el método `createReply()`, que facilita la elaboración de un mensaje en respuesta a uno recibido. Establece apropiadamente ciertos atributos del mensaje, como por ejemplo el destinatario y el identificador de la conversación, dejando que el programador especifique el acto comunicativo y el contenido del mensaje. Cuando se trata de conversaciones más complejas, JADE proporciona siguiendo el estándar FIPA un conjunto de protocolos de comunicación que detallaremos más adelante.

3.3.3. *Comportamiento del Agente*

La programación de un agente consiste en definir las tareas que necesita ejecutar para llevar a cabo su intención. Cada tarea del agente es una instancia de la clase `Behaviours` del paquete `jade.core.behaviours`.

La clase `Agent` proporciona dos métodos para añadir y eliminar comportamientos en un agente, `addBehaviour()` y `removeBehaviour()`, respectivamente. Los comportamientos pueden ser añadidos o eliminados en cualquier momento del ciclo de vida del agente, desde el método `setup()` o desde otro comportamiento, incluso también desde otros agentes.

Un agente solo puede ejecutar sus comportamientos cuando se encuentra en estado activo. Si en cualquiera de sus comportamientos se invoca al método `doWait()`, el agente completo y todas sus actividades se bloquean, no solo el comportamiento que lo invocó.

El planificador de comportamientos de un agente trabaja de modo no apropiativo, es decir, que cuando se ejecuta un comportamiento no puede ser interrumpido hasta que este finalice completamente. Una vez que un comportamiento se selecciona para su ejecución se invoca su método `action()` y se ejecuta hasta que devuelve el control, por lo que se recomienda que este método no tenga un tiempo de ejecución alto. Al finalizar el método `action()` se invoca al método `done()`. Éste retorna un valor booleano que indica que el comportamiento ha llegado a su fin.

Otros métodos proporcionado por la clase `Behaviour` son `restart()`, `onStart()`, `onEnd()`, `reset()` y `block()`. Entre ellos se puede destacar el método `block()`, el cual bloquea el comportamiento a la espera de un evento, o introduce un retardo en la ejecución si se indica una periodo de tiempo. Este

método debe ejecutarse en lugar de `sleep()`, ya que este último paralizaría al agente por completo y no solo al comportamiento deseado.

Cómo hemos comentado anteriormente, los comportamientos nos permiten definir cómo va a actuar el agente. En algunos casos esta tarea puede ser compleja y puede ser necesaria la utilización de varios comportamientos diferentes para conseguir el deseado. Para ello, JADE nos proporciona una librería de clases que nos permite implementar diferentes tipos de comportamientos (*behaviours*).

Podemos distinguir entre:

- **Comportamientos Simples**. La clase `SimpleBehaviour` modela comportamientos atómicos. Dentro de ella se encuentran las siguientes subclases:
  - Clase `OneShotBehaviour`: Solo ejecuta el método `action()` una sola vez. El método `done()` ya está implementado y siempre devuelve true.
  - Clase `TickerBehaviour`: Tiene sus métodos `action()` y `done()` ya implementados. Es un comportamiento cíclico que ejecuta las operaciones especificadas en método `onTick()` tras un periodo de tiempo indicado en el constructor.
  - Clase `WakerBehaviour`: Tiene sus métodos `action()` y `done()` ya implementados. El método donde se especificarán las acciones será `handleElpsedTimeout()`, el cual se ejecuta solo una vez después de un tiempo indicado en el constructor del comportamiento.
  - Clase `CyclicBehaviour`: Este tipo de comportamiento nunca termina mientras el agente está activo, y su método `action()` ejecuta las mismas operaciones cada vez que es llamado. El método `done()` ya está implementado y siempre devuelve false.
- **Comportamientos Compuestos.** La clase `Composite` modela comportamientos a partir de la composición de otros comportamientos (hijos). La operaciones del comportamiento no son definidas directamente en sus método `action()`, sino en los comportamientos de los hijos. En esta clase no se define la política de planificación, dejando esta tarea a las siguientes subclases:

- Clase `SequentialBehaviour`: Ejecuta sus subcomportamientos de forma secuencial y acaba una vez éstos hayan finalizado. Este comportamiento es útil cuando la tarea a realizar se puede dividir en subtareas atómicas con la recepción de un mensaje o evento entre ellas.
- Clase `ParallelBehaviour`: Ejecuta sus subcomportamientos de forma concurrente, y acaba cuando se cumple una cierta condición; que todos los subcomportamientos finalicen, que sólo sea necesario que finalice uno o bien que sea necesario que finalicen un número indicado de subcomportamientos.
- Clase `FSMBehaviour`: ejecuta los comportamientos hijo de acuerdo a una máquina de estados finitos definidos por el usuario. Cada hijo representará las tareas a realizar en un estado determinado. Cuando un hijo finaliza su ejecución, su valor de terminación, devuelto en el método `onEnd()`, será utilizado para decidir a qué nuevo estado pasar. Cabe destacar que algunos de los estados deberán ser definidos como finales, de forma que el comportamiento `FSMBehaviour` finalice al alcanzar uno de estos estados.

3.3.4. *Protocolos de Comunicación*

Para toda conversación entre agentes, JADE distingue entre el rol *Initiator*, agente que inicia la conversación, y el rol *Responder*, agente destino de la conversación. JADE proporciona comportamientos para ambos roles siguiendo los protocolos de interacción FIPA. Estas clases se encuentran en el paquete `jade.proto`:

- `AchieveREInitiator/Responder`. Proporciona una efectiva implementación de los protocolos de interacción FIPA Request, Query, Request-When, Recruiting y Brokering.
- `ContractNetInitiator/Responder`. Implementa los protocolos de interacción FIPA Contract-Net o Iterated-Contract-Net
- `SimpleAchiveREInitiator/Responder`. Es una implementación sencilla de las clase `AchieveREInitiator/Responder` para el caso particular de conversaciones 1:1.

- `SubscriptionInitiator/Responder`. Implementa el protocolo de interacción FIPA Subscribe.
- `ProposeInitiator/Responder`. Implementa el protocolo de interacción FIPA-Propose.

En el siguiente apartado se describe en detalle las clase AchieveREInitiator/Responder ya que ha sido el protocolo de comunicación utilizado en el desarrollo de este proyecto.

3.3.5. *Clases AchieveREInitiator/Responder*

Como se ha indicado anteriormente estas clases implementan los protocolos de interacción FIPA Request, Query, Request-When, Recruiting y Brokering, los cuales permiten al *Initiator* verificar si el efecto racional esperado (Rational Effect, RE) de un solo acto comunicativo se ha alcanzado.

3.3.5.1 Clase AchieveREInitiator

Esta clase implementa el rol del *Initiator*. El constructor de la clase AchieveREInitiator recibe como parámetros una referencia al agente y el mensaje usado para iniciar el protocolo. Es importante que este mensaje contenga en el campo protocol el nombre del protocolo de interacción correspondiente. Los nombres de los protocolos se encuentran disponibles en la interfaz FIPANames.InteractionProtocols del paquete `jade.domain`. Para la gestión de la conversación esta clase proporciona los siguientes métodos:

- `prepareRequest(ACLMessage)`. Este método se llama en el primer estado del protocolo. Devuelve un vector de mensajes a enviar, ya que el *Initiator* puede gestionar 1:N conversaciones.
- `handleNotUnderstood/Refuse/Agree(ACLMessage)`. Estos métodos se utilizan para tratar la primera respuesta. Se disparan cuando se recibe un mensaje con una performativa *NotUnderstood*, *Refuse* o *Agree*.
- `handleFailure/Inform()`. Se utilizan en el tratamiento de la segunda respuesta. Se disparan cuando se recibe un mensaje con una performativa *Failure* o *Inform*.

- `handleAllResponses/ResultNotificationsProtocolos(java.util.Vector)`. Se utiliza en protocolos 1:N. Se disparan cuando se han recibido todas las respuestas o los resultados de notificaciones.
- `handleOutOfSequence()`. Trata mensajes fuera de secuencia, pero cuyo valor en el campo `conversation-id` o `in-replay-to` es correcto.
- `registerHandle...(Behaviour)`. Permite registrar nuevos comportamientos para el tratamiento de mensajes con el fin de sustituir los predefinidos.

Para obtener información del *DataStore,* objeto que guarda información de todos los mensajes y que puede ser utilizado como repositorio para intercambiar datos entre tareas, existe un conjunto de variables para utilizar con el método `getDataStore().get(…_KEY)`:

- `REQUEST_KEY`. Devuelve el mensaje pasado en el constructor.
- `ALL_REQUESTS_KEY`. Devuelve el vector de mensajes devuelto por el método `handleRequest`.
- `ALL_RESPONSES_KEY`. Devuelve un vector de mensajes con todas las primeras respuestas (*NotUnderstood*, *Refuse* o *Agree*).
- `ALL_RESULT_NOTIFICATIONS_KEY`. Devuelve un vector de mensajes con todas las segundas respuestas (*Failure* o *Inform*).

3.3.5.2 Clase AchieveREResponder

Esta clase implementa el rol del Responder. El constructor de la clase recibe como parámetros una referencia del agente y una plantilla de mensaje que se usa para seleccionar los mensajes a responder.

Los métodos proporcionados por esta clase para la gestión de la conversación son los siguientes:

- `createMessageTemplate(String)`. Se usa para crear la plantilla de mensaje `(MessageTemplate)`. En este caso se le indica como parámetro el nombre del protocolo a seguir en la comunicación.
- `handleRequest(ACLMessage)` que sustituye al obsoleto `prepareResponse(ACLMessage)`. Este método se ejecuta cuando llega un mensaje que coincide con la plantilla definida al crear el

comportamiento. Proporciona la primera respuesta al protocolo, la cual puede ser *NotUnderstood*, *Refuse* o *Agree*.

- `prepareResultNotification(ACLMessage)`. Cuando la primera respuesta ha sido *Agree*, se ejecuta este método. Proporciona la segunda respuesta al protocolo.
- `registerPrepare…(Behaviour)`. Permite sustituir los comportamientos que tratan los mensajes. Cuando se utiliza este método, es responsabilidad del nuevo comportamiento, insertar en el *DataStore* (`getDataStore().put(RESULT_NOTIFICATION_KEY, reply)`) el mensaje que debe ser enviado por el *Responder*.

Para acceder al contenido del *DataStore* de este comportamiento existe el siguiente conjunto de variables para utilizar con el método `getDataStore().get(…_KEY)`:

- `REQUEST_KEY`. Devuelve el mensaje recibido del *Initiator*.
- `RESPONSE_KEY`. Devuelve el primer mensaje enviado al *Initiator*.
- `RESULT_NOTIFICATION_KEY`. Devuelve el segundo mensaje enviado al *Initiator*.

3.4. Instalación y ejecución de JADE

El software y la documentación necesaria para trabajar con JADE se pueden descargar del sitio web de JADE (http://jade.tilab.com/). La versión de JADE utilizada en el desarrollo de este proyecto ha sido JADE v4.3.0, aunque actualmente existe una versión más actualizada JADE v4.3.3.

El archivo comprimido JADE-all proporciona una distribución completa que incluye:

1. JADE-src: El código fuente de JADE.
2. JADE-examples: El código fuente de los ejemplos.
3. JADE-doc: Documentación que incluye también el javadoc del API de JADE.
4. JADE-bin: El binario de JADE, que incluye por ejemplo los ficheros jar con todas las clases Java.

Después de descomprimir los archivos, es conveniente añadir los archivos jar que se encuentra en jade\lib en el CLASSPATH.

El único software requerido para ejecutar JADE es la máquina virtual de Java versión 5 o superior.

La plataforma JADE se lanza desde la línea de comando utilizando la siguiente instrucción:

```
java [-cp lib\jade.jar ]jade.Boot [options] [AgentSpecifier list]
```

Donde:

- `-cp lib\jade.jar`: Si no está definido previamente el CLASSPATH.
- `options`: De entre todas las opciones posibles destacamos las siguientes:
  - `-container`: especifica el contenedor si es distinto del principal.
  - `-host`: nombre del host.
  - `-port`: puerto, por defecto es el 1099.
  - `-gui`: para lanzar la herramienta gráfica RMA (Remote Monitoring Agent)
  - `-platform-id o -name`: para asignar un nombre concreto a una plataforma.
- `AgentSpecifier list`: es una lista de "Agent Specifier", agente que se desea ejecutar, separada por puntos y comas (;) con el siguiente formato:

AgentName:ClassName(ArgumentList)

Donde:

- *AgentName*: nombre del agente en la plataforma.
- *ClassName*: nombre cualificado de la clase que implementa el agente.
- *ArgumentList*: lista de argumentos separados por comas (,) que se pasan al agente en su construcción. Si algún argumento contiene espacios en blanco, deberá ir entre comillas dobles.

3.5. Herramientas Gráficas

Para llevar a cabo la difícil tarea de depurar aplicaciones Multiagentes, JADE proporciona un conjunto de herramientas para la administración y monitorización de las AP. Cada herramienta está implementada como un agente, por lo que está sujeto a las mismas reglas, es decir, tiene la misma capacidad de comunicación y el mismo ciclo de vida que cualquier otro agente. Las clases que proporcionan estas herramientas se encuentran en el paquete `jade.tools`.

3.5.1. *RMA*

El RMA permite controlar mediante su interfaz gráfica el ciclo de vida de la AP y de todos los agentes registrados en ella.

Para ejecutar esta herramienta, se puede lanzar desde la línea de comandos como cualquier otro agente (`java jade.Boot myrma:jade.tools.rma.rma`), o bien utilizando la opción `-gui` como parámetro de la línea de comandos (`java jade.Boot -gui`).

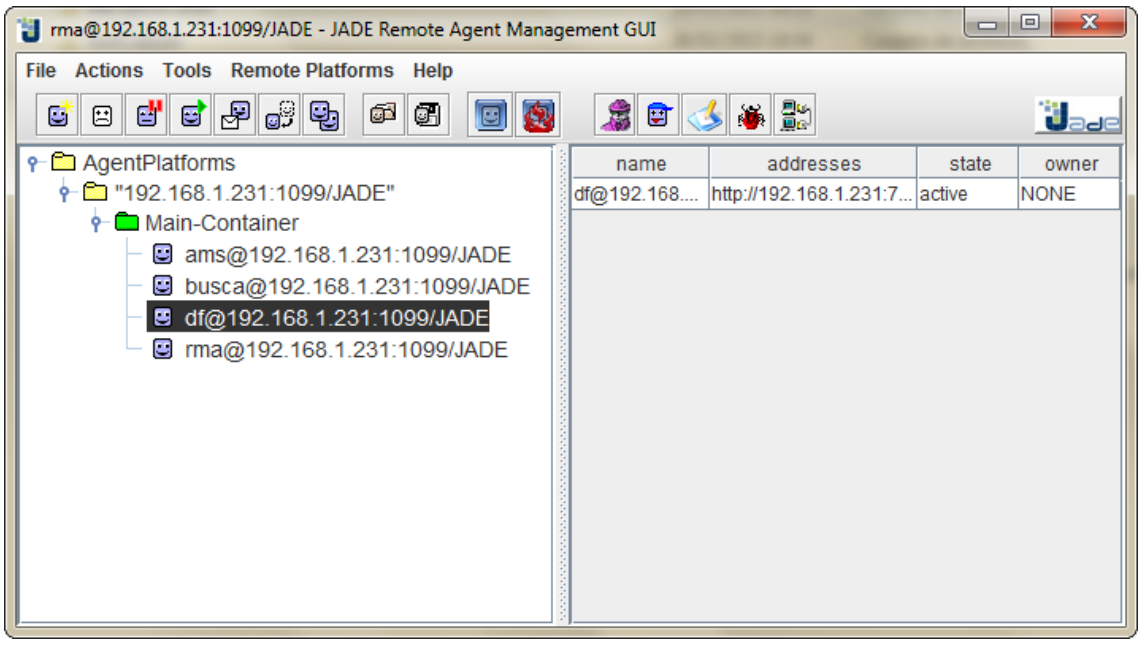


Figura 3.6 RMA GUI

Como se observa en la figura 3.6, desde esta herramienta podemos lanzar el resto de utilidades gráficas. Otra funcionalidad que nos ofrece la

RMA es la supervisión de plataformas remotas que cumplen con el estándar FIPA.

Seleccionando el primer icono de la izquierda (Start New Agent), se abre una ventana (figura 3.7) en la que se puede introducir todos los valores necesarios para iniciar un nuevo agente.

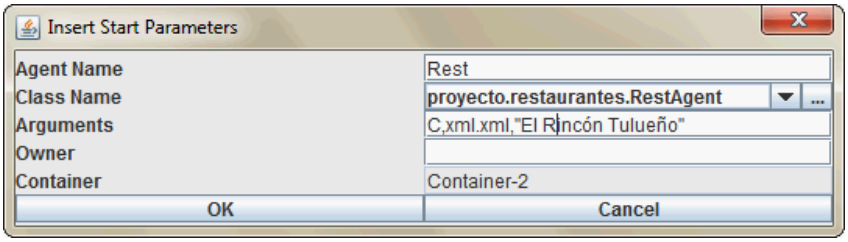


Figura 3.7 Inicio nuevo Agente

3.5.2. *DF GUI*

Esta herramienta (figura 3.8) facilita al usuario la interacción con el DF. Permite de una forma sencilla registrar y deregistrar agentes, consultar la descripción de los agentes registrados y modificar dicha descripción y buscar agentes que coincidan con una descripción dada.

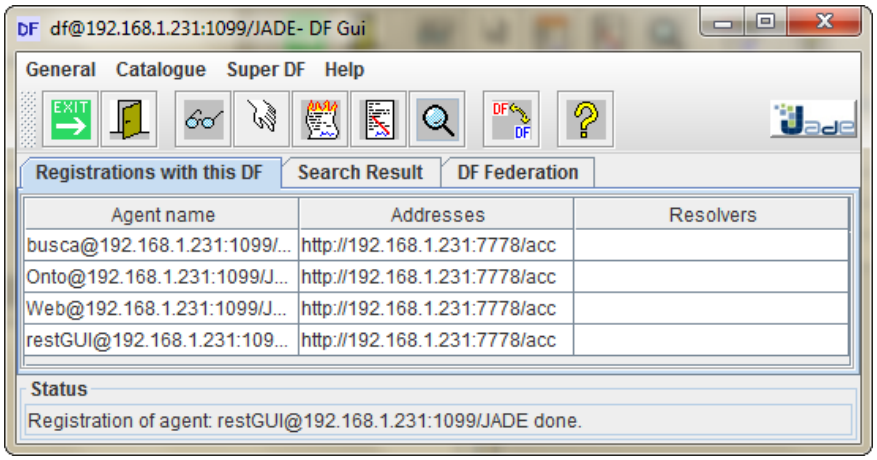


Figura 3.8 DF GUI

3.5.3. *DummyAgent*

Esta herramienta permite a los usuarios interactuar con agentes JADE de una forma personalizada. Permite redactar y enviar mensajes ACL y mantiene una lista de todos los mensajes enviados y recibidos. Cada mensaje

de la lista se puede ver de forma detallada. La lista de mensajes puede ser guardada en disco para ser recuperada posteriormente.

El DummyAgent se puede lanzar desde el RMA o bien por línea de comando con:

```
java jade.Boot theDummy:jade.tools.DummyAgent.DummyAgent.
```

En la figura 3.9 se puede ver un ejemplo de funcionamiento de esta herramienta.

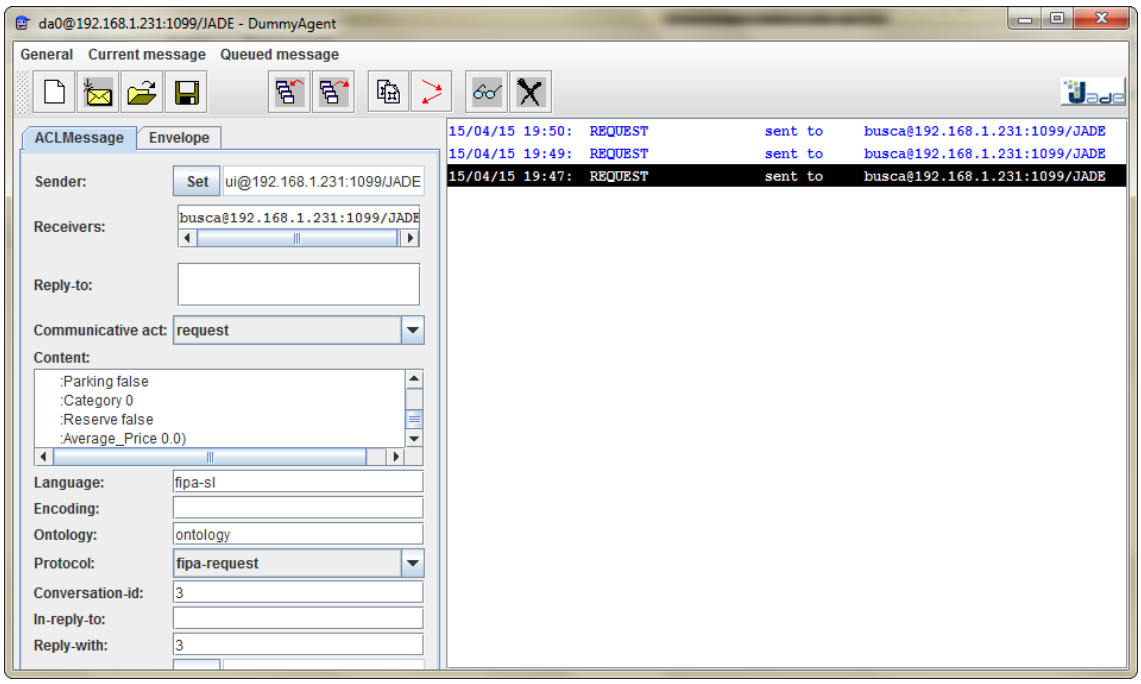


Figura 3.9 DummyAgent GUI

3.5.4. *Sniffer Agent*

El agente Sniffer permite interceptar los mensajes de un agente o grupo de agentes seleccionados. Este agente sigue la pista de cada mensaje dirigido hacia/desde cualquier agente y muestra en su interfaz todo el intercambio realizado. Seleccionando un mensaje se puede ver el contenido del mismo. La traza puede ser grabada para su posterior análisis. Esta herramienta proporciona una gran ayuda para la depuración de la interacción entre los agentes.

Esta herramienta se puede lanzar desde el RMA, o bien por línea de comando con:

```
java jade.Boot sniffer:jade.tools.sniffer.Sniffer.
```

La figura 3.10 muestra una traza capturada por el agente Sniffer.

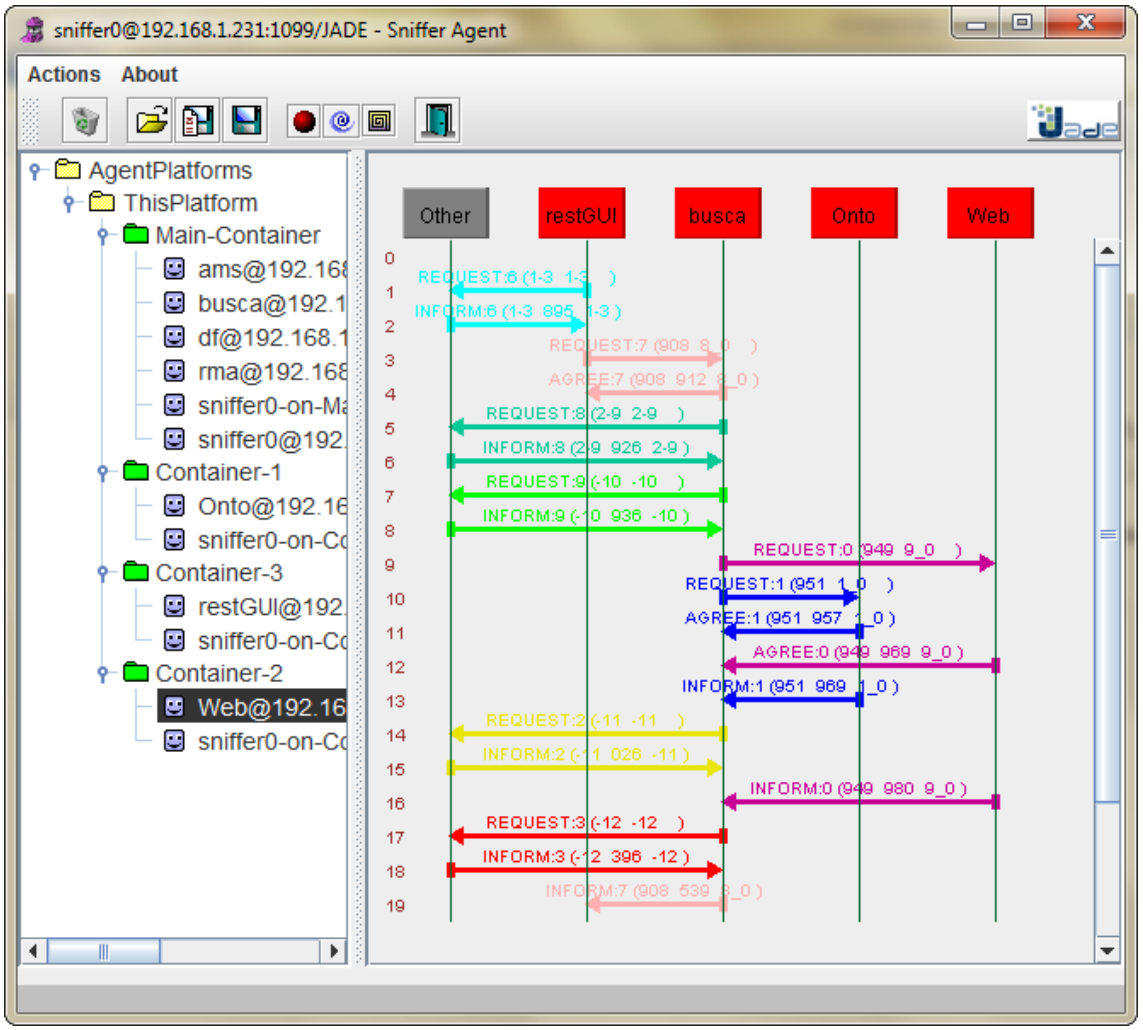


Figura 3.10 Sniffer Agent GUI

3.5.5. *Introspector Agent*

Esta herramienta permite monitorizar y controlar el ciclo de vida de un agente en ejecución y el intercambio de mensajes, tanto la cola de mensajes enviados como la de recibidos. También permite supervisar la cola de comportamientos, permitiendo la ejecución de los mismos paso a paso.

Esta herramienta se puede lanzar desde el RMA, o bien por línea de comando con:

```
jade.Boot introspector:jade.tools.introspector.Introspector.
```

En la figura 3.11 se puede observar toda la funcionalidad que ofrece esta herramienta, en la parte superior se visualizar las colas de mensajes y en la inferior la de comportamientos.

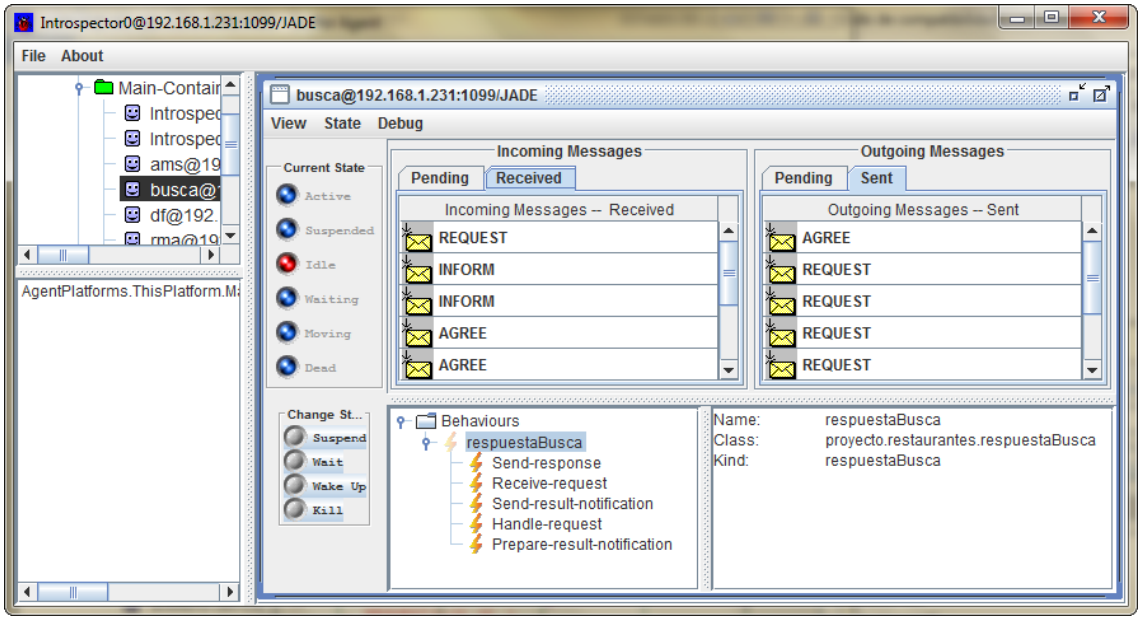


Figura 3.11 Introspector Agent GUI

Capítulo 4. Objetivos

El objetivo final de este proyecto fin de carrera ha sido la construcción de una herramienta capaz de proporcionar al usuario información de restaurantes. Para ello, se ha implementado un SMA utilizando la herramienta de desarrollo de agentes JADE.

Para alcanzar el objetivo final ha sido necesario cumplir primero con los siguientes subojetivos:

1. Familiarización con el lenguaje JAVA.
2. Comprender el concepto de Agente y de SMA.
3. Conocer las funcionalidades proporcionadas por las librerías de JADE para implementar Agentes, especialmente para gestionar los comportamientos e implementar los protocolos de comunicación.
4. Aprender a utilizar las herramientas gráficas proporcionadas por JADE, para utilizarlas de ayuda a la hora de depurar la aplicación.
5. Implementación en JAVA del SMA.
6. Realización de pruebas para verificar el correcto funcionamiento de la herramienta construida.

Capítulo 5. Trabajo Realizado

En este capítulo se describe el trabajo realizado, consistente en el desarrollo de un SMA para implementar un buscador de información de restaurantes.

5.1. Introducción

El proyecto desarrollado, Buscador de Información de Restaurantes, es un ejemplo de funcionamiento de un SMA. Para implementar dicho SMA se ha utilizado la herramienta JADE.

El objetivo de este SMA es resolver las peticiones de los usuarios proporcionando información específica sobre los restaurantes demandados. Para conseguir este objetivo se han implementado diferentes agentes, cada uno de ellos con un rol específico:

- **BuscaAgent**: Este agente atiende las solicitudes de RestAgent. BuscaAgent conversará con WebAgent y OntoAgent para obtener la información solicitada por RestAgent.
- **RestAgent**: Agente que atiende las peticiones de los usuarios.
- **WebAgent**: Agente especializado en realizar búsquedas en la web.
- **OntoAgent**: Agente especializado en realizar búsquedas en un fichero que contiene información de restaurantes almacenados con el formato de la ontología.
- **ActualizarOntoAgent**: Cuando el agente BuscaAgent ha obtenido alguna información útil del agente WebAgent, reenvía dicha información a este agente, para que se encargue de actualizarlo en la ontología.

En la figura 5.1 se puede observar la arquitectura del SMA para tener una visión clara de las interacciones entre los agentes, y de los agentes con los elementos externos. Por otra parte, permite ver de forma sencilla, y a grandes rasgos, el flujo que sigue la información, desde que el usuario lanza una

petición hasta que recibe una respuesta. Cuando un usuario lanza una consulta a RestAgent, este realiza una petición de consulta a BuscaAgent, el cual traslada dicha consulta a los agentes de tipo WebAgent y OntoAgent que se encuentren activos en el SMA. BuscaAgent analiza las respuestas a su consulta y decide, según la estrategia configurada, que respuesta proporciona a RestAgent.

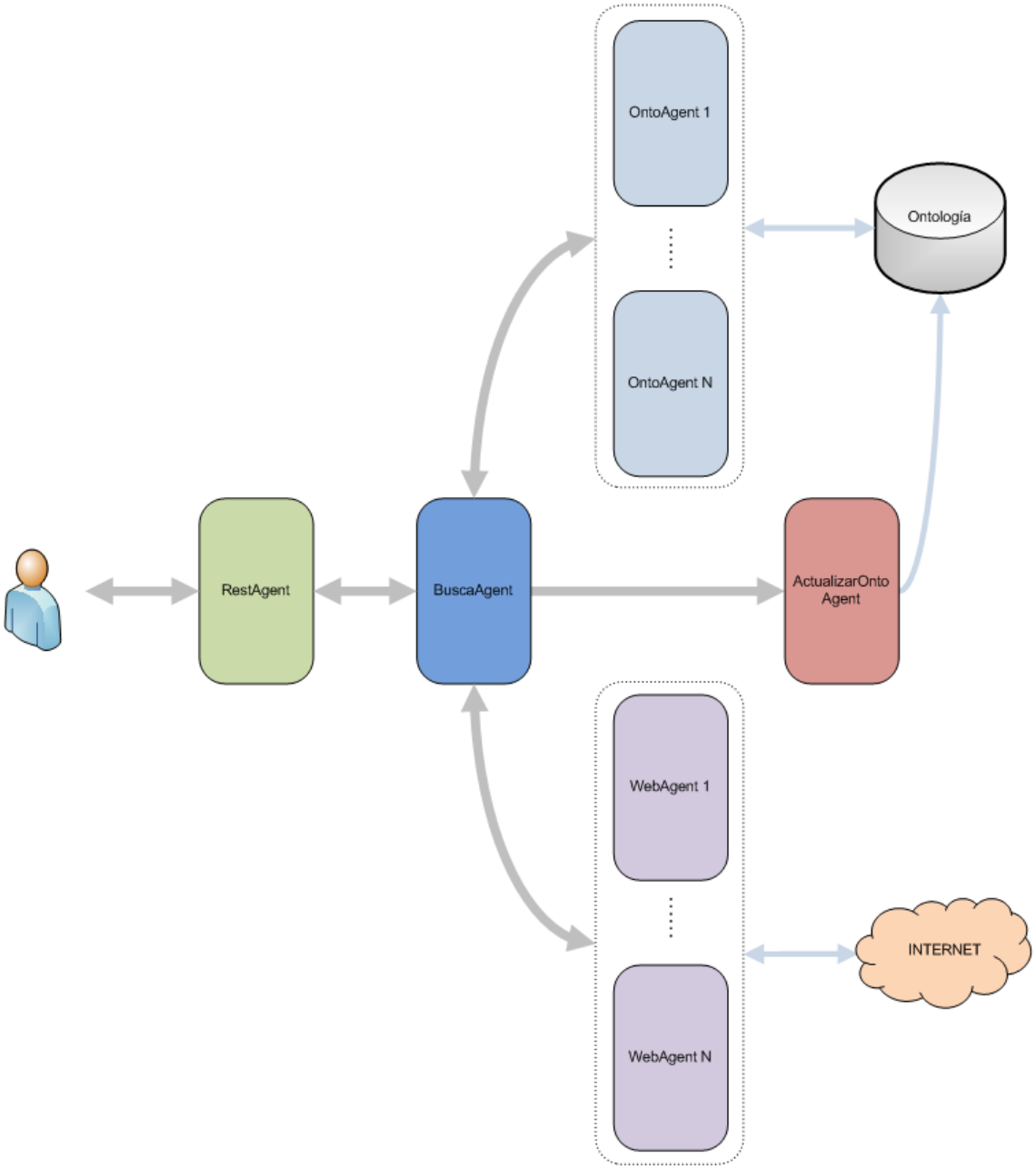


Figura 5.1 Arquitectura SMA: Buscador de Información de Restaurantes

Las conversaciones mantenidas entre los diferentes agentes se han realizado de acuerdo a las normas que dicta el protocolo de comunicación FIPA Request. Además, para que todos los agentes puedan interpretar de forma correcta la semántica del contenido de los mensajes, deben utilizar un vocabulario común; es decir, deben de utilizar la misma Ontología. En el Anexo 1 se incluye un esquema con la definición de la ontología utilizada en el desarrollo de este proyecto.

En los siguientes apartados profundizaremos más en detalle en la implementación y en el comportamiento de cada agente, haciendo especial hincapié en el agente BuscaAgent, por ser este agente el núcleo de este sistema.

5.2. BuscaAgent

BuscaAgent es el núcleo del SMA. Para obtener la información de los restaurantes que RestAgent le solicita, conversa con los agentes WebAgent y OntoAgent. La clase `BuscaAgent` hereda de la clase `jade.core.Agent`.

Cuando el agente BuscaAgent es creado, recibe cuatro argumentos que van a configurar su comportamiento:

1. TIPO_BUSQUEDA: Este argumento configura la forma de ejecución de los comportamientos ejecutados para mantener conversaciones con WebAgent y OntoAgent, de forma sencuencial (S) o en paralelo (P)
2. ORDEN: Indica a quién se va a enviar en primera opción la consulta, a WebAgent(WEB) o a OntoAgent(ONTO)
3. ESTRATEGIA: Para decidir qué respuesta proporciona a RestAgent, se han definido tres estrategias diferentes, por tanto, los valores admitidos son 1, 2 y 3.
4. TODOS: El valor 1 indica que buscará en el DF a todos los agentes de un determinado tipo, si el valor es 0 solo buscará a uno.

Además de estos cuatro argumentos, opcionalmente puede recibir un quinto. Este último argumento es MEDICIÓN, y como su propio nombre indica se utiliza cuándo queremos tomar mediciones. El único valor que admite es M.

En el método `Setup`, como ya mencionamos en el capítulo anterior, se registra el agente en el DF. Este método es el primero en ejecutarse cuando el agente es creado. En él se analizan los argumentos recibidos y se añade el

primer comportamiento que va a tener este agente y que le otorga el rol de *Responder,* la clase `respuestaBusca` que hereda de la clase `AchieveREResponder` (figura 5.2). Una peculiaridad de este agente es que por una parte va a desempeñar este rol de *Responder* respecto a la comunicación con RestAgent, y por otra parte el rol de *Initiator,* clase `iniciadorBusca` que hereda de la clase `AchieveREInitiator`, en su comunicación con WebAgent y OntoAgent.

```
MessageTemplate plantilla =
       AchieveREResponder.createMessageTemplate(FIPAProtocolNames.FIPA_REQUEST);
respBusca = new respuestaBusca(this,plantilla);
addBehaviour(respBusca);
```

Figura 5.2 Fragmento de código

5.2.1. *Clase respuestaBusca*

Como hemos comentado, cuando BuscaAgent recibe una petición de consulta de RestAgent, la traslada a los agentes WebAgent y OntoAgent. Esto implica que no va a poder responder de forma inmediata a RestAgent, y que va a tener que esperar a recibir las respuestas de WebAgent y OntoAgent para analizarlas y decidir qué respuesta proporciona a RestAgent. Para que BuscaAgent se comporte de este modo, ha sido necesario redefinir el método `prepareResultNotification` y sustituirlo por un comportamiento adaptado a las necesidades de BuscaAgent.

```
registerPrepareResultNotification(new comprobarRespuesta(m_agente, 30 , peticion));
```

Figura 5.3 Fragmento de código

La figura 5.3 muestra cómo se sustituye el método `prepareResultNotification` por el método `comprobarRespuesta`. Además, el hecho de realizar esta sustitución, obliga a introducir manualmente en el *DataStore* los mensajes a enviar.

En resumen, la clase `respuetaBusca` ejecuta dos métodos, el método `handleRequest` y el método `comprobarRespuesta`.

El método `handleRequest` recibe las peticiones de consulta de RestAgent, es decir, mensajes con la performativa *Request*. Si no entiende el mensaje, le devuelve la performativa *NotUnderstood*. En caso contrario, le envía la performativa *Agree* y comienza a trabajar en la búsqueda de la

respuesta. Primero añade el identificador de la conversación en una lista de conversaciones activas. El uso de esta lista se explicará más adelante. Posteriormente se lanzarán una serie de comportamientos destinados a localizar agentes WebAgent y OntoAgent e iniciar conversaciones con ellos para obtener la información del restaurante solicitado. En función del valor de los argumentos TIPO_BUSQUEDA y ORDEN se realizará la planificación de los comportamientos. Sin lugar a dudas, una vez superados los problemas iniciales, la parte más compleja fue la de conseguir que la planificación de los comportamientos se ejecutara en el orden y la forma deseada. En el caso de la comunicación agente-agente resulta relativamente sencillo, pero cuando la comunicación se realiza agente-agentes adquiere una mayor complejidad. Esta dificultad se salvó añadiendo algunos de los comportamientos de forma dinámica, de tal forma, que por cada agente encontrado del tipo OntoAgent o WebAgent se agrega un nuevo comportamiento para iniciar la conversación con él; es decir, se agregará el comportamiento que le otorga el rol de *Initiator,* la clase `iniciadorBusca`.

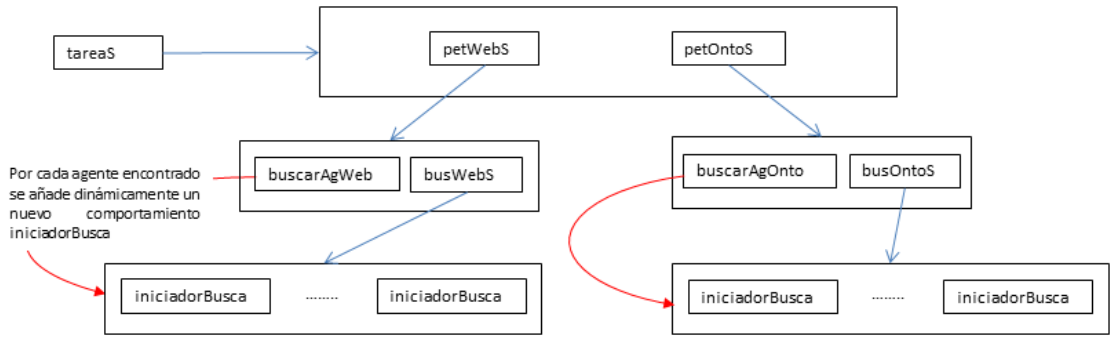


Figura 5.4 Planificación de Comportamientos: TIPO_BUSQUEDA = S, ORDEN = WEB

La figura 5.4 muestra de forma gráfica la planificación de los comportamientos cuando la búsqueda se realiza de forma secuencial y se lanzan las peticiones de consulta primeramente a los agentes WebAgent. En la figura 5.5 se puede observar como varía la planificación en el caso de que la búsqueda se realice en paralelo. En este caso, el orden no es un parámetro a tener en cuenta.

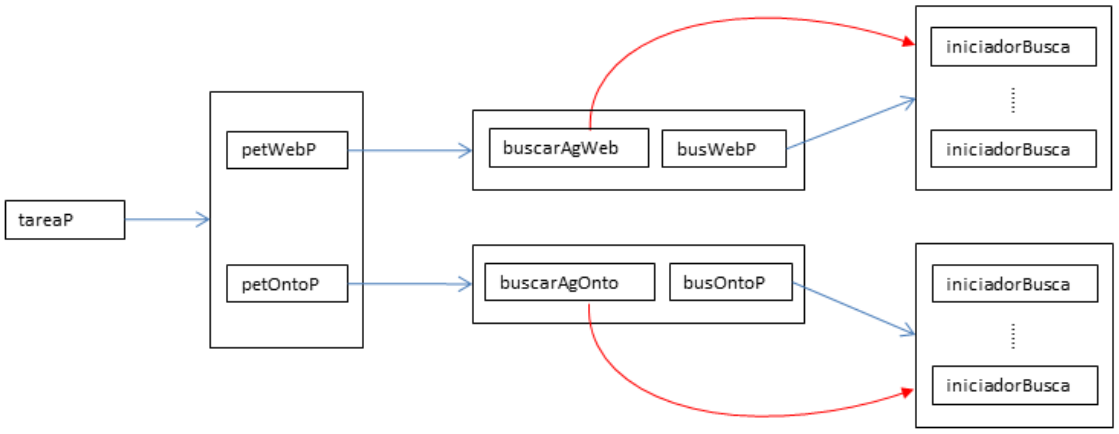


Figura 5.5 Planificación de Comportamientos: TIPO_BUSQUEDA = P

Por todo lo visto anteriormente, está claro que BuscaAgent va a recibir diferentes respuestas por cada petición de información de un restaurante. Para analizar estas respuestas y decidir cuál se envía a RestAgent se ha creado un comportamiento implementado en la clase `comprobarRespuesta`. Esta clase se ayuda de dos herramientas para que BuscaAgent conozca en todo momento las conversaciones activas con RestAgent pendientes de dar respuesta y qué mensajes recibidos de WebAgent y OntoAgent le sirven para dar respuesta a una determinada conversación con RestAgent. Estas dos herramientas son la lista de conversaciones activas, ya mencionada anteriormente, y una lista de mensajes pendientes en la cual se almacenan todas las respuestas recibidas de WebAgent y OntoAgent junto con alguna información más necesaria para su correcto procesamiento. En el siguiente apartado veremos en detalle la estructura de esta lista. Una vez comentadas las herramientas que nos van a ayudar, vamos a explicar qué hace el comportamiento `comprobarRespuesta`.

El comportamiento `comprobarRespuesta` hereda de la clase `TickerBehaviour`, ya que nos interesa que se ejecute cíclicamente cada cierto tiempo. Por otra parte, no deseamos que esté de forma indefinida. Por ello, pasado 30 ciclos, si no hemos obtenido respuesta de WebAgent y OntoAgent se optará por enviar un mensaje a RestAgent informándole de que el tiempo de búsqueda ha expirado y se dará por finalizada la tarea. En cambio, si antes de finalizar el periodo establecido, se encuentran mensajes en la lista de mensajes pendientes hay que proceder a procesarlos para decidir qué respuesta final se envía a RestAgent. Aquí entra en juego el argumento

ESTRATEGIA. En función del valor de este argumento BuscaAgent decidirá la respuesta final a enviar a RestAgent. Hay 3 estrategias definidas:

- Estrategia 1: Se busca en la lista de mensajes pendientes el primer registro que exista para el identificador de la conversación a tratar. Si este mensaje ya proporciona información sobre el restaurante solicitado, se selecciona como respuesta definitiva. En caso contrario, se procede a buscar un segundo mensaje en la lista de mensajes pendientes, teniendo en cuenta que además este segundo mensaje debe de proceder de un originador diferente al primer mensaje; es decir, si el primer mensaje provenía de un agente WebAgent, el segundo mensaje deberá provenir de un agente OntoAgent y viceversa. En este caso, el segundo mensaje será el seleccionado, contenga o no la información solicitada. Si no se ha localizado un segundo mensaje antes de que se cumplan los 6 ciclos, el primer mensaje será el seleccionado. Con esta estrategia se pretende que aunque el tiempo de respuesta pueda ser mayor, aumente la probabilidad de que la respuesta sea positiva; es decir, que contenga la información solicitada.
- Estrategia 2: Se selecciona de la lista de mensajes pendientes el primer registro que exista para el identificador de la conversación a tratar. Este mensaje será el seleccionado independientemente de contenga una respuesta positiva o no.
- Estrategia 3: Se selecciona de la lista de mensajes pendientes, siempre que sea posible, un mensaje proveniente de WebAgent y otro de OntoAgent. Si ambos contienen información, se realiza una suma con la información de ambos, que será la que se envíe a RestAgent. Si por el contrario, sólo uno de los mensajes contiene una respuesta positiva, será el que se seleccione como definitivo. Por supuesto, si cumplido el periodo establecido únicamente se ha localizado un mensaje, este será el seleccionado.

Una vez decidida la información que se va a trasladar a RestAgent, se inserta el mensaje con la misma en el *DataStore* (figura 5.6).

```
getDataStore().put(((AchieveREResponder) parent).RESULT_NOTIFICATION_KEY, reply);
```

Figura 5.6 Fragmento de código

Dado que esta petición se puede dar por finalizada, se procede a borrar de la lista de mensajes pendientes, si los hubiera, el resto de los mensajes asociados a esta petición y de la lista de conversaciones activas la conversación el identificador de conversación correspondiente.

5.2.2. *Clase iniciadorBusca*

Esta clase implementa el rol de *Initiator* del agente BuscaAgent, es decir, desde esta clase se van a iniciar las conversaciones con WebAgent y OntoAgent. El método `handleInform` es el responsable de atender las respuestas procedentes de WebAgent y OntoAgent cuya performativa sea *Inform*. Cuando recibe un mensaje nuevo, lo inserta en la lista de mensajes pendientes.

Por cada mensaje se guarda un conjunto de datos necesarios para su procesamiento:

- Destinatario: AID del agente RestAgent que inició la consulta.
- Identificador de la conversación: Identificador de la conversación con el agente RestAgent.
- Contenido: Resultado de la consulta.
- Informador: El agente que ha proporcionado la información, WEB si es de tipo WebAgent y ONTO si es del tipo OntoAgent.

Adicionalmente, si el mensaje recibido proviene de WebAgent, añade un nuevo comportamiento al agente para que envíe dicha información al agente ActualizarOntoAgent.

5.3. RestAgent

La clase `RestAgent` hereda de la clase `jade.core.Agent`. El agente RestAgent es el responsable de interactuar directamente con el usuario. Puede hacerlo de dos formas, por línea de comando o mediante un GUI. Este último método ofrece una mayor funcionalidad. Para conseguir que RestAgent se comporte de un modo u otro, en su creación recibe uno o varios argumentos que se describen a continuación:

1. TIPO_INTERFAZ: Línea de comandos (C) o GUI (G).

2. RUTA_RESULT: Nombre del fichero XML en el que volcar el resultado de la búsqueda.
3. NOMBRE: Nombre del restaurante a buscar.

El método `Setup` registra el agente en el DF y analiza estos argumentos. Si el argumento TIPO_INTERFAZ es G, el resto de los argumentos pueden ser omitidos. Si por el contrario es C, es obligatorio introducir el argumento NOMBRE. El argumento RUTA_RESULT siempre es opcional, aunque se recomienda en el caso de que el TIPO_INTERFAZ sea C.

Cuando RestAgent recibe el nombre de un restaurante, bien desde la línea de comando en su creación o bien desde el GUI, añade el comportamiento `enviarSolicitudBusca`. Este comportamiento realiza la búsqueda de un agente de tipo BuscaAgent que pueda atender la solicitud. Si lo encuentra prepara el mensaje para iniciar la conversación con él y añade un nuevo comportamiento que va a ser el que le otorgue el rol de *Initiator* al agente, clase `iniciadorRest` que hereda de la clase `AchieveReInitiator`. El método `handleInform` de esta clase es el que va procesar la respuesta final recibida de BuscaAgent, de tal forma que la información recibida la mostrará en el GUI o en la línea de comando dependiendo de la configuración inicial del agente. En el caso de que el argumento RUTA_RESULT tenga contenido, se llama al método `generarXML` de la clase `UtilXML` que genera un fichero XML con la información obtenida.

Se ha comentado que el agente puede interactuar con el usuario a través de un GUI. Para ello, se ha implementado la clase `GUIRest`. Esta clase se vincula al agente de tal modo que los eventos generado en el GUI se transmiten al agente RestAgent para que este opere en consecuencia.

Código en GUIRest

```
rest.setName(nombre);
GuiEvent ev = new GuiEvent(null,m_agente2.ENVIAR);
ev.addParameter(rest);
m_agente2.postGuiEvent(ev);
```

Código en RestAgent

```
protected void onGuiEvent(GuiEvent ev){
        switch(ev.getType()){
            case ENVIAR:
                res = (RESTAURANT) ev.getParameter(0);
                addBehaviour(new enviarSolicitudBusca(this, res));
                      break;
```

Figura 4.7 Fragmento de código

En la figura 5.7 se muestra el código implementado en ambas clases para conseguir esta vinculación.

5.4. WebAgent

Al igual que ocurre con el resto de agentes, la clase `WebAgent` hereda de la clase `jade.core.Agent`. El agente WebAgent está especializado en obtener información de la WEB sobre un determinado restaurante. Para ello se vale de la ayuda de un *wrapper,* código generado para analizar la estructura y el contenido de una página WEB con el fin de extraer de ella determinada información. Ya se comentó, que los *wrappers* eran reutilizados de un proyecto que no se llegó a finalizar. Por ello, los agentes de tipo WebAgent solo tienen disponible un *wrapper,* la clase `Metropoli`. Este *wrapper* extrae la información de la guía Metrópoli del sitio WEB ElMundo.es (http://www.elmundo.es/metropoli/restaurantes/buscador_avanzado.html).

En su creación puede recibir varios argumentos:

1. WRAPPER: Wrapper que va a utilizar el agente. En este caso, siempre será M (Metropoli).
2. COMBINACION: Argumento opcional. Se indica el nombre del fichero CSV que se genera con las mediciones de los tiempos de búsqueda.

Desde el método `Setup` se registra el agente en el DF y se añade el comportamiento que le va a otorgar el rol de *Responder* en la conversación que mantiene con BuscaAgent, la clase `respuestaWeb` que hereda de la clase `AchieveREResponder`.

WebAgent hace la llamada al *wrapper* para obtener la información sobre el restaurante por el que BuscaAgent le preguntó en el método `prepareResultNotification` de la clase `respuestaWeb` (figura 5.8).

```
List listado = new ArrayList();
if (m_agente.WRAPPER.equalsIgnoreCase("M")){
    try{
        listado = datos.Metropoli(res.getName());
    }catch (NullPointerException e) {
        e.printStackTrace();
    }
}
```

Figura 5.8 Fragmento de código

5.5. OntoAgent

Como el resto de agentes, la clase `OntoAgent` hereda de la clase `jade.core.Agent`. El agente OntoAgent está especializado en buscar información sobre el restaurante solicitado en un fichero con información de restaurantes almacenados según el formato de la ontología. El encargado de generar dichos ficheros es el agente ActualizarOntoAgent.

En su creación puede recibir varios argumentos:

1. FICHERO_ONTOLOGIA: Nombre del fichero de la ontología en el que va a realizar las búsquedas.
2. COMBINACION: Argumento opcional. Se indica el nombre del fichero CSV que se genera con las mediciones de los tiempos de búsqueda.

En el método `Setup` se registra el agente en el DF y se añade el comportamiento que le va a otorgar el rol de *Responder* en la conversación que mantiene con BuscaAgent, la clase `respuestaOnto` que hereda de la clase `AchieveREResponder`.

En el método `prepareResultNotification` de la clase `respuestaOnto` se realiza la llamada al método `Fichero` de la clase `ListaRestaurante` (figura 5.9), el cuál va a realizar la búsqueda de información del restaurante solicitado.

```
ListaRestaurantes datos = new ListaRestaurantes();
List listado = new ArrayList();
listado = datos.Fichero(m_agente.FICHERO_ONTOLOGIA,"Name",res.getName());
```

Figura 5.9 Fragmento de código

5.6. ActualizarOntoAgent

El agente ActualizarOntoAgent es el agente encargado de gestionar el fichero con la información de restaurantes guardados con el formato de la ontología. La clase `ActualizarOntoAgent` hereda también de la clase `jade.core.Agent`. En su creación el agente puede recibir hasta dos argumentos:

1. FICHERO_ONTOLOGIA: Nombre del fichero de la ontología que se va a mantener.

2. TIME: Argumento opcional. Tiempo en milisegundos para indicar cada cuanto tiempo se va a ejecutar el comportamiento `actualizarOntoAuto`.

En el método `Setup` de esta clase, se registra el agente en el DF y se añaden los comportamientos que va a tener el agente, `actualizarRestaurante` y `actualizarOntoAuto`. Este último siempre y cuando en la creación del agente se haya indicado un valor para el argumento TIME. Si no se ha introducido ningún valor para dicho argumento, es necesario que el fichero indicado en FICHERO_ONTOLOGIA exista, ya que la única función del agente será actualizar dicho fichero.

La clase `actualizarRestaurante` hereda de la clase `jade.core.behaviours.SimpleBehaviour`. Cuando este comportamiento es añadido, se le facilita una plantilla de mensaje (figura 5.10). Los mensajes recibidos deben de cumplir la condición indicada en la plantilla. En este caso se espera un mensaje con una performativa INFORM

```
MessageTemplate plantilla2 = MessageTemplate.MatchPerformative(ACLMessage.INFORM);
addBehaviour(new  actualizarRestaurante(this,plantilla2));
```

Figura 5.10 Fragmento de código

Cuando se recibe la información de un restaurante, se comprueba si ya existe una instancia del restaurante en el fichero de la ontología. En caso contrario, se añade la nueva información.

La clase `actualizarOntoAuto` hereda de la clase `jade.core.behaviours.TickerBehaviour`. De este modo, este comportamiento se ejecutará de forma cíclica cada cierto tiempo. En este caso, el que se haya indicado en la creación del agente en el argumento TIME. Este comportamiento va a llamar a un método de la clase `ListaRestaurantes`, el cuál va a realizar una búsqueda automática de restaurantes y los va a almacenar con el formato de la ontología en el fichero indicado. Como se ha mencionado anteriormente, únicamente está desarrollado el *wrapper* `Metropoli`, por lo tanto, en este caso se hace la llamada al método `TodoMetropoli`. Dado que el fichero se va actualizando con las informaciones que el agente recibe de BuscaAgent, no se requiere ejecutar esta tarea con mucha frecuencia. En el Anexo 3, se muestra un ejemplo de la estructura que tiene el fichero con los datos almacenados con el formato de la ontología.

5.7. Estructura del Proyecto

En los apartados anteriores de este capítulo se ha explicado el código principal de este proyecto, es decir, el código de cada agente. Ahora bien, para que el código que implementa cada agente sea más legible, se ha desarrollado un conjunto de clases con el fin de proporcionar ciertas facilidades a la hora de implementar este proyecto. A continuación se detalla la organización de los paquetes con sus clases y una breve descripción:

- Paquete `proyecto.restaurantes`: Contiene las clases principales de este proyecto las cuales han sido descritas anteriormente.
  - `ActualizarOntoAgent.java`
  - `BuscaAgent.java`
  - `GUIRest.java`
  - `OntoAgent.java`
  - `RestAgent.java`
  - `WebAgent.java`
- Paquete `proyecto.wrapper`: Contiene todas las clases relacionadas con el *wrapper* y la generación del fichero con el formato de la ontología.
  - `FicheroOnto.java`
  - `ListaRestaurantes.java`
  - `Metropoli.java`
  - `pagina.java`
- Paquete `proyecto.utilidades`: Contiene las clases que implementan las utilidades desarrolladas.
  - Traza.java: Clase desarrollada para medir los tiempos de cada búsqueda y otros datos de interés para su posterior análisis.
  - ListaTraza.java: Clase que proporciona utilidades para la gestión de la lista de trazas.
  - UtilFichero.java: Clase que proporciona los métodos para la gestión del fichero en el que se guardan las trazas recogidas.
  - MSG_PENDIENTES.java: Clase que utiliza BuscaAgent para la gestión de los mensajes provenientes de OntoAgent y WebAgent (ver apartado 4.2.2).

- Util.java: Esta clase principalmente implementa métodos para la gestión de los agentes con el DF.
- UtilXml.java: Clase que proporciona los métodos necesarios para la creación del fichero XML con el resultado de la búsqueda de un restaurante.

- Paquete `proyecto.onto_servicios`: Contiene las clases de la ontología.

5.8. Manual de Usuario

Los requisitos previos para poder ejecutar el SMA implementado son tener instalada una máquina virtual de Java versión 5 o superior y JADE (ver apartado 3.4).

Para poner en funcionamiento el SMA implementado hay que lanzar los agentes que lo componen tal y como se explica en el apartado 3.4. Se pueden lanzar todos a la vez, o bien de uno en uno. Para mayor comodidad, el o los comandos para lanzar el SMA de pueden almacenar en un fichero (con extensión .bat o .cmd). En la figura 5.11 se muestra un ejemplo que lanzaría un agente de cada tipo y lanzaría a la vez el RMA de JADE.

```
java jade.Boot –gui
busca:proyecto.restaurantes.BuscaAgent(P,ONTO,3,0);Onto:proyecto.restaurantes.OntoAge
nt(C:\\PFC\metropoli.txt);Web:proyecto.restaurantes.WebAgent(M);ActualizarOnto:proyec
to.restaurantes.ActualizarOntoAgent("C:\\PFC\\metropoli.txt");restGUI:proyecto.restau
rantes.RestAgent(G)
```

Figura 5.11 Ejemplo de comando para lanzar el SMA

Como se puede observar, cada agente recibe diferentes argumentos, los cuáles han sido ya descritos en detalle en los apartados anteriores de este capítulo.

Siempre debe existir al menos un agente de tipo BuscaAgent, ya que como se ha mencionado anteriormente es el núcleo del sistema y sin él el resto de los agentes carecen de sentido, y un agente de tipo RestAgent para recibir las peticiones. Al menos debe existir un agente de tipo OntoAgent o WebAgent, para garantizar un mínimo de búsqueda con resultado satisfactorio. El agente ActualizarOntoAgent es opcional.

Una vez que el SMA está funcionando, vamos a explicar cómo el usuario interactúa con él, es decir, cómo el usuario interactúa con el agente RestAgent. Hay que recordar que este agente puede recibir las peticiones del usuario a través de un GUI o directamente en su creación. En este último caso, el nombre del restaurante a buscar se pasa como argumento y el resultado obtenido se vuelca en un fichero xml. Sólo permite una única consulta. En la figura 5.12 se muestra un ejemplo que lanza un agente de tipo RestAgent sin GUI.

```
java jade.Boot -container restCMD:proyecto.restaurantes.RestAgent(C,resultado.xml,"La Dorada")
```

Figura 5. 12 Ejemplo de comando para lanzar agente RestAgent

Si el agente RestAgent ha sido creado para interactuar con el usuario mediante un GUI, el usuario podrá realizar múltiples consultas. El GUI de esta aplicación es muy sencillo, no obstante, vamos a explicar su funcionamiento.

La figura 5.13 muestra el GUI cuando se inicia.

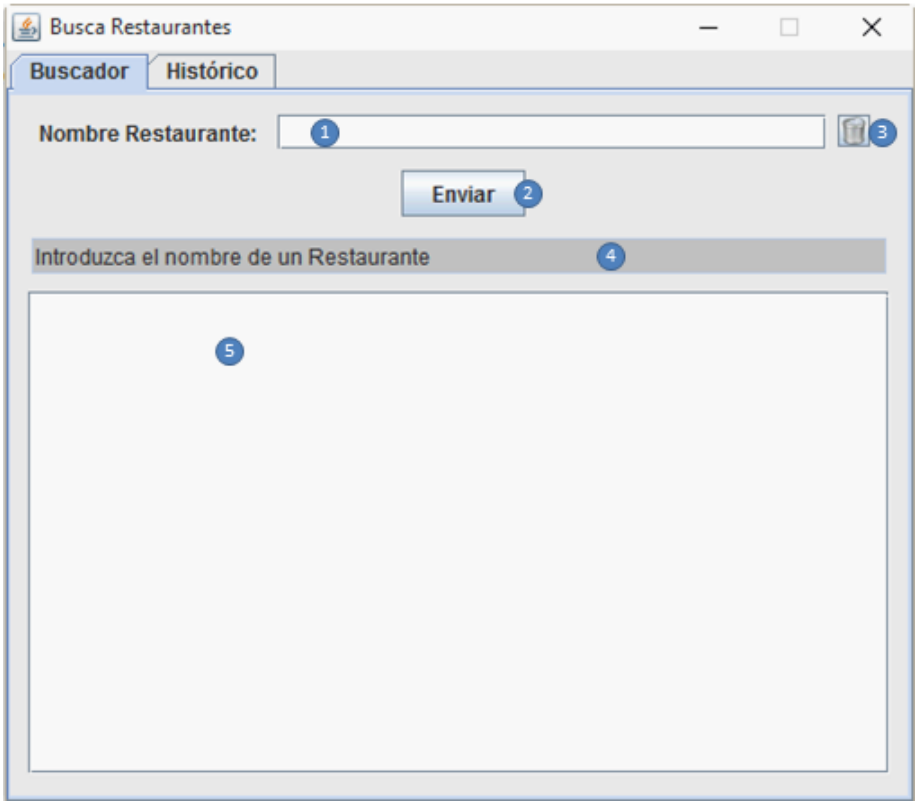


Figura 5.13 GUI de RestAgent

El GUI tiene dos pestañas:

- **Buscador**: En esta pestaña se realiza la búsqueda de información de restaurantes. Los elementos que la componen son:
    1. Nombre Restaurante: Aquí se introduce el nombre del restaurante del cual se desea obtener información.
    2. Botón Enviar: Envía la solicitud de búsqueda
    3. Botón Borrar: Si se pulsa una vez, borra el nombre del restaurante y se prepara para una nueva búsqueda. Si se pulsa dos veces, se borra el contenido del Panel de Información y se pasa al Histórico.
    4. Panel de Mensajes: Muestra información de ayuda.
    5. Panel de Información: Muestra el resultado de la búsqueda. La figura 5.14 muestra un ejemplo en el que la información recibida se muestra en el panel de información.

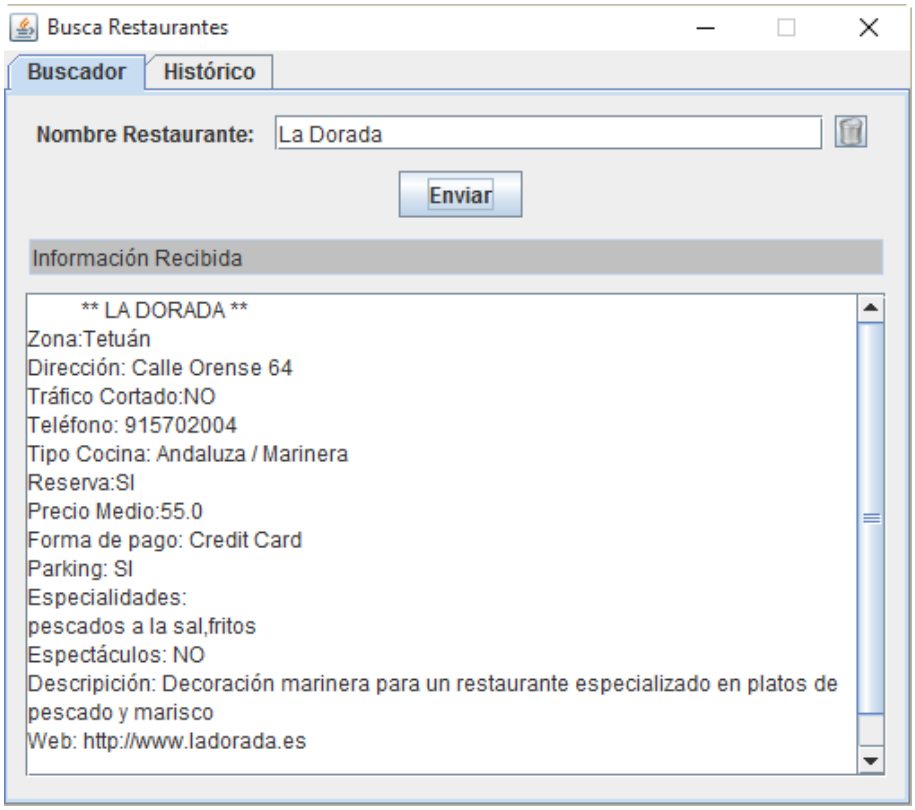


Figura 5.14 GUI resultado de búsqueda

- **Histórico:** En esta pestaña se muestra el histórico de resultados de las búsquedas realizadas. La figura 5.15 muestra un ejemplo de la pestaña histórico.

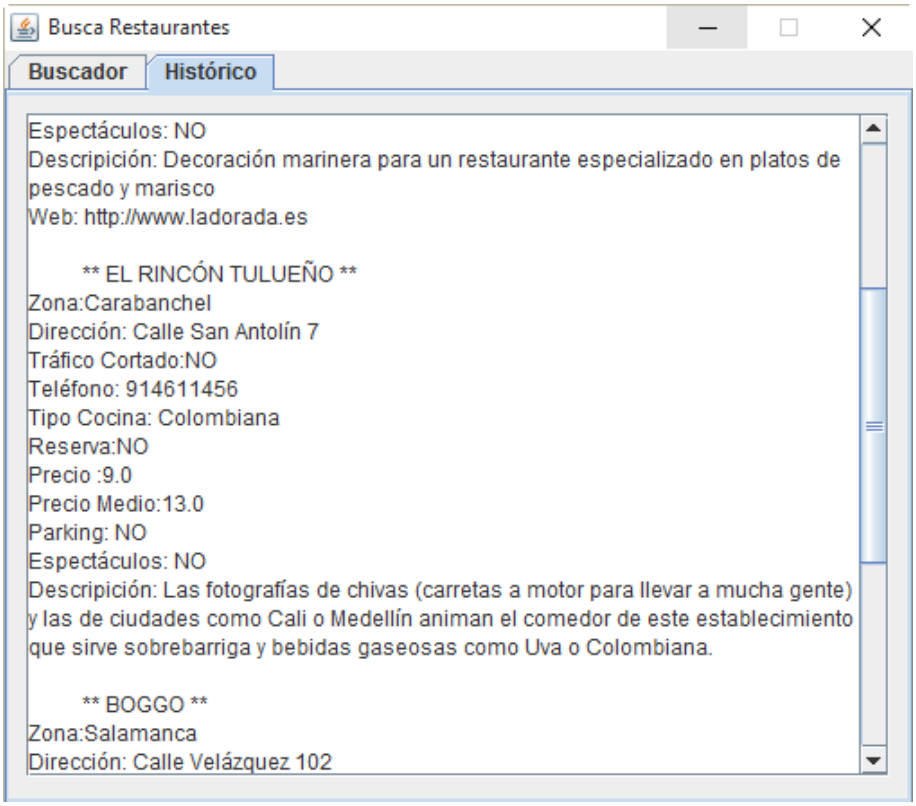


Figura 5.15 GUI histórico de búsquedas

Capítulo 6. Estudio y Resultados

En este capítulo se realiza un estudio para analizar las diferentes configuraciones del agente BuscaAgent y se exponen varios ejemplos de consultas con sus correspondientes resultados.

6.1. Estudio Realizado

En el capítulo anterior se ha descrito en detalle el SMA implementado y se ha mencionado que el agente BuscaAgent, agente principal del SMA, puede funcionar con múltiples configuraciones. Ahora bien, ¿qué configuración es la más adecuada? Para tratar de responder a esta respuesta se ha realizado un estudio registrando diversas variables para la búsqueda de un determinado restaurante con las distintas configuraciones posibles.

El agente BuscaAgent recibe varios argumentos: TIPO_BUSQUEDA, ORDEN, ESTRATEGIA y TODOS. Teniendo en cuenta todos los argumentos salen un total de 24 configuraciones distintas, pero para simplificar el estudio, a la hora de tomar las mediciones en el SMA sólo se ha registrado un agente de cada tipo; por lo que el último argumento se puede descartar. Así pues, para la realización del estudio se va a tener en cuenta, 3 variables dependientes (TIPO_BUSQUEDA, ORDEN, ESTRATEGIA) reduciendo el número de combinaciones a 12 (P-ONTO-1, P-ONTO-2, P-ONTO-3, P-WEB-1, P-WEB-2, P-WEB-3, S-ONTO-1, S-ONTO-2, S-ONTO-3, S-WEB-1, S-WEB-2, S-WEB-3).

Para realizar el estudio se han seleccionado un total de 20 restaurantes, todos ellos existentes inicialmente en la ontología:

- Sicilia In Bocca
- El Rincón Tulueño
- Casa Gallega
- Cuando salí de Cuba

- El Bogavante de Almirante
- El Cosaco
- Entretacos
- La Lumbre
- Las Reses
- Sergi Arola Gastro
- El Cortijillo
- El Gamo
- Kobe
- Chicago's
- La Pesquera
- Portonovo
- El Independiente
- Boggo
- La Dorada
- Don Jamibe

De esta selección, los 10 siguientes han sido eliminados de la ontología antes de la toma de datos:

- El Rincón Tulueño
- Cuando salí de Cuba
- El Cosaco
- La Lumbre
- Sergi Arola Gastro
- El Gamo
- Chicago's
- Portonovo
- Boggo
- Don Jamibe

Por cada restaurante y configuración se han realizado tres consultas, para obtener un valor medio de los parámetros registrados. A continuación se

muestra un gráfico que representa los tiempos medios de búsqueda de cada restaurante para cada configuración.

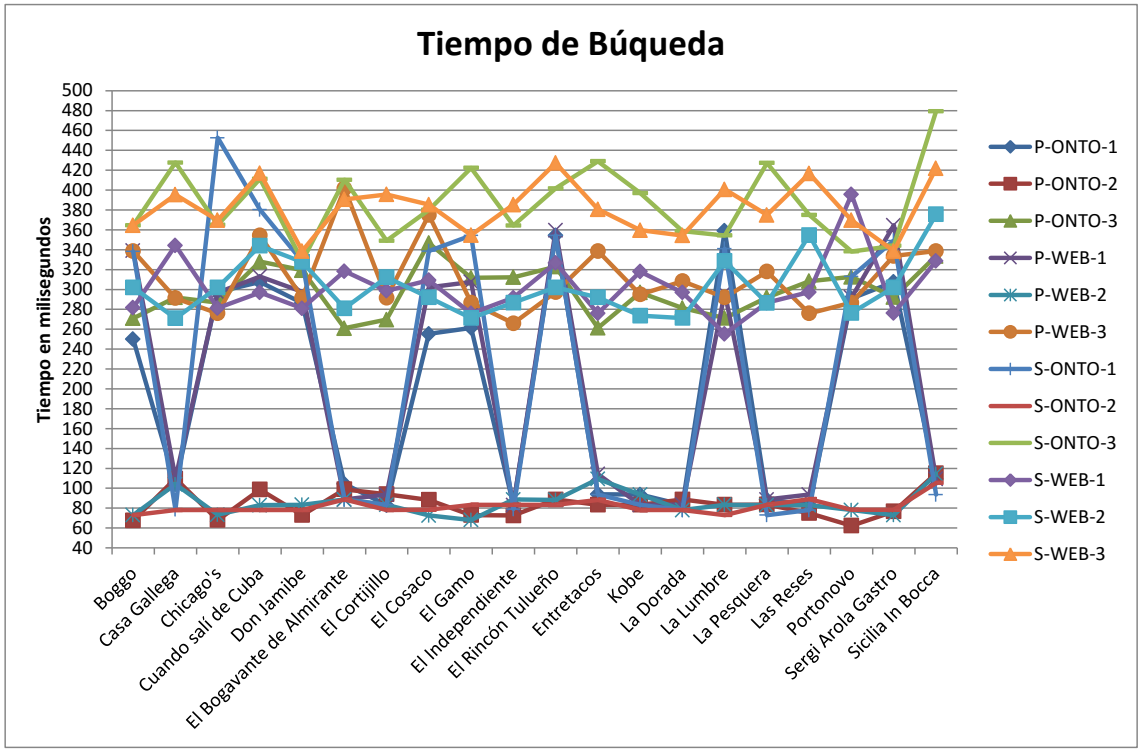


Figura 6.1 Gráfico Tiempo de Búsqueda

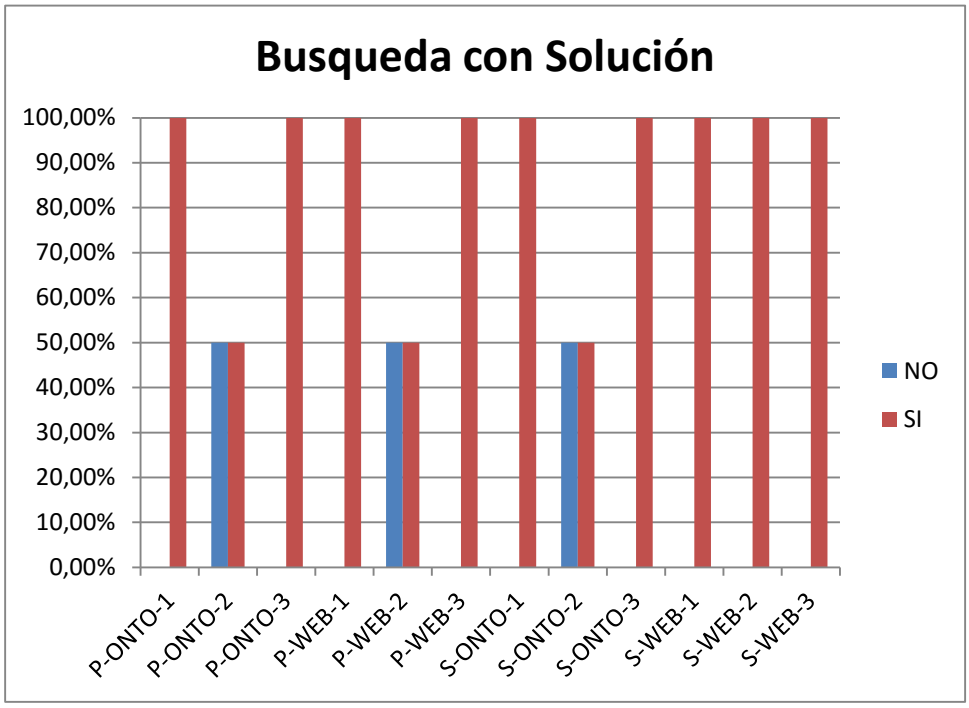


Figura 6.2 Gráfico porcentaje Solución

En él gráfico de la figura 6.1 se observa que casi todos los tiempos de búsqueda oscilan entre los 60 y 480 milisegundos. Las configuraciones P-ONTO-2, P-WEB-2 y S-ONTO-2 destacan, ya que presentan tiempos de búsqueda siempre por debajo de los 120 milisegundos. ¿Quiere decir esto que son las mejores configuraciones? No, este gráfico representa los tiempos medios de búsqueda con independencia de que el resultado de la búsqueda haya devuelto la información solicitada o no. Ya que la diferencia de los tiempos de búsqueda no es muy grande, una variable que ayude a decidir cuáles son las mejores configuraciones puede ser esta. En el gráfico de la figura 6.2 se muestra para cada configuración qué porcentaje de respuestas contiene la información solicitada o no.

Aunque por los tiempos medios de búsqueda parecía que P-ONTO-2, P-WEB-2 y S-ONTO-2 era las mejores opciones de configuración, los datos del gráfico indican lo contrario. Volviendo al gráfico Tiempo de Búsqueda (figura 6.1), se observa que las configuraciones S-ONTO-3 y S-WEB-3 tienen unos tiempos medios de búsqueda superior al resto, por lo que se pueden descartar también. De este modo, las 12 configuraciones iniciales se han reducido a 7: P-ONTO-1, P-ONTO-3, P-WEB-1, P-WEB-3, S-ONTO-1, S-WEB-1 y S-WEB-2. El siguiente gráfico muestra el promedio del tiempo de búsqueda para cada una de estas configuraciones.

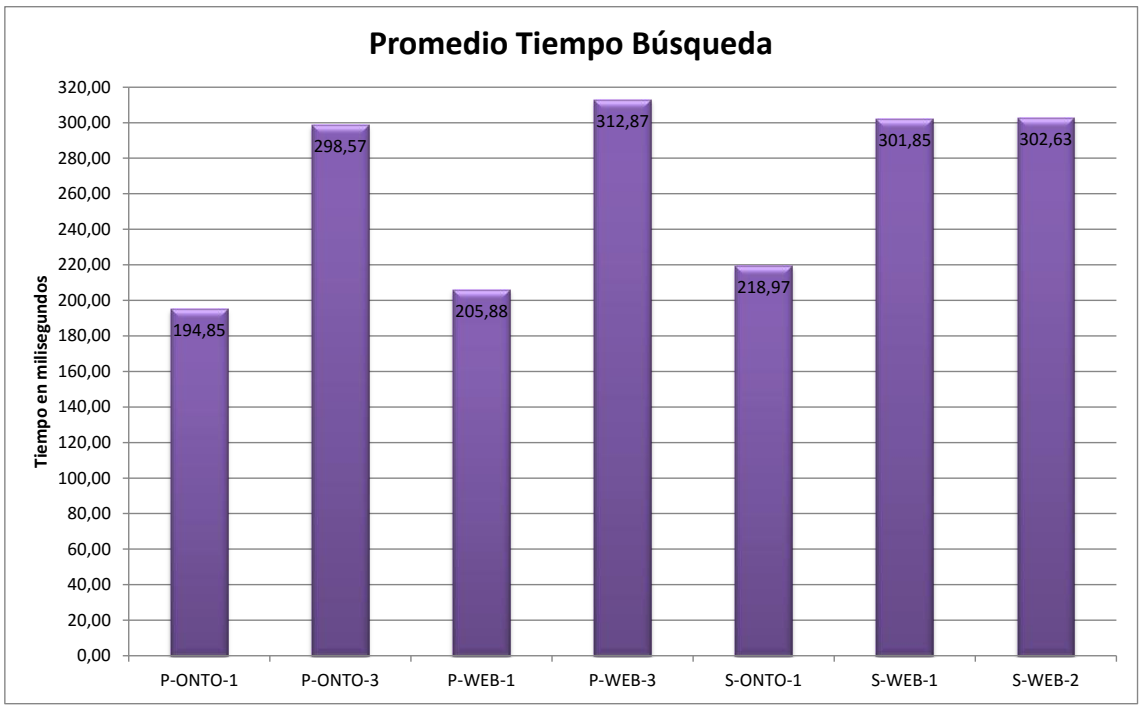


Figura 6.3 Gráfico Promedio Tiempo Búsqueda

Se observa claramente en el gráfico de la figura 6.3 que las configuraciones que mejor promedio de búsqueda tienen son P-ONTO-1 y P-WEB-1 seguida de S-ONTO-1, por tanto, se puede concluir que estas tres configuraciones son la mejores.

Analizando las características de estas tres configuraciones y teniendo en cuenta los datos obtenidos en este estudio, se comprueba lo que ya esperábamos:

1. Una configuración en paralelo es más efectiva que una secuencial.
2. Siempre que exista la posibilidad de que la información sea proporcionada por un agente OntoAgent, el tiempo medio de búsqueda será menor.
3. Sin lugar a dudas, la mejor estrategia es la estrategia 1; es decir, si el primer mensaje recibido ya contiene la información solicitada, se envía como respuesta final sin esperar un segundo mensaje, mejorando así el tiempo de respuesta.

6.2. Resultados

En este apartado se exponen varios ejemplos de consultas con los resultados obtenidos en cada una de ellas. Para cada consulta se refleja la petición realizada, el resultado obtenido, la captura obtenida con el agente Sniffer que muestra el intercambio de mensajes entre los agentes y el contenido de dichos mensajes.

En el momento de realizar las consultas en el SMA se encontraban registrados un agente BuscaAgente, dos agentes OntoAgent, dos agentes WebAgent, y el agente RestAgent correspondiente.

6.2.1. *Consulta 1*

Se solicita información sobre el restaurante “El Cosaco” a través del GUI proporcionado por el agente RestAgent. La configuración del agente BuscaAgent cuando se realizaba esta consulta era S-WEB-2.

La figura 6.4 muestra la petición realizada por el usuario y el resultado obtenido. La captura de la traza obtenida por el agente Sniffer se muestra a continuación en la figura 6.5.

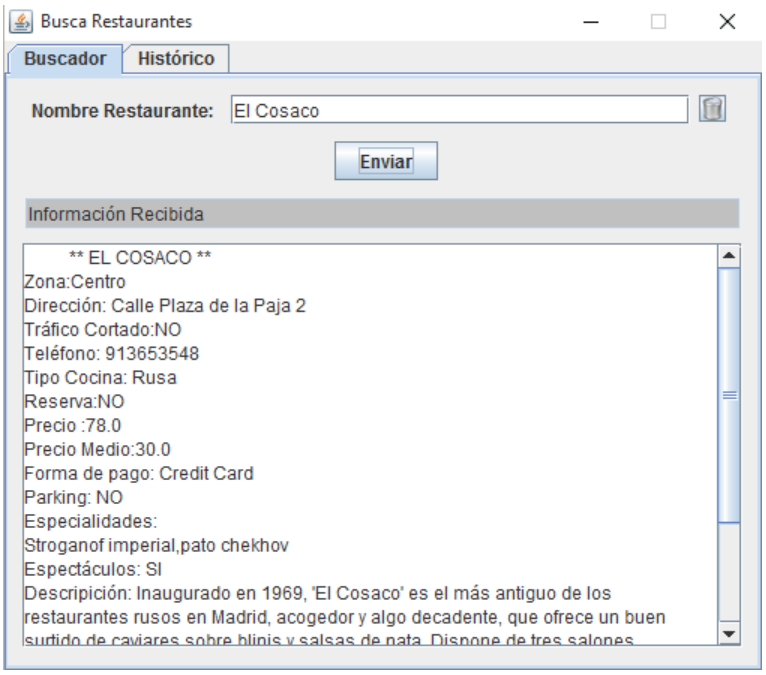


Figura 6.4 Resultado Consulta 1

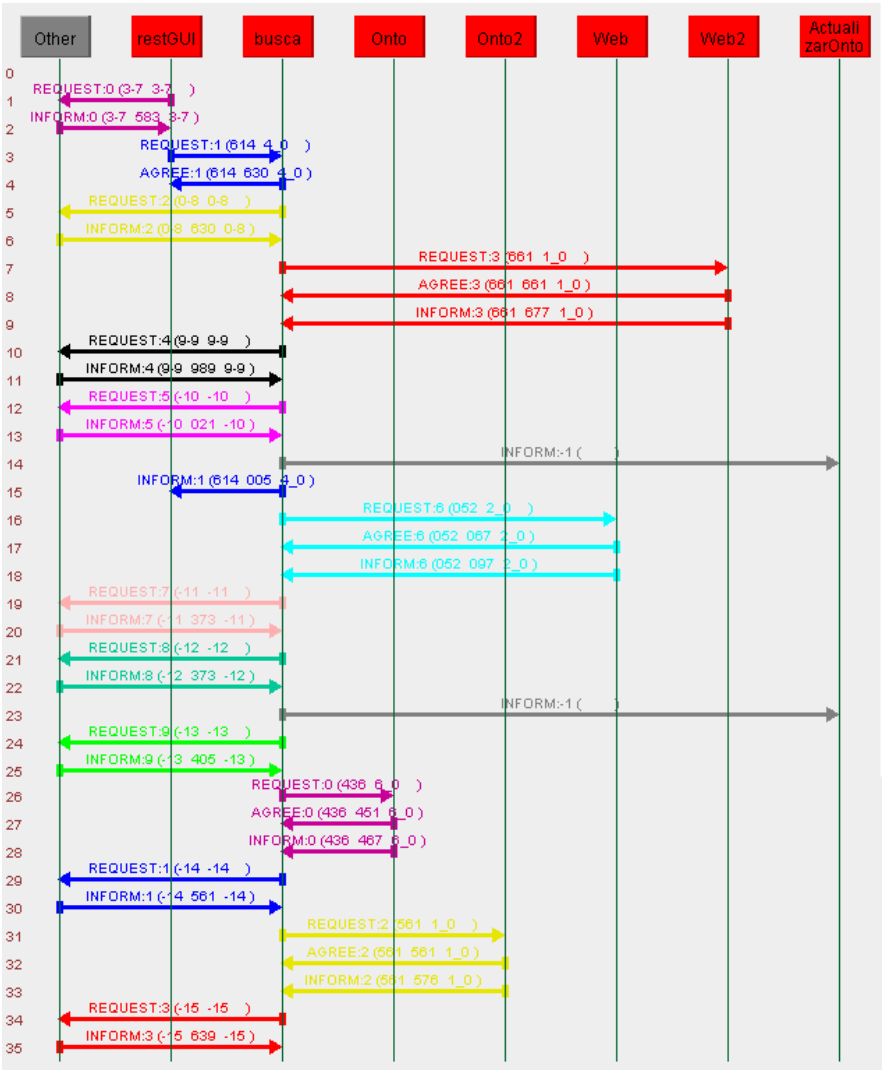


Figura 6.5 Captura Sniffer Consulta 1

La traza muestra el intercambio de mensajes producidos entre los diferentes agentes del SMA para conseguir la información solicitada. En ella se refleja que las peticiones se han realizado de forma secuencial y primeramente se han enviado a los agentes de tipo WebAgent. Como está configurada la estrategia 2, se envía el primer mensaje que reciba con independencia del resultado. En la figura 6.6 se muestra el detalle de los mensajes.

(**REQUEST**
:sender (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443540035614_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C1672562165_1443540035614)

(**AGREE**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:reply-with restGUI@192.168.1.231:1099/JADE1443540035630 :in-reply-to R1443540035614_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C1672562165_1443540035614)

(**REQUEST**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443540035661_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C209141991_1443540035661)

(**AGREE**
:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:reply-with busca@192.168.1.231:1099/JADE1443540035661 :in-reply-to R1443540035661_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C209141991_1443540035661)

(**INFORM**
:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((result (action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"913653548\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restauranteelcosaco.com/ :Fall_Probability 0.0)) :Price 78.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Plaza de la Paja\" :Street_Type Calle) :Traffic_Cut false) :Number 2) :Payment_Form (sequence \"Credit Card\") :Name \"El Cosaco\" :Description \"Inaugurado en 1969, 'El Cosaco' es el m?s antiguo de los restaurantes rusos en Madrid, acogedor y algo decadente, que ofrece un buen surtido de caviares sobre blinis y salsas de nata. Dispone de tres salones peque?os con seis mesas cada uno en un ambiente rom?ntico. Para los que tengan gustos culinarios atrevidos: prueben la tradicional y muy rusa sopa fr?a de yogur con huevo, pepino, cebolleta, eneldo y pasas del Cosaco. Conviene reservar los fines de semana.\" :Cuisine_type Rusa :Performance true

:Parking false :Specialities (sequence \"Stroganof imperial\" \"pato chekhov\" \"\") :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443540035677 :in-reply-to R1443540035661_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C209141991_1443540035661)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"913653548\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restauranteelcosaco.com/ :Fall_Probability 0.0)) :Price 78.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Plaza de la Paja\" :Street_Type Calle) :Traffic_Cut false) :Number 2) :Payment_Form (sequence \"Credit Card\") :Name \"El Cosaco\" :Description \"Inaugurado en 1969, 'El Cosaco' es el m?s antiguo de los restaurantes rusos en Madrid, acogedor y algo decadente, que ofrece un buen surtido de caviares sobre blinis y salsas de nata. Dispone de tres salones peque?os con seis mesas cada uno en un ambiente rom?ntico. Para los que tengan gustos culinarios atrevidos: prueben la tradicional y muy rusa sopa fr?a de yogur con huevo, pepino, cebolleta, eneldo y pasas del Cosaco. Conviene reservar los fines de semana.\" :Cuisine_type Rusa :Performance true :Parking false :Specialities (sequence \"Stroganof imperial\" \"pato chekhov\" \"\") :Category 0 :Reserve false :Average_Price 30.0))))"

:language fipa-sl :ontology ontology)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence ENCONTRADO (RESTAURANT :Telephone \"913653548\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restauranteelcosaco.com/ :Fall_Probability 0.0)) :Price 78.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Plaza de la Paja\" :Street_Type Calle) :Traffic_Cut false) :Number 2) :Payment_Form (sequence \"Credit Card\") :Name \"El Cosaco\" :Description \"Inaugurado en 1969, 'El Cosaco' es el m?s antiguo de los restaurantes rusos en Madrid, acogedor y algo decadente, que ofrece un buen surtido de caviares sobre blinis y salsas de nata. Dispone de tres salones peque?os con seis mesas cada uno en un ambiente rom?ntico. Para los que tengan gustos culinarios atrevidos: prueben la tradicional y muy rusa sopa fr?a de yogur con huevo, pepino, cebolleta, eneldo y pasas del Cosaco. Conviene reservar los fines de semana.\" :Cuisine_type Rusa :Performance true :Parking false :Specialities (sequence \"Stroganof imperial\" \"pato chekhov\" \"\") :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with restGUI@192.168.1.231:1099/JADE1443540036005 :in-reply-to R1443540035614_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1672562165_1443540035614)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443540036052_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C915569520_1443540036052)

(**AGREE**

:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443540036067 :in-reply-to R1443540036052_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C915569520_1443540036052)

(**INFORM**

:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"913653548\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restauranteelcosaco.com/ :Fall_Probability 0.0)) :Price 78.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Plaza de la Paja\" :Street_Type Calle) :Traffic_Cut false) :Number 2) :Payment_Form (sequence \"Credit Card\") :Name \"El Cosaco\" :Description \"Inaugurado en 1969, 'El Cosaco' es el m?s antiguo de los restaurantes rusos en Madrid, acogedor y algo decadente, que ofrece un buen surtido de caviares sobre blinis y salsas de nata. Dispone de tres salones peque?os con seis mesas cada uno en un ambiente rom?ntico. Para los que tengan gustos culinarios atrevidos: prueben la tradicional y muy rusa sopa fr?a de yogur con huevo, pepino, cebolleta, eneldo y pasas del Cosaco. Conviene reservar los fines de semana.\" :Cuisine_type Rusa :Performance true :Parking false :Specialities (sequence \"Stroganof imperial\" \"pato chekhov\" \"\") :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443540036097 :in-reply-to R1443540036052_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C915569520_1443540036052)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"913653548\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restauranteelcosaco.com/ :Fall_Probability 0.0)) :Price 78.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Plaza de la Paja\" :Street_Type Calle) :Traffic_Cut false) :Number 2) :Payment_Form (sequence \"Credit Card\") :Name \"El Cosaco\" :Description \"Inaugurado en 1969, 'El Cosaco' es el m?s antiguo de los restaurantes rusos en Madrid, acogedor y algo decadente, que ofrece un buen surtido de caviares sobre blinis y salsas de nata. Dispone de tres salones peque?os con seis mesas cada uno en un ambiente rom?ntico. Para los que tengan gustos culinarios atrevidos: prueben la tradicional y muy rusa sopa fr?a de yogur con huevo, pepino, cebolleta, eneldo y pasas del Cosaco. Conviene reservar los fines de semana.\" :Cuisine_type Rusa :Performance true :Parking false :Specialities (sequence \"Stroganof imperial\" \"pato chekhov\" \"\") :Category 0 :Reserve false :Average_Price 30.0))))"

:language fipa-sl :ontology ontology)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443540036436_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1531086769_1443540036436)

(**AGREE**

:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443540036451 :in-reply-to R1443540036436_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1531086769_1443540036436)

(**INFORM**

:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone

\"913653548\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restauranteelcosaco.com/ :Fall_Probability 0.0)) :Price 78.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Plaza de la Paja\" :Street_Type Calle) :Traffic_Cut false) :Number 2) :Payment_Form (sequence \"Credit Card\") :Name \"El Cosaco\" :Description \"Inaugurado en 1969, 'El Cosaco' es el m?s antiguo de los restaurantes rusos en Madrid, acogedor y algo decadente, que ofrece un buen surtido de caviares sobre blinis y salsas de nata. Dispone de tres salones peque?os con seis mesas cada uno en un ambiente rom?ntico. Para los que tengan gustos culinarios atrevidos: prueben la tradicional y muy rusa sopa fr?a de yogur con huevo, pepino, cebolleta, eneldo y pasas del Cosaco. Conviene reservar los fines de semana.\" :Cuisine_type Rusa :Performance true :Parking false :Specialities (sequence \"Stroganof imperial\" \"pato chekhov\") :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443540036467 :in-reply-to R1443540036436_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1531086769_1443540036436)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443540036561_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1826268229_1443540036561)

(**AGREE**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443540036561 :in-reply-to R1443540036561_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1826268229_1443540036561)

(**INFORM**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"El Cosaco\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"913653548\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restauranteelcosaco.com/ :Fall_Probability 0.0)) :Price 78.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Plaza de la Paja\" :Street_Type Calle) :Traffic_Cut false) :Number 2) :Payment_Form (sequence \"Credit Card\") :Name \"El Cosaco\" :Description \"Inaugurado en 1969, 'El Cosaco' es el m?s antiguo de los restaurantes rusos en Madrid, acogedor y algo decadente, que ofrece un buen surtido de caviares sobre blinis y salsas de nata. Dispone de tres salones peque?os con seis mesas cada uno en un ambiente rom?ntico. Para los que tengan gustos culinarios atrevidos: prueben la tradicional y muy rusa sopa fr?a de yogur con huevo, pepino, cebolleta, eneldo y pasas del Cosaco. Conviene reservar los fines de semana.\" :Cuisine_type Rusa :Performance true :Parking false :Specialities (sequence \"Stroganof imperial\" \"pato chekhov\") :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443540036576 :in-reply-to R1443540036561_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1826268229_1443540036561)

Figura 6.6 Mensajes Consulta 1

6.2.2. *Consulta 2*

Se solicita información sobre el restaurante “Cuando salí de Cuba” a través del GUI proporcionado por el agente RestAgent. La configuración del agente BuscaAgent cuando se realizaba esta consulta era S-ONTO-1.

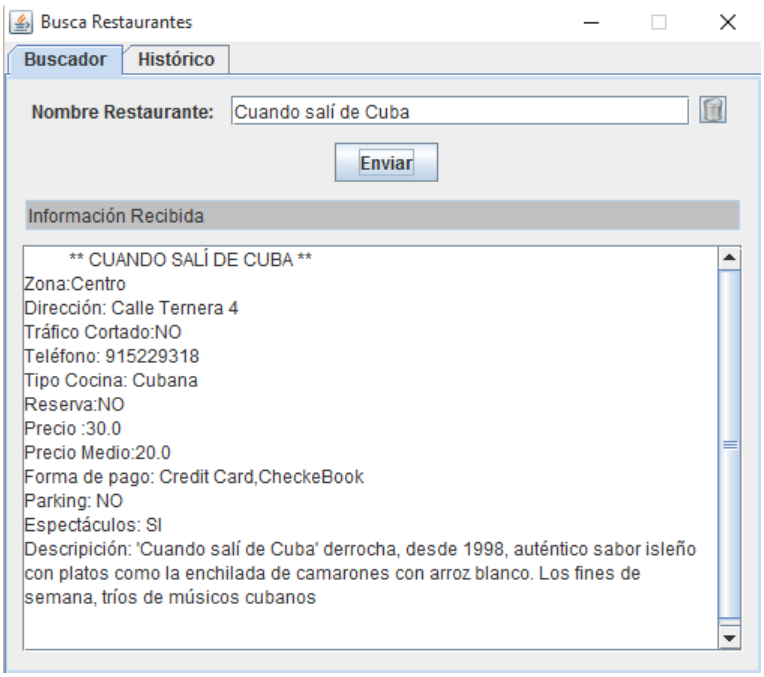


Figura 6.7 Resultado Consulta 2

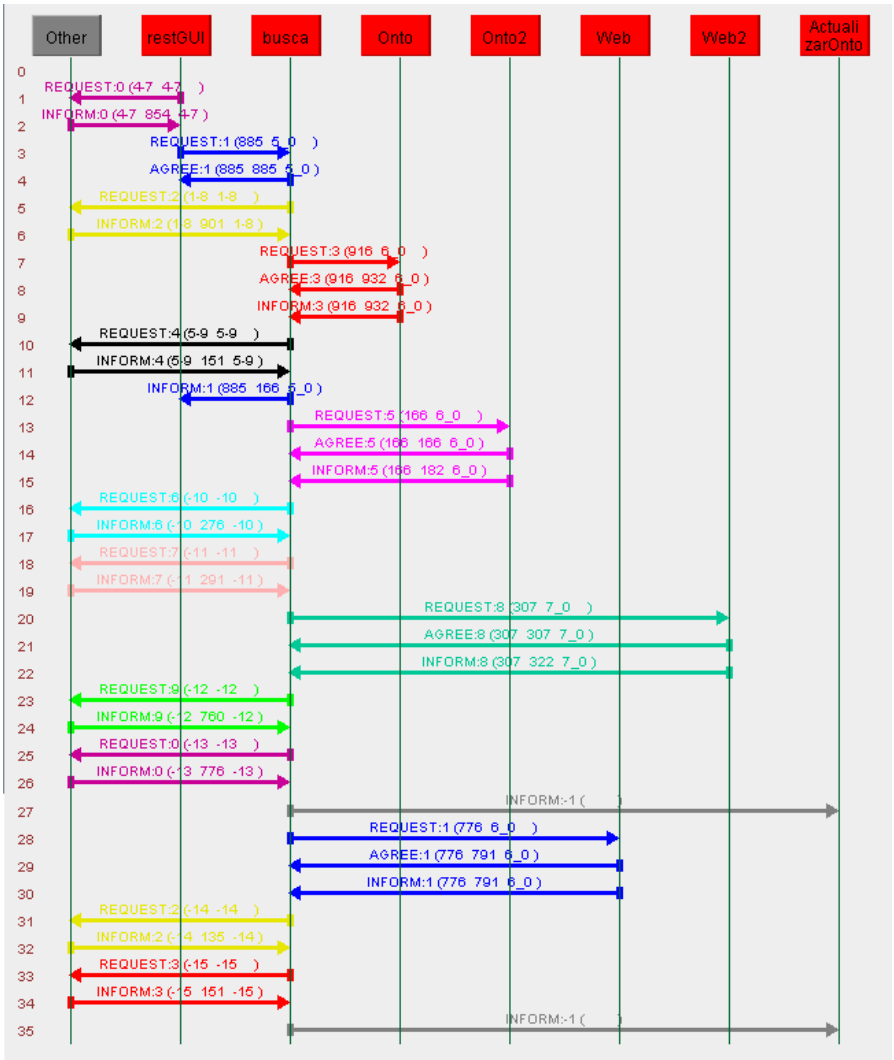


Figura 6.8 Captura Sniffer Consulta 2

La figura 6.7 muestra la petición y el resultado obtenido. La captura de la traza obtenida por el agente Sniffer se muestra en la figura 6.8, donde se aprecia el intercambio de mensajes producidos entre los distintos agentes del SMA para conseguir la información solicitada. En ella se refleja que las peticiones se han realizado de forma secuencial y primeramente se han enviado a los agentes de tipo OntoAgent. Se observa que en este caso también ha seleccionado el primer mensaje como definitivo. Esto ha ocurrido porque el primer mensaje ya contenía la información solicitada, en caso contrario, hubiera esperado a consultar las siguientes respuestas recibidas.

A continuación en la figura 6.9 se muestra el detalle de los mismos.

(**REQUEST**
:sender (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443543205885_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C2028524776_1443543205885)
(**AGREE**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:reply-with restGUI@192.168.1.231:1099/JADE1443543205885 :in-reply-to R1443543205885_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C2028524776_1443543205885)
(**REQUEST**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443543205916_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C257805244_1443543205916)
(**AGREE**
:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:reply-with busca@192.168.1.231:1099/JADE1443543205932 :in-reply-to R1443543205916_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C257805244_1443543205916)
(**INFORM**
:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((result (action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone

\"915229318\" :Zone Centro :Price 30.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name Ternera :Street_Type Calle) :Traffic_Cut false) :Number 4) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name \"Cuando sal? de Cuba\" :Description \"'Cuando sal? de Cuba' derrocha, desde 1998, aut?ntico sabor isle?o con platos como la enchilada de camarones con arroz blanco. Los fines de semana, tr?os de m?sicos cubanos \" :Cuisine_type Cubana :Performance true :Parking false :Category 0 :Reserve false :Average_Price 20.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443543205932 :in-reply-to R1443543205916_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C257805244_1443543205916)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence ENCONTRADO (RESTAURANT :Telephone \"915229318\" :Zone Centro :Price 30.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name Ternera :Street_Type Calle) :Traffic_Cut false) :Number 4) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name \"Cuando sal? de Cuba\" :Description \"'Cuando sal? de Cuba' derrocha, desde 1998, aut?ntico sabor isle?o con platos como la enchilada de camarones con arroz blanco. Los fines de semana, tr?os de m?sicos cubanos \" :Cuisine_type Cubana :Performance true :Parking false :Category 0 :Reserve false :Average_Price 20.0))))"

:reply-with restGUI@192.168.1.231:1099/JADE1443543206166 :in-reply-to R1443543205885_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C2028524776_1443543205885)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443543206166_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1438932900_1443543206166)

(**AGREE**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443543206166 :in-reply-to R1443543206166_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1438932900_1443543206166)

(**INFORM**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915229318\" :Zone Centro :Price 30.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name Ternera :Street_Type Calle) :Traffic_Cut false) :Number 4) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name \"Cuando sal? de Cuba\" :Description \"'Cuando sal? de Cuba' derrocha, desde 1998, aut?ntico sabor isle?o con platos como la enchilada de camarones con arroz blanco. Los fines de semana, tr?os de m?sicos cubanos \" :Cuisine_type Cubana :Performance true :Parking false :Category 0 :Reserve false :Average_Price 20.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443543206182 :in-reply-to R1443543206166_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1438932900_1443543206166)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443543206307_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C156356573_1443543206307)

(**AGREE**

:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443543206307 :in-reply-to R1443543206307_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C156356573_1443543206307)

(**INFORM**

:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915229318\" :Zone Centro :Price 30.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name Ternera :Street_Type Calle) :Traffic_Cut false) :Number 4) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name \"Cuando sal? de Cuba\" :Description \"'Cuando sal? de Cuba' derrocha, desde 1998, aut?ntico sabor isle?o con platos como la enchilada de camarones con arroz blanco. Los fines de semana, tr?os de m?sicos cubanos \" :Cuisine_type Cubana :Performance true :Parking false :Specialities (sequence \"\") :Category 0 :Reserve false :Average_Price 20.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443543206322 :in-reply-to R1443543206307_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C156356573_1443543206307)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915229318\" :Zone Centro :Price 30.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name Ternera :Street_Type Calle) :Traffic_Cut false) :Number 4) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name \"Cuando sal? de Cuba\" :Description \"'Cuando sal? de Cuba' derrocha, desde 1998, aut?ntico sabor isle?o con platos como la enchilada de camarones con arroz blanco. Los fines de semana, tr?os de m?sicos cubanos \" :Cuisine_type Cubana :Performance true :Parking false :Specialities (sequence \"\") :Category 0 :Reserve false :Average_Price 20.0))))"

:language fipa-sl :ontology ontology)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443543206776_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1512422130_1443543206776)

(**AGREE**

:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443543206791 :in-reply-to R1443543206776_0

:language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C1512422130_1443543206776)
(**INFORM**
:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((result (action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"Cuando sal? de Cuba\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915229318\" :Zone Centro :Price 30.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name Ternera :Street_Type Calle) :Traffic_Cut false) :Number 4) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name \"Cuando sal? de Cuba\" :Description \"'Cuando sal? de Cuba' derrocha, desde 1998, aut?ntico sabor isle?o con platos como la enchilada de camarones con arroz blanco. Los fines de semana, tr?os de m?sicos cubanos \" :Cuisine_type Cubana :Performance true :Parking false :Specialities (sequence \"\") :Category 0 :Reserve false :Average_Price 20.0))))"
:reply-with busca@192.168.1.231:1099/JADE1443543206791 :in-reply-to R1443543206776_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C1512422130_1443543206776)
(**INFORM**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915229318\" :Zone Centro :Price 30.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name Ternera :Street_Type Calle) :Traffic_Cut false) :Number 4) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name \"Cuando sal? de Cuba\" :Description \"'Cuando sal? de Cuba' derrocha, desde 1998, aut?ntico sabor isle?o con platos como la enchilada de camarones con arroz blanco. Los fines de semana, tr?os de m?sicos cubanos \" :Cuisine_type Cubana :Performance true :Parking false :Specialities (sequence \"\") :Category 0 :Reserve false :Average_Price 20.0))))"
:language fipa-sl :ontology ontology)

Figura 6.9 Mensajes Consulta 2

6.2.3. *Consulta 3*

Se solicita información sobre el restaurante “Public” a través del GUI proporcionado por el agente RestAgent. La configuración del agente BuscaAgent cuando se realizaba esta consulta era P-ONTO-3.

La figura 6.10 muestra la petición y el resultado obtenido.

La captura de la traza obtenida por el agente Sniffer se muestra en la figura 6.11, donde se puede ver el intercambio de mensajes producidos entre los agentes del SMA para conseguir la información solicitada. Se observa que las peticiones realizadas a los agentes OntoAgent y Web agent se han realizado en paralelo, ya que en este caso el agente BuscaAgent estaba configurado para gestionar las tareas de ese modo.

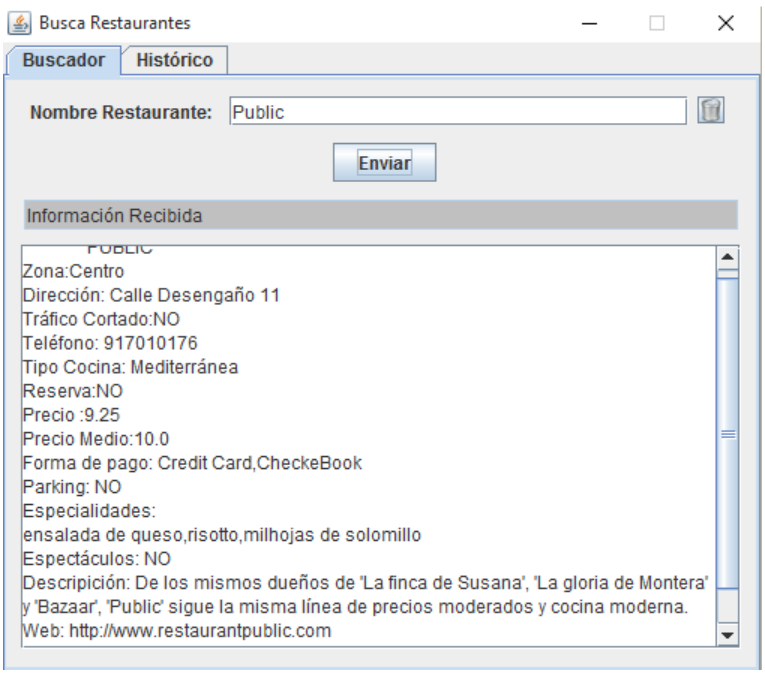


Figura 6.10 Resultado Consulta 3

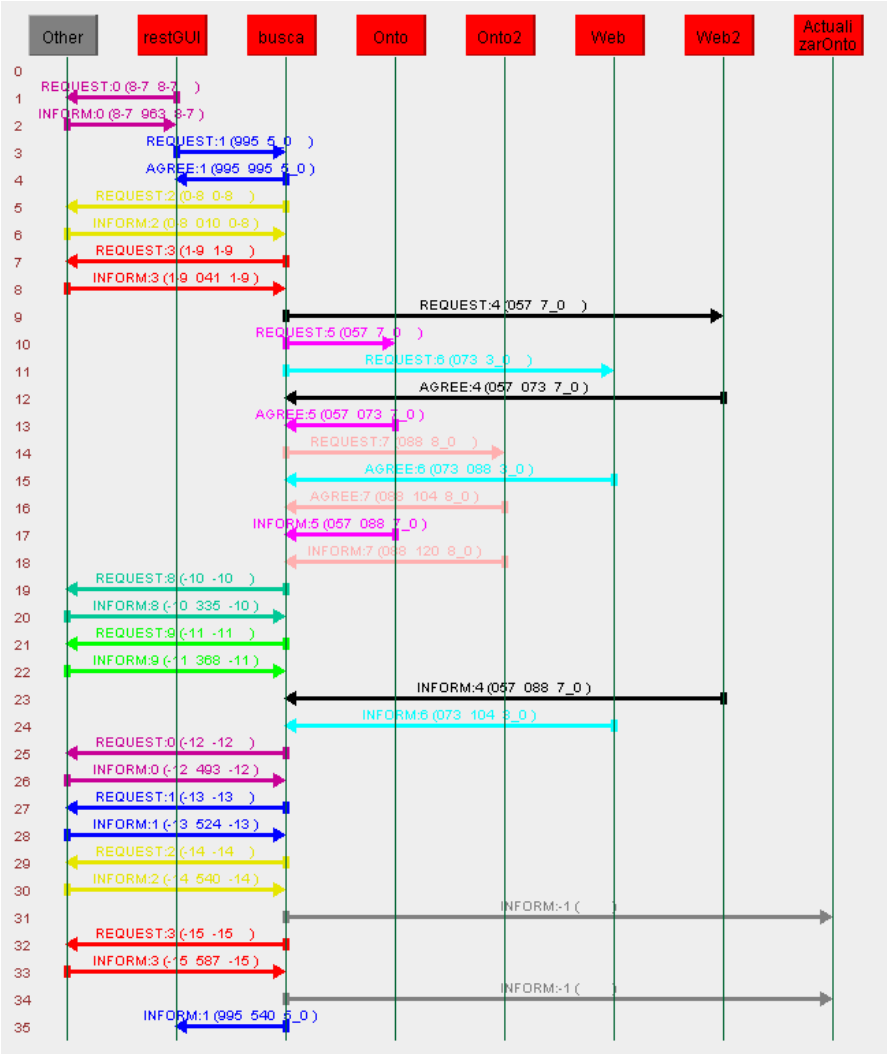


Figura 6.11 Captura Sniffer Consulta 3

A continuación en la figura 6.12 se muestra el detalle de los mismos.

(**REQUEST**
:sender (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443546348995_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C1345298406_1443546348995)

(**AGREE**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:reply-with restGUI@192.168.1.231:1099/JADE1443546348995 :in-reply-to R1443546348995_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C1345298406_1443546348995)

(**REQUEST**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443546349057_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C137489369_1443546349057)

(**REQUEST**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443546349057_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C291000572_1443546349057)

(**REQUEST**
:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:content "((action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"
:reply-with R1443546349073_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C222244100_1443546349073)

(**AGREE**
:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))
:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))
:reply-with busca@192.168.1.231:1099/JADE1443546349073 :in-reply-to R1443546349057_0 :language fipa-sl :ontology ontology :protocol fipa-request
:conversation-id C137489369_1443546349057)

(**AGREE**
:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443546349073 :in-reply-to R1443546349057_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C291000572_1443546349057)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443546349088_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1026349385_1443546349088)

(**AGREE**

:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443546349088 :in-reply-to R1443546349073_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C222244100_1443546349073)

(**AGREE**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443546349104 :in-reply-to R1443546349088_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1026349385_1443546349088)

(**INFORM**

:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"917010176\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restaurantpublic.com :Fall_Probability 0.0)) :Price 9.25 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Desenga?o\" :Street_Type Calle) :Traffic_Cut false) :Number 11) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name Public :Description \"De los mismos due?os de 'La finca de Susana', 'La gloria de Montera' y 'Bazaar', 'Public' sigue la misma l?nea de precios moderados y cocina moderna.\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Specialities (sequence \"ensalada de queso\" risotto \"milhojas de solomillo\") :Category 0 :Reserve false :Average_Price 10.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443546349088 :in-reply-to R1443546349057_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C291000572_1443546349057)

(**INFORM**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"917010176\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restaurantpublic.com :Fall_Probability 0.0)) :Price 9.25 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Desenga?o\" :Street_Type Calle) :Traffic_Cut false) :Number 11) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name Public :Description \"De los mismos due?os de 'La finca de Susana', 'La gloria de Montera' y 'Bazaar', 'Public' sigue la misma l?nea de precios moderados y cocina moderna.\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false

:Specialities (sequence \"ensalada de queso\" risotto \"milhojas de solomillo\") :Category 0 :Reserve false :Average_Price 10.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443546349120 :in-reply-to R1443546349088_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1026349385_1443546349088)

(**INFORM**

:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"917010176\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restaurantpublic.com :Fall_Probability 0.0)) :Price 9.25 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Desenga?o\" :Street_Type Calle) :Traffic_Cut false) :Number 11) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name Public :Description \"De los mismos due?os de 'La finca de Susana', 'La gloria de Montera' y 'Bazaar', 'Public' sigue la misma l?nea de precios moderados y cocina moderna.\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Specialities (sequence \"ensalada de queso\" risotto \"milhojas de solomillo\" \"\") :Category 0 :Reserve false :Average_Price 10.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443546349088 :in-reply-to R1443546349057_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C137489369_1443546349057)

(**INFORM**

:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"917010176\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restaurantpublic.com :Fall_Probability 0.0)) :Price 9.25 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Desenga?o\" :Street_Type Calle) :Traffic_Cut false) :Number 11) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name Public :Description \"De los mismos due?os de 'La finca de Susana', 'La gloria de Montera' y 'Bazaar', 'Public' sigue la misma l?nea de precios moderados y cocina moderna.\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Specialities (sequence \"ensalada de queso\" risotto \"milhojas de solomillo\" \"\") :Category 0 :Reserve false :Average_Price 10.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443546349104 :in-reply-to R1443546349073_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C222244100_1443546349073)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"917010176\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restaurantpublic.com :Fall_Probability 0.0)) :Price 9.25 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Desenga?o\" :Street_Type Calle) :Traffic_Cut false) :Number 11) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name Public :Description \"De los mismos due?os de 'La finca de Susana', 'La gloria de Montera' y 'Bazaar', 'Public' sigue la misma l?nea de precios moderados y cocina moderna.\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Specialities (sequence \"ensalada de queso\" risotto \"milhojas de solomillo\" \"\") :Category 0 :Reserve false :Average_Price 10.0))))"

:language fipa-sl :ontology ontology)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses

(sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"917010176\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restaurantpublic.com :Fall_Probability 0.0)) :Price 9.25 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Desenga?o\" :Street_Type Calle) :Traffic_Cut false) :Number 11) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name Public :Description \"De los mismos due?os de 'La finca de Susana', 'La gloria de Montera' y 'Bazaar', 'Public' sigue la misma l?nea de precios moderados y cocina moderna.\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Specialities (sequence \"ensalada de queso\" risotto \"milhojas de solomillo\" \"\") :Category 0 :Reserve false :Average_Price 10.0))))"

:language fipa-sl :ontology ontology)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **restGUI**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name Public :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence ENCONTRADO (RESTAURANT :Telephone \"917010176\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.restaurantpublic.com :Fall_Probability 0.0)) :Price 9.25 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Desenga?o\" :Street_Type Calle) :Traffic_Cut false) :Number 11) :Payment_Form (sequence \"Credit Card\" CheckeBook) :Name Public :Description \"De los mismos due?os de 'La finca de Susana', 'La gloria de Montera' y 'Bazaar', 'Public' sigue la misma l?nea de precios moderados y cocina moderna.\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Specialities (sequence \"ensalada de queso\" risotto \"milhojas de solomillo\") :Category 0 :Reserve false :Average_Price 10.0))))"

:reply-with restGUI@192.168.1.231:1099/JADE1443546349540 :in-reply-to R1443546348995_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1345298406_1443546348995)

Figura 6.12 Mensajes Consulta 3

Analizando la traza y detalle de los mensajes, se observa que aunque recibe mensajes con información de los agentes OntoAgent, no envía inmediatamente la información, sino que queda a la espera de recibir respuesta por parte de los agentes WebAgent. Esto es debido a que BuscaAgent tiene configurada la estrategia 3. Si recibe información tanto de los agentes OntoAgent como de los agentes WebAgent, realiza una suma de la información proporcionada por ambos y la envía como respuesta final.

6.2.4. *Consulta 4*

Se solicita información sobre el restaurante “La gloria de Montera”. La petición se realiza por línea de comando directamente en la creación del agente RestAgent, como se indica en la figura 6.13.

```
java jade.Boot -container restCMD:proyecto.restaurantes.RestAgent(C,resultado.xml,"La Gloria de Montera")
```

Figura 6.13 Petición Consulta 4

La configuración del agente BuscaAgent cuando se realizaba esta consulta era P-WEB-1. Dado que la petición se realiza directamente por comando y no interviene ningún GUI, el resultado de la consulta se vuelca a un fichero XML, como se muestra en la figura 6.14.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<RESTAURANTES>
    <RESTAURANT>
        <NAME>La Gloria de Montera</NAME>
        <ZONE>Centro</ZONE>
        <ADDRESS>
            <NUMBER>10</NUMBER>
            <STREET_SECTION>
                <STREET>
                    <STREET_NAME>Caballero de Gracia</STREET_NAME>
                    <STREET_TYPE>Calle</STREET_TYPE>
                </STREET>
                <TRAFFIC/>
                <TRAFFIC_CUT>NO</TRAFFIC_CUT>
            </STREET_SECTION>
        </ADDRESS>
        <TELEPHONE>915234407</TELEPHONE>
        <CUISINE_TYPE>Mediterránea</CUISINE_TYPE>
        <CATEGORY>0</CATEGORY>
        <RESERVE>NO</RESERVE>
        <PRICE>10.0</PRICE>
        <AVERAGE_PRICE>30.0</AVERAGE_PRICE>
        <PAYMENT_FORM>
            <PAYMENT>Credit Card</PAYMENT>
        </PAYMENT_FORM>
        <PARKING>NO</PARKING>
        <SPECIALITIES/>
        <PERFORMANCE>NO</PERFORMANCE>
        <DESCRIPTION>Comedor amplio y diáfano de techos altos, con mesas
apretadas, platos mediterráneos ligeros</DESCRIPTION>
        <WEB_PAGE>
            <WEB_INFORMATION>
                <AVERAGE_ACCESS_TIME/>
                <LINK>http://www.lagloriademontera.com</LINK>
                <SCORE/>
                <FALL_PROBABILITY>0.0</FALL_PROBABILITY>
            </WEB_INFORMATION>
        </WEB_PAGE>
        <ACCESSIBILITIES/>
    </RESTAURANT>
</RESTAURANTES>
```

Figura 6.14 Resultado Consulta 4

La captura de la traza obtenida por el agente Sniffer se muestra en la figura 6.15 donde se puede ver el intercambio de mensajes producidos entre los agentes del SMA para conseguir la información solicitada. Se observa que las peticiones de información por parte de BuscaAgent se han ejecutado en paralelo, de acuerdo a la configuración que tenía en ese momento. Al tener

configurada la estrategia 1, según ha recibido el primer mensaje con la información solicitada, ha procedido a contestar la petición de RestAgent.

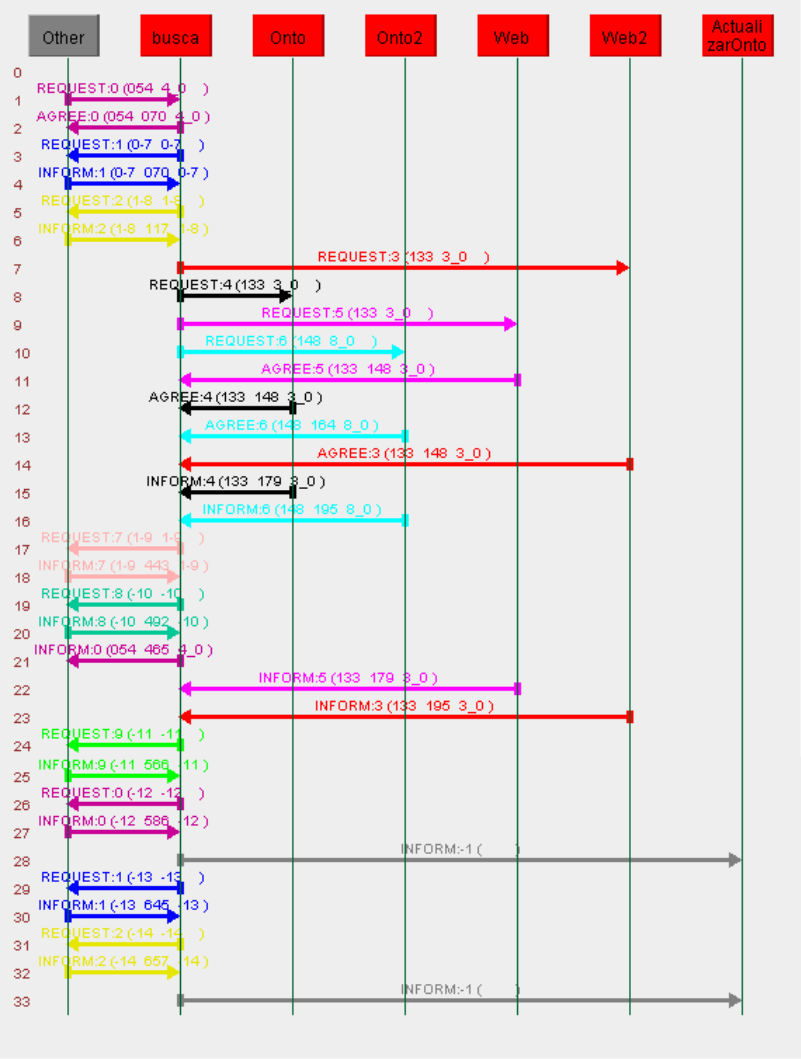


Figura 6.15 Captura Sniffer Consulta 4

A continuación en la figura 6.16 se muestra el detalle de los mensajes.

(**REQUEST**

:sender (agent-identifier :name **Rest1**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443550377054_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C50738407_1443550377054)

(**AGREE**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Rest1**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with Rest1@192.168.1.231:1099/JADE1443550377070 :in-reply-to R1443550377054_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C50738407_1443550377054)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C940931322_1443550377133)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C12380264_1443550377133)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1879359552_1443550377133)

(**REQUEST**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)))"

:reply-with R1443550377148_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1916666936_1443550377148)

(**AGREE**

:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443550377148 :in-reply-to R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1879359552_1443550377133)

(**AGREE**

:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443550377148 :in-reply-to R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C12380264_1443550377133)

(**AGREE**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence

http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443550377164 :in-reply-to R1443550377148_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1916666936_1443550377148)

(**AGREE**

:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:reply-with busca@192.168.1.231:1099/JADE1443550377148 :in-reply-to R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C940931322_1443550377133)

(**INFORM**

:sender (agent-identifier :name **Onto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Onto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915234407\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.lagloriademontera.com :Fall_Probability 0.0)) :Price 10.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Caballero de Gracia\" :Street_Type Calle) :Traffic_Cut false) :Number 10) :Payment_Form (sequence \"Credit Card\") :Name \"La Gloria de Montera\" :Description \"Comedor amplio y di?fano de techos altos, con mesas apretadas, platos mediterr?neos ligeros\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443550377179 :in-reply-to R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C12380264_1443550377133)

(**INFORM**

:sender (agent-identifier :name **Onto2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Onto2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915234407\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.lagloriademontera.com :Fall_Probability 0.0)) :Price 10.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Caballero de Gracia\" :Street_Type Calle) :Traffic_Cut false) :Number 10) :Payment_Form (sequence \"Credit Card\") :Name \"La Gloria de Montera\" :Description \"Comedor amplio y di?fano de techos altos, con mesas apretadas, platos mediterr?neos ligeros\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443550377195 :in-reply-to R1443550377148_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1916666936_1443550377148)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **Rest1**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name busca@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence ENCONTRADO (RESTAURANT :Telephone \"915234407\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.lagloriademontera.com :Fall_Probability 0.0)) :Price 10.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Caballero de Gracia\" :Street_Type Calle) :Traffic_Cut false) :Number 10) :Payment_Form (sequence \"Credit Card\") :Name \"La Gloria de Montera\" :Description \"Comedor amplio y di?fano de techos altos, con mesas apretadas, platos mediterr?neos ligeros\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with Rest1@192.168.1.231:1099/JADE1443550377465 :in-reply-to R1443550377054_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C50738407_1443550377054)

(**INFORM**

:sender (agent-identifier :name **Web**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Web@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915234407\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.lagloriademontera.com :Fall_Probability 0.0)) :Price 10.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Caballero de Gracia\" :Street_Type Calle) :Traffic_Cut false) :Number 10) :Payment_Form (sequence \"Credit Card\") :Name \"La Gloria de Montera\" :Description \"Comedor amplio y di?fano de techos altos, con mesas apretadas, platos mediterr?neos ligeros\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443550377179 :in-reply-to R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C1879359552_1443550377133)

(**INFORM**

:sender (agent-identifier :name **Web2**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name Web2@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Name \"La Gloria de Montera\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915234407\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.lagloriademontera.com :Fall_Probability 0.0)) :Price 10.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Caballero de Gracia\" :Street_Type Calle) :Traffic_Cut false) :Number 10) :Payment_Form (sequence \"Credit Card\") :Name \"La Gloria de Montera\" :Description \"Comedor amplio y di?fano de techos altos, con mesas apretadas, platos mediterr?neos ligeros\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 30.0))))"

:reply-with busca@192.168.1.231:1099/JADE1443550377195 :in-reply-to R1443550377133_0 :language fipa-sl :ontology ontology :protocol fipa-request

:conversation-id C940931322_1443550377133)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915234407\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.lagloriademontera.com :Fall_Probability 0.0)) :Price 10.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Caballero de Gracia\" :Street_Type Calle) :Traffic_Cut false) :Number 10) :Payment_Form (sequence \"Credit Card\") :Name \"La Gloria de Montera\" :Description \"Comedor amplio y di?fano de techos altos, con mesas apretadas, platos mediterr?neos ligeros\" :Cuisine_type \"Mediterr?nea\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 30.0))))"

:language fipa-sl :ontology ontology)

(**INFORM**

:sender (agent-identifier :name **busca**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc))

:receiver (set (agent-identifier :name **ActualizarOnto**@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)))

:content "((result (action (agent-identifier :name ActualizarOnto@192.168.1.231:1099/JADE :addresses (sequence http://192.168.1.231:7778/acc)) (RESTAURANT :Price 0.0 :Performance false :Parking false :Category 0 :Reserve false :Average_Price 0.0)) (sequence (RESTAURANT :Telephone \"915234407\" :Zone Centro :Web_Page (sequence (WEB-INFORMATION :Link http://www.lagloriademontera.com :Fall_Probability 0.0)) :Price 10.0 :Address (ADDRESS :Localization (STREET-SECTION :Length_Section 0.0 :Street (STREET :Street_Name \"Caballero de Gracia\" :Street_Type Calle) :Traffic_Cut false) :Number 10) :Payment_Form (sequence \"Credit Card\") :Name \"La Gloria de Montera\" :Description \"Comedor amplio

```
y di?fano de techos altos, con mesas apretadas, platos mediterr?neos ligeros\" :Cuisine_type
\"Mediterr?nea\" :Performance false :Parking false :Category 0 :Reserve false :Average_Price 30.0))))"
 :language  fipa-sl  :ontology  ontology )
```

Figura 6.16 Mensajes Consulta 4

Capítulo 7. Conclusiones

Cuando comencé a desarrollar este proyecto hace ya algunos años, la documentación existente sobre JADE era bastante escasa. Prácticamente la única documentación de referencia era la proporcionada por ellos. Hoy en día se ha producido un cambio y se puede encontrar fácilmente documentación y ejemplos de JADE, lo que me hace pensar que JADE se ha instaurado cómo la herramienta más habitual para el desarrollo de agentes.

Comenzar a trabajar con JADE me resultó bastante costoso. No tanto por JADE en sí, sino más bien por la poca familiaridad que poseía para programar en JAVA y lo que es más importante, entender lo que quería programar: un Agente. Después de muchas horas y a medida que iba entendiendo el concepto de agente y de SMA fui siendo consciente de como JADE, simplifica y facilita el trabajo a la hora de programar agentes.

Al haber sido tan largo el periodo comprendido entre el comienzo y la finalización de este proyecto, ha sido necesario ir actualizando las versiones de JADE y por supuesto de JAVA. El hecho de que JADE esté implementado en JAVA ha sido una gran ventaja para mí, ya que la labor de realizar dichas actualizaciones resulta relativamente sencilla y prácticamente sin repercusión. Otra ventaja de que JADE esté implementado en JAVA, es que permite que la herramienta desarrollada sea independiente de la plataforma de ejecución.

En el proyecto desarrollado sólo se hace uso de un *wrapper*, pero la integración de más agentes especializados en buscar en otros orígenes sería muy sencilla, ya que sólo implicaría cambiar un par de líneas de código.

Otra posible adaptación futura de este SMA sería la integración con un *Web Service,* de forma que las peticiones que llegarán al SMA vendrían por este medio.

Bibliografía y Referencias

[Bellifemine, 2007] F. Bellifemine, G. Caire, D. Greenwood (2007). Developing Multi-Agent Systems with JADE. John Wiley & Sons.

[Bellifemine, 2010] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco (TILAB S.p.A., formerly CSELT), Giovanni Rimassa (FRAMeTech s.r.l.), Roland Mungenast (PROFACTOR GmbH) (2010). JADE Administrator's GUIDE.

[Caire, 2009] Giovanni Caire (TILAB, formerly CSELT) (2009). JADE TUTORIAL JADE PROGRAMMING FOR BEGINNERS.

[Caire, 2010] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco (TILAB, formerly CSELT), Giovanni Rimassa (University of Parma)(2010). JADE Programmer's GUIDE.

[Carrascosa, 2001] C. Carrascosa, V. J. Julián, M. Rebollo (2001). Una taxonomía para los Agentes de Información. En: Actas del V Congreso ISKO-España. Alcalá de Henares, 2001.

[FIPA] The Foundation for Intelligent Physical Agents, http://www.fipa.org/

[FIPA00037] FIPA Communicative Act Library Specification, http://www.fipa.org/

[FIPA00061] FIPA ACL Message Structure Specification, http://www.fipa.org/

[FIPA00023] FIPA Agent Management Specification, http://www.fipa.org/

[Franklin, 1996] S. Franklin, A. Graesser (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents.

[KQML, 1993] Specification of the KQML agent communication language. DARPA knowledge sharing initiative external interfaces working group

[Maes, 1995] P. Maes (1995). Artificial Life Meets Entertainment: Life like Autonomous Agents. Communications of the

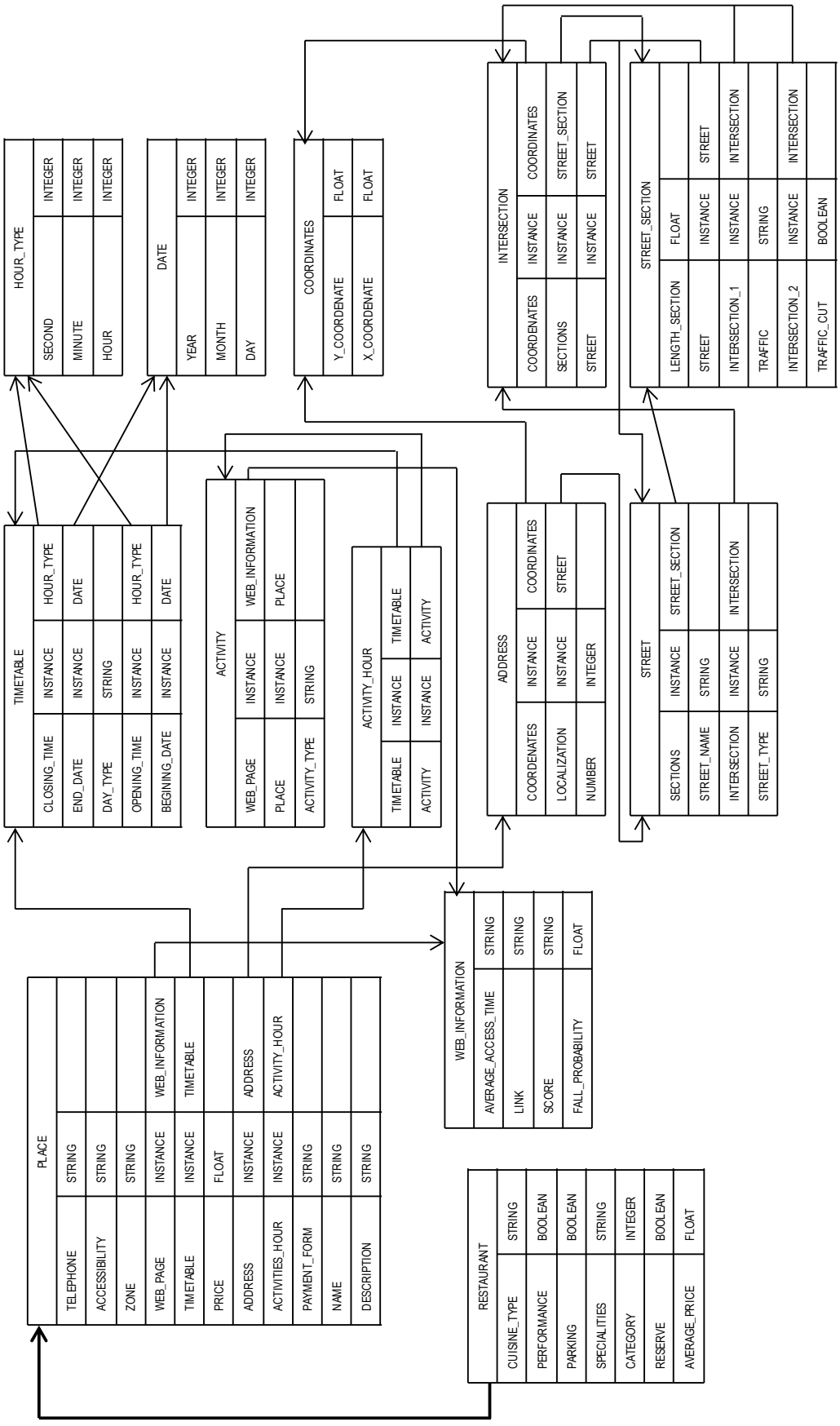
ACM.

[Mas, 2005] A. Mas (2005). Agentes software y sistemas multiagente. Conceptos, arquitecturas y aplicaciones. PEARSON EDUCACIÓN.

[Russell & Norvig, 1995] S. J. Russell, P. Norvig (1995) Artificial Intelligence: A Modern Approach. Prentice Hall.

[Wooldridge, 2002] Wooldridge, M (2002). An Introduction to Multiagent Systems. John Wiley & Sons.

Anexo 1: Definición de la Ontología



Anexo 2: Formato fichero Ontología

```
      ([ontology_00060] of  %3APAL-CONSTRAINT
      )
      ([ontology_Instance_8514] of  RESTAURANT
      (Address [ontology_Instance_8515])
      (Web_Page [ontology_Instance_8516])
      (Reserve "true")
      (Average_Price 55.0)
      (Price 0.0)
      (Telephone "915702004")
      (Performance "false")
      (Parking "true")
      (Payment_Form
      "Credit Card")
      (Specialities
      "pescados a la sal"
      "fritos")
      (Description "Decoración marinera para un restaurante especializado en platos
de pescado y marisco")
      (Name "La Dorada")
      (Category 0)
      (Cuisine_type "Andaluza / Marinera")
      (Zone "Tetuán"))
      ([ontology_Instance_8515] of  ADDRESS
      (Localization [ontology_Instance_8517])
      (Number 64))
      ([ontology_Instance_8517] of STREET-SECTION
      (Street [ontology_Instance_8518]))
      ([ontology_Instance_8518] of STREET
      (Street_Type "Calle")
      (Street_Name "Orense"))
      ([ontology_Instance_8516] of WEB-INFORMATION
      (Link "http://www.ladorada.es"))
```