

---

## **PROYECTO FINAL DE CARRERA**



.....

# **Agent0: Sistema Multi-Agente que controla las situaciones de urgencia.**

**Cristóbal Ledesma Olías,**  
**Ingeniería Técnica en Informática de Sistemas.**

**Directores:**  
**Aïda Valls, URV, [avalls@etse.urv.es](mailto:avalls@etse.urv.es)**  
**Antonio Moreno, URV, [amoreno@etse.urv.es](mailto:amoreno@etse.urv.es)**

**Escola Tècnica Superior d'Enginyeria (ETSE)**  
**Universitat Rovira i Virgili (URV), <http://www.etse.urv.es>**

**Febrero, 2003**

---



## **1. Introducción (página 7)**

Figura 1.1 Dependencias jerárquicas entre coordinadores. (página 7)

Figura 1.2 Distribución del territorio nacional por zonas según la ONT.(página 8)

**1.1 ¿Qué aporta la informatización de la gestión de trasplantes de órganos? (página 8).**

**1.2 Ventajas que aporta la utilización de un sistema multi-agente.(página 9)**

**1.3 ¿Qué podemos encontrar en esta documentación? (página 9)**

## **2. Agentes y Sistemas Multi-Agente (página 11)**

**2.1. Definición de agente (página 11)**

**2.2. Tipos de arquitecturas (página 12)**

**2.3 Tipos de agentes (página 12)**

**2.4. Métodos para comunicar dos agentes. (Agentificación). (página 12)**

**2.5. Métodos de comunicación. (página 13).**

**2.6. Entornos para el desarrollo de Sistemas Multiagente. (página 13)**

**2.7. ¿Qué recursos se han utilizado para implementar este proyecto? (página 14)**

## **3. Especificaciones realizadas por la FIPA. (página 15)**

Figura 3.1 Entorno básico para ejecutar agentes. (página 15)

**3.1 Descripción de los protocolos implementados en JADE. (página 16)**

**3.1.1 Protocolo FIPA-Query. (página 17)**

Figura 3.2 Evolución del protocolo FIPA Query. (página 17)

**3.1.2 Protocolo FIPA-request. (página 18)**

Figura 3.3 Evolución del protocolo FIPA Request (página 18)

**3.1.3 Protocolo FIPA-Contract Net. (página 19)**

Figura 3.4 Evolución del protocolo FIPA Contract Net. (página 19)

**3.2 Formato de los mensajes ACL (Agent Communication Language). (página 20)**

Figura 3.5 Mensaje ACL (página 20)

Figura 3.6 Tabla que relaciona acto comunicativo con función que ejerce dentro de un protocolo. (página 20)

## **4. Entorno de ejecución de agentes JADE**

### **(Java Agent Development Enviroment). (página 22)**

Figura 4.1 Ventana inicial de entorno gráfico de JADE. (página 23)

**4.1 Herramientas proporcionadas por JADE. (página 23)**

**4.1.1 El Menu Actions. (página 24)**

Figura 4.2 Ventana inicial de JADE con menú Actions seleccionado. (página 24)

Figura 4.3 Ventana que permite iniciar un agente en JADE. (página 24)

Figura 4.4 Ventana que permite enviar un mensaje a un agente. (página 24)

**4.1.2 El menú Tools. (página 25)**

Figura 4.5 Ventana inicial de JADE con menú tools seleccionado. (página 25)

Figura 4.6 Ventana con mensajes que se envían los agentes. (página 25)

**4.2 Construcción de agentes en JADE. (página 26)**

## **5. Descripción del sistema. (página 27)**

Figura 5.1 Visión general de sistema multiagente desarrollado en este proyecto. (página 27)

Figura 5.2 Estados desde que paciente solicita un órgano hasta que se le trasplanta. (página 28)

**5.1 Descripción componentes de AgentCTH (página 29)**

**5.1.1 Descripción de la interficie gráfica. (página 29)**

**5.1.1.1 Menú Principal. (página 29)**

Figura 5.3 Menú principal de la interfaz gráfica. (página 29)

- 5.1.1.2 Menú Opciones. (página 29)  
Figura 5.4 Menú opciones de la interfaz gráfica. (página 29)
- 5.1.1.3 Menú Introducir Paciente. (página 30)  
Figura 5.5 Ventana principal para registrar paciente-órgano en el sistema. (página 30)
- 5.1.1.4 Menú Donante. (página 30)  
Figura 5.6 Ventana para introducir órgano donado. (página 31)
- 5.1.1.5 Menú Muerte Paciente. (página 31)  
Figura 5.7 Ventana para indicar momento muerte de donante. (página 31)
- 5.1.1.6 Menú Receptor. (página 32)  
Figura 5.8 Ventana para introducir datos de órgano que se espera recibir. (página 32)
- 5.1.1.7 Menú Anotar Muerte Paciente. (página 32)  
Figura 5.9 Ventana para anotar la muerte de un paciente. (página 32)
- 5.1.1.8 Menú Encontrar Pacientes Compatibles. (página 33)  
Figura 5.10 Ventana para buscar pacientes compatibles con un órgano. (página 33)
- 5.1.1.9 Menú Anotar Asignación Órgano. (página 33)  
Figura 5.11 Ventana para asignar un órgano a un paciente. (página 33)
- 5.1.1.10 Menú Anotar Órgano Rechazado. (página 34)  
Figura 5.12 Ventana para indicar que un órgano ha sido rechazado por un paciente. (página 34)
- 5.1.1.11 Menú Anotar Operación Realizada. (página 34)  
Figura 5.13 Ventana para indicar momento de un trasplante. (página 34)
- 5.1.2 Ontología propia del agente AgentCTH. (página 35)
  - 5.1.2.1 Ontología asociada a la acción introducir pacientes. (página 35)
  - 5.1.2.2 Ontología asociada a la acción anotar muerte paciente. (página 36)
  - 5.1.2.3 Ontología asociada a la acción anotar la asignación de un órgano a un paciente. (página 37)
  - 5.1.2.4 Ontología asociada a la acción anotar el rechazo de un órgano por un paciente. (página 38)
  - 5.1.2.5 Ontología asociada a la acción anotar trasplante realizado a un paciente. (página 39)
  - 5.1.2.6 Ontología asociada a la acción encontrar pacientes compatibles con un determinado órgano. (página 39)
- 5.2 Descripción componentes de Ag0. (página 42)
  - 5.2.1 Ontología propia del agente Ag0. (página 42)
    - 5.2.1.1 Ontología asociada a la acción encontrar pacientes compatibles con un determinado órgano. (página 42)
  - 5.2.2 Base de datos que gestiona el agente Ag0. (página 44)
    - 5.2.2.1 Modelo Entidad-Interrelación de la base de datos. (página 44)  
Figura 5.14 Modelo Entidad-Interrelación BD de Ag0. (página 44)
    - 5.2.2.2 Tablas utilizadas para implementar la base de datos. (página 45)
    - 5.2.2.3 ¿Cómo gestiona Ag0 la base de datos?. (página 46)
      - 5.2.2.3.1 Gráfico que describe el comportamiento de Ago sobre la base de datos. (página 46)  
Figura 5.15 Proceso que siguen los órganos en el sistema. (página 46)
      - 5.2.2.3.2 Respuesta Introducción Paciente. (página 46)
      - 5.2.2.3.3 Respuesta Anotación Muerte Paciente. (página 47)

- 5.2.2.3.4 Respuesta Asignación Órgano. (página 47)
- 5.2.2.3.5 Respuesta Operación Realizada. (página 47)
- 5.2.2.3.6 Respuesta Órgano Rechazado. (página 48)
- 5.2.2.3.7 Respuesta Encontrar Pacientes Compatibles. (página 48)

### **5.3 Descripción del SMA que se encarga de la ordenación de pacientes. (página 49)**

Figura 5.16 Estructura de sistema multi-agente encargado en la ordenación de pacientes. (página 49)

## **6. Juego de pruebas. (página 51)**

- 6.1 Errores controlados por la propia interfaz gráfica. (página 51)
- 6.2 Respuesta del sistema a acciones solicitadas por usuario. (página 51)
  - 6.2.1 Respuesta del sistema a acciones que involucran solo al AgentCTH y Ag0. (página 52)
  - 6.2.2 Respuesta del sistema en búsqueda de pacientes compatibles. (página 57)

## **7. Contenido de la base de datos. (página 61)**

- 7.1 Caso 1: Contenido de la base de datos. (página 61)
- 7.2 Caso 2: Contenido de la base de datos. (página 66)

## **8. Código fuente del sistema multi-agente implementado. (página 79)**

- 8.1 Distribución modular del sistema. (página 80)
  - 8.1.1 AgentCTH. (página 80)
    - 8.1.1.1 Archivos contenidos en ./Agent0/agentes/CTH. (página 80)
    - 8.1.1.2 Archivos contenidos en ./Agent0/agentes/CTH/interficie. (página 81)
    - 8.1.1.3 Archivos contenidos en ./Agent0/agentes/CTH/Ontology. (página 81)
  - 8.1.2 Ag0. (página 82)
    - 8.1.2.1 Archivos contenidos en ./Agent0/agentes/A0. (página 82)
    - 8.1.2.2 Archivos contenidos en ./Agent0/agentes/A0/Ontology. (página 83)
    - 8.1.2.3 Archivos contenidos en ./Agent0/agentes/A0/BDAgent0. (página 83)
  - 8.1.3 Especialista. (página 83)
    - 8.1.3.1 Archivos contenidos en ./Agent0/agentes/especial. (página 83)
    - 8.1.3.2 Archivos contenidos en ./Agent0/agentes/especial/Ontology. (página 83)
- 8.2 Impresión del código fuente del sistema. (página 84)
  - 8.2.1 Impresión de código contenido en ./Agent0/agentes/CTH. (página 84)
    - 8.2.1.1 AgentCTH.java. (página 84)
    - 8.2.1.2 AnotacioAsignacionOrganoRequest.java. (página 92)

- 8.2.1.3 AnotacionMuertePacienteRequest.java (página 96)
- 8.2.1.4 AnotacionOperacionRealizadaRequest.java (página 99)
- 8.2.1.5 AnotacionOrganoRechazadoRequest.java (página 103)
- 8.2.1.6 EncontrarPacientesCompatiblesRequest.java (página 107)
- 8.2.1.7 IntroduccionPacienteRequest.java (página 112)
- 8.2.2 Impresión de código contenido en ./Agent0/agentes/interficie.  
(página 115)
  - 8.2.2.1 MenuAnotarAsignacionOrgano.java (página 115)
  - 8.2.2.2 MenuAnotarMuertePaciente.java (página 120)
  - 8.2.2.3 MenuAnotarOrganoRechazado.java (página 125)
  - 8.2.2.4 MenuDonante.java (página 129)
  - 8.2.2.5 MenuEncontrarPacientesCompatibles.java (página 134)
  - 8.2.2.6 MenuIntroducirPaciente.java (página 139)
  - 8.2.2.7 MenuMuertePaciente.java (página 147)
  - 8.2.2.8 MenuOpciones.java (página 152)
  - 8.2.2.9 MenuOperacionRealizada.java (página 157)
  - 8.2.2.10 MenuPrincipal.java (página 162)
  - 8.2.2.11 MenuReceptor.java (página 168)
  - 8.2.2.12 Utilidades.java (página 173)
- 8.2.3 Impresión de código contenido en ./Agent0/agentes/CTH/Ontology.  
(página 180)
  - 8.2.3.1 OntologiaCTH.java (página 180)
  - 8.2.3.2 Adreza.java (página 194)
  - 8.2.3.3 AgentNoRegistreat.java (página 196)
  - 8.2.3.4 AlgoritmeNoImplementat.java (página 196)
  - 8.2.3.5 AnotarMortPacient.java (página 197)
  - 8.2.3.6 AssignacioOrgan.java (página 198)
  - 8.2.3.7 Associacio.java.(página 199)
  - 8.2.3.8 CaracteristiquesBusqueda.java (página 200)
  - 8.2.3.9 ClassesInsuficients.java (página 202)
  - 8.2.3.10 DadesCliniques.java (página 203)
  - 8.2.3.11 Domain\_OntologyObj.java (página 206)
  - 8.2.3.12 ErrorAccesBD.java (página 207)
  - 8.2.3.13Hospital.java (página 208)
  - 8.2.3.14 HospitalSenseAutoritzar.java (página 209)
  - 8.2.3.15 IntroduirPacient.java (página 210)
  - 8.2.3.16 MassaAtributs.java (página 211)
  - 8.2.3.17 MassaObjectes.java (página 211)
  - 8.2.3.18 MortPacient.java (página 212)
  - 8.2.3.19 Operacio.java (página 214)
  - 8.2.3.20 OperacioRealitzada.java (página 216)
  - 8.2.3.21 Organ.java (página 217)
  - 8.2.3.22 OrganExistent.java (página 219)
  - 8.2.3.23 OrganInexistent.java (página 219)
  - 8.2.3.24 OrganJaEstavaRebutxat.java (página 220)
  - 8.2.3.25 OrganNoDisponible.java (página 220)
  - 8.2.3.26 OrganRebutxat.java (página 221)
  - 8.2.3.27 Pacient.java (página 222)
  - 8.2.3.28 PacientInexistent.java (página 226)

- 8.2.3.29 PacientMort.java (página 226)
- 8.2.3.30 PacientOperat.java(página 226)
- 8.2.3.31 PacientSenseEsperarOrgan.java (página 227)
- 8.2.3.32 PacientSenseOperar.java (página 227)
- 8.2.3.33 PacientSenseOrganAssignat.java (página 228)
- 8.2.3.34 PacientTeOrganAssignat.java (página 228)
- 8.2.3.35 PesosNoValids.java (página 229)
- 8.2.3.36 ResultatBusqueda.java (página 230)
- 8.2.3.37 TrobarPacientsCompatibles.java (página 231)
- 8.2.4 Impresión de código contenido en ./Agent0/agentes/A0.  
(página 232)
  - 8.2.4.1 Ag0.java (página 232)
  - 8.2.4.2 ESPECIALISTABehaviour.java (página 236)
  - 8.2.4.3 RespuestaAsignacionOrgano.java (página 241)
  - 8.2.4.4 RespuestaEncontrarPacientesCompatibles.java (página 253)
  - 8.2.4.5 RespuestaIntroduccionPaciente.java (página 265)
  - 8.2.4.6 RespuestaOperacionRealizada.java (página 277)
  - 8.2.4.7 RespuestaOrganoRechazado.java (página 288)
  - 8.2.4.8 RespuestaAnotacionMuertePaciente.java (página 298)
  - 8.2.4.9 UtilidadesAgent0.java (página 305)
- 8.2.5 Impresión de código contenido en ./Agent0/agentes/A0/Ontology.  
(Página 309)
  - 8.2.5.1 OntologiaAg0.java (página 309)
  - 8.2.5.2 AccioConsultaEspecialista. (página 326)
  - 8.2.5.3 ConsultaEspecialista.java (página 327)
  - 8.2.5.4 DadesPacients.java (página 329)
  - 8.2.5.5 Grau.java (página 331)
  - 8.2.5.6 OrdenacioImpossible.java (página 332)
  - 8.2.5.7 Pacients.java (página 333)
  - 8.2.5.8 Pesos.java (página 337)
  - 8.2.5.9 Resultat.java (página 338)

## **9. ¿Cómo se pone en marcha el sistema? (página 340)**

## **10. Línea de trabajos futuros. (página 342)**

## **11. Conclusiones (página 343)**

## **12. Bibliografía (página 344)**

## 1. Introducción

El proyecto descrito en este documento, es uno más de los desarrollados por el grupo GruSMA (grupo de sistemas multi-agente creado por Aïda Valls y Antonio Moreno), que tienen la finalidad de coordinar los trasplantes de órganos a pacientes a nivel nacional. Este grupo ha desarrollado proyectos como los siguientes:

**-Logística de las operaciones de trasplante ( Jaime Bocio Sanz, 2000):** Se encarga de gestionar quirófanos, médicos, anestelistas y enfermeras, cuando le llega un órgano y fijado un tiempo límite para realizar un trasplante.

**-Logística de transporte de órganos ( Alberto Ribes García, 2000):** Se encarga de decidir cual es la ruta idónea para transportar un órgano de un hospital a otro.

**-Coordinador de trasplantes a nivel nacional ( Jaime Bocio Sanz, 2002):** Se encarga de encontrar pacientes compatibles con un determinado órgano a nivel nacional.

**-Ayuda a la asignación de órganos en trasplantes. Método ClusDM con criterios lingüísticos (David Sánchez Ruenes, 2001):** Se encarga de ordenar una lista de pacientes candidatos a recibir un órgano, según la compatibilidad de los pacientes con un determinado órgano donado.

Todos estos proyectos tienen la finalidad de gestionar los trasplantes que se realizan a nivel nacional, y lo hacen siguiendo “El modelo español” que establece la ONT (Organización Nacional de Trasplantes). Según este modelo, el territorio nacional se divide en 6 zonas y se hace la gestión teniendo en cuenta la siguiente jerarquía, vista de forma genérica:

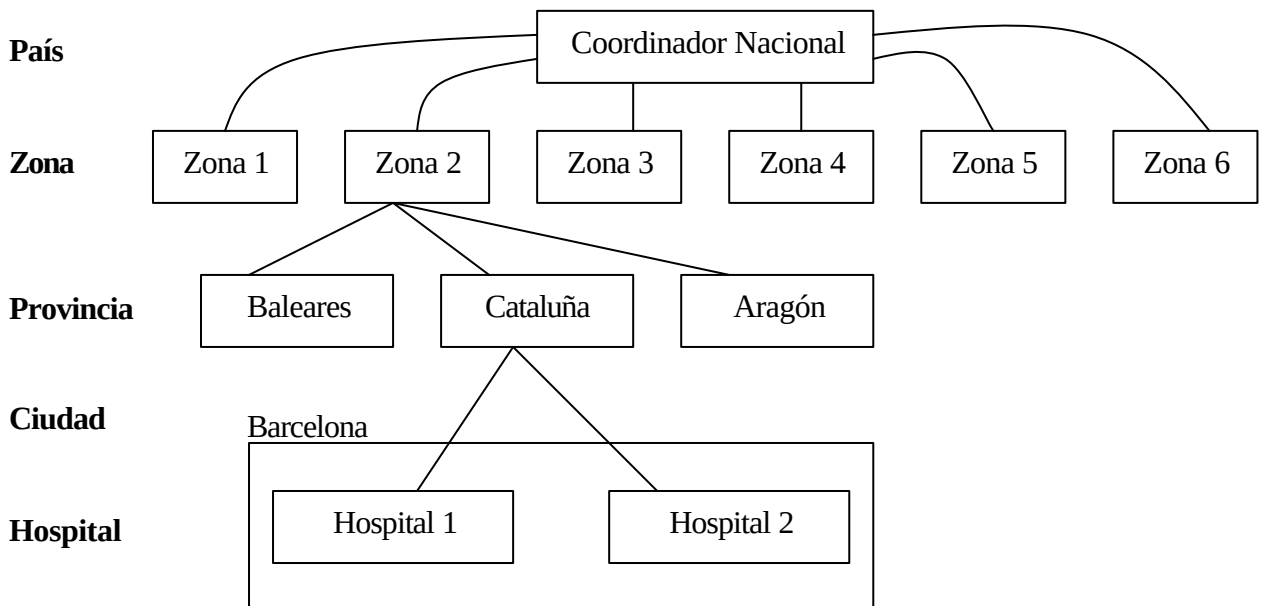


**Figura 1.1 Dependencias jerárquicas entre coordinadores.**



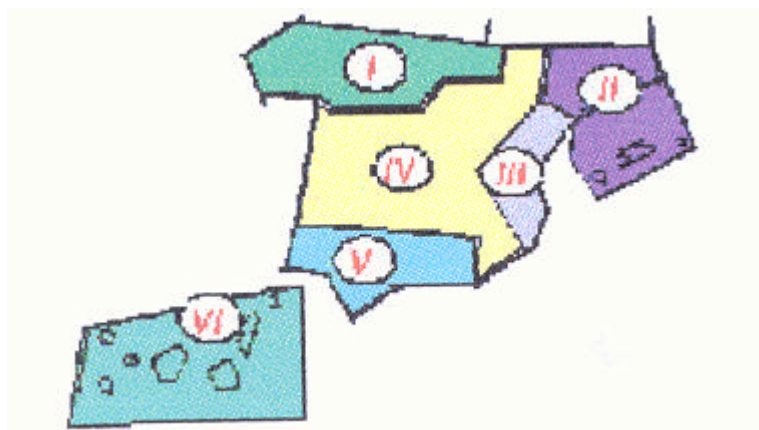


**Figura 1.2 Distribución del territorio nacional por zonas según la ONT.**

La funcionalidad de los proyectos que ha desarrollado GruSMA relativos a la coordinación de trasplantes, se puede dividir en tres grandes etapas:

-La primera etapa consiste en obtener los datos para poder realizar trasplantes. Se recogen los datos relativos a los pacientes donantes y a los pacientes que esperan recibir un órgano, los órganos donados, una descripción de los órganos que se esperan para realizar un trasplante, etc... El sistema multi-agente implementado en este proyecto, se encarga de recoger esta información para los pacientes que necesitan un órgano urgentemente, y tiene una base de datos que se va actualizando cada vez que se produce un evento significativo desde que el paciente se registra en el hospital (fallecimiento de un paciente, asignación de un órgano a un paciente, etc...).

-La segunda etapa consiste en encontrar el paciente idóneo para trasplantarle un determinado órgano. El proyecto de Jaime Bocio Sanz 2002 busca pacientes compatibles con un determinado órgano, que necesitan ser trasplantados sin urgencia. El proyecto, que se comenta en esta documentación, busca pacientes compatibles con un determinado órgano; pero en este caso, los pacientes necesitan que se les trasplante el órgano de forma inmediata, y se les da preferencia en la búsqueda. El proyecto de David Sánchez Ruenes 2001, implementa los agentes que asignan el grado de compatibilidad de los pacientes con el órgano, utilizando técnicas de inteligencia artificial basadas en la lógica difusa. **(Jaime Bocio Sanz 2002, David Sánchez Ruenes 2001, y este proyecto).**

-En la tercera etapa se busca una ruta óptima, para que el órgano viaje de un hospital a otro. **(Alberto Ribes García 2000).** En esta tercera etapa se gestiona todo el equipo médico para realizar una operación (Médicos, especialistas, enfermeras, y anestelistas). Se hace teniendo en cuenta el tiempo del que se dispone para aprovechar el órgano. **(Jaime Bocio Sanz 2000).**

## **1.1 ¿Qué aporta la informatización de la gestión de trasplantes de órganos?**

La principal ventaja que proporciona la informatización de la gestión de trasplantes de órganos sobre la forma manual, es que es más eficiente en cuanto a tiempo para tomar una decisión, y se tiene guardada la información de una forma organizada y accesible fácilmente.

En todos los proyectos que se han comentado en la introducción, el tiempo es un factor muy importante, y el objetivo principal de este sistema es minimizarlo. Con el sistema que se viene utilizando hasta ahora, se necesitaría una persona en cada hospital que pudiese acceder a la información deseada. La comunicación entre estas personas y el acceso a la información, sería mucho más lenta que la producida por los agentes.

Otro factor importante, es el mantenimiento de la documentación asociada a toda esta gestión. Con la informatización de este sistema, se guarda la información de una forma organizada, que a su vez hace más difícil la pérdida de datos y se accede de forma más rápida a la información deseada.

La finalidad del sistema informático es presentar una información en un tiempo mínimo, para que el médico pueda tomar una decisión adecuada.

## **1.2 Ventajas que aporta la utilización de un sistema multi-agente.**

-Se pueden distribuir tareas de forma fácil. A cada agente se le asocia unas acciones que facilitan su implementación y la organización del sistema.

-Es más eficiente por los puntos comentados antes y porque se obtiene paralelismo a la hora de realizar varias solicitudes al mismo tiempo. (ej.: Se pueden cargar dos agentes iguales en el sistema, haciendo la misma solicitud sobre los dos agentes).

-Es más fiable debido a la redundancia que puede haber de código de programación y de los datos que utiliza el sistema.

-Es muy flexible, ya que se pueden incorporar de una forma sencilla agentes al sistema, añadiendo nueva funcionalidad.

## **1.3 ¿Qué podemos encontrar en esta documentación?**

En el capítulo 2, se explica el concepto de agente y de sistema multi-agente, y qué características pueden tener.

En el capítulo 3, se comenta qué es la FIPA, y los estándares que ha aportado al campo de los sistemas multi-agentes.

En el capítulo 4, se describe el entorno que proporciona JADE para la ejecución de agentes, y las herramientas que contiene para implementarlos.

En el capítulo 5, se describen los componentes que forman parte del sistema multi-agente de este proyecto.

El capítulo 6 explica cómo responde el sistema a las posibles peticiones del usuario.

El capítulo 7 muestra el contenido de la base de datos en dos casos diferentes: uno al hacer todas las peticiones del punto 6.2.1, partiendo de la base de datos vacía; y el otro muestra el contenido que tiene la base de datos, sobre el que se ha hecho el juego de pruebas del punto 6.2.2

El capítulo 8 contiene el código fuente del sistema multi-agente de este proyecto.

En el capítulo 9, se da una pequeña explicación de cómo se pone en marcha el sistema desde una plataforma Windows.

El capítulo 10 menciona algunos puntos sobre los que hace falta seguir trabajando, relativos al Sistema Multi-Agente que gestiona el trasplante de órganos en general.

En el capítulo 11, he hecho un pequeño comentario de lo que ha aportado los sistemas multi-agentes al campo de la informática en general. También he hecho una valoración personal de lo que me ha aportado la realización de este proyecto en el ámbito personal y académico.

## **2. Agentes y Sistemas Multi-Agente**

### **2.1. Definición de agente**

No hay una definición precisa de agente, pero se puede dar una lista de características que pueden tener y ser considerados agentes software ó hardware:

-**Autonomía:** Capacidad que tienen los agentes de actuar sin la intervención directa de los humanos.

-**Cooperación:** Capacidad que tiene un agente para comunicarse con otros agentes y conseguir un fin determinado.

-**Reactividad:** Capacidad que tiene un agente para reaccionar ante estímulos activando reglas.

-**Pro-actividad:** Capacidad que tiene un agente de tomar la iniciativa a la hora de realizar una acción. (No es esencial para que se considere agente).

-**Movilidad:** Capacidad que tiene un agente para moverse por una red electrónica. (No es esencial para que se considere agente).

-**Razonamiento y aprendizaje:** Capacidad que tiene un agente para mejorar su actuación. (No es esencial para que se considere agente).

-**Continuidad temporal:** Los agentes están continuamente ejecutando procesos.

-**Veracidad:** Un agente no comunicará información falsa premeditadamente.

-**Benevolencia:** Los agentes no tienen objetivos conflictivos entre si, cada agente intentará hacer aquello para lo que es requerido.

\*Una definición de agente podría ser la siguiente:

Un agente es un sistema computacional que habita en un entorno complejo y dinámico. El agente puede sentir y actuar autónomamente en su entorno, y tiene un conjunto de objetivos y motivaciones que intenta conseguir a través de sus acciones. **(definición realizada por Pattie Maes).**

Yo me atrevería a decir, que la definición de agente puede ser comparable a la de programa. Ya que, la finalidad de un agente, es añadir funcionalidad a un SMA, como un programa lo hace sobre un ordenador. Por este motivo, quizás sea difícil definir el término agente a partir de unas propiedades mínimas, porque un agente es susceptible de ser programado para hacer cualquier cosa. La diferencia sería que un agente es la unidad básica de un sistema multiagente (SMA), y funciona sobre una plataforma (agent platform), que proporciona unas propiedades diferentes a la de una plataforma de más bajo nivel (el ordenador).

## **2.2. Tipos de arquitecturas**

Un agente ó sistema multi-agente (SMA) se puede englobar dentro de las tres siguientes arquitecturas a la hora de implementarse:

**-Deliberativas:** A partir de una representación del mundo real, se pueden ejecutar acciones inteligentes. Siguen el modelo BDI (Beliefs, Desires and Intentions).

**-Reactivas:** Un agente responde con una acción ante un estímulo determinado.

**-Híbridas:** Estructuras que mezclan diferentes técnicas. Por ejemplo un SMA reactivo con otro SMA deliberativo.

## **2.3 Tipos de agentes**

Los agentes se pueden asociar a una ó varias características, que se detallan a continuación:

**-Colaboradores:** Los agentes se comunican entre ellos para cooperar o negociar, y resolver un problema común.

**-Información / Internet:** Tienen acceso a datos de internet y pueden manipular la información obtenida para responder a solicitudes impuestas por el usuario y / o otros agentes.

**-Interfaz:** Sistema dónde los agentes colaboran con un usuario para resolver un problema. Pueden aprender de los hábitos del usuario y realizar sugerencias.

**-Móviles:** Agentes que pueden moverse por una red. Los agentes recogen información ó realizan tareas.

**-Reactivos:** Los agentes actúan / responden a un estímulo determinado por el estado, especificado por un patrón, en el que se encuentran.

**-Heterogéneos:** Integración de dos o más clases diferentes de agentes.

**-Híbridos:** Unión de diferentes tipos de agentes en uno sólo, para minimizar las desventajas que tienen al estar implementados independientemente.

## **2.4. Métodos para comunicar dos agentes. (Agentificación).**

Los siguientes métodos se utilizan para que un programa se pueda considerar agente y pueda comunicarse con otros agentes:

**-Traductor:** Pasar mensajes del lenguaje de agentes al lenguaje del sistema informático.

**-Wrapper:** Añadir código a un programa para que pueda comunicarse con otros agentes.

-Reescritura del programa original.

## **2.5. Métodos de comunicación.**

Un sistema multi-agente son un conjunto de agentes colaboradores, que para conseguir un objetivo común necesitan comunicarse de alguna forma. Los siguientes métodos son ejemplos de técnicas que utilizan los agentes para comunicarse:

**-Sistemas de pizarra:** Los agentes se comunican a través de una estructura de datos común, a la cual todos pueden acceder.

**-Paso de mensajes:** Los agentes se pasan mensajes entre ellos para comunicarse directamente. En este caso se necesita especificar el lenguaje de comunicación, protocolos de comunicación, formato del mensaje que utilizarán...

\*Ejemplos de lenguaje de comunicación que pueden utilizar dos agentes para comunicarse (**ACL Lenguajes de comunicación para agentes**), son los definidos por los dos siguientes estándares:

**FIPA** (Foundation for Intelligent Physical Agents), que es el utilizado en este proyecto.

**KQML** (Knowledge Query and Manipulation Language).

**Características básicas de estos dos estándares:**

- Definen el formato y la semántica de los mensajes.
- Se tienen un conjunto de acciones predefinidas y el usuario puede añadir más.
- Son lenguajes estándar y pueden unirse diferentes aplicaciones con facilidad.

También existen **ACLs ad hoc** , o hechos a medida, por ejemplo: Bee-gent (Toshiba), Agent Talk (NTT),... Definen el formato y la semántica de los mensajes para determinados dominios. Para unir diferentes aplicaciones hace falta un traductor.

## **2.6. Entornos para el desarrollo de Sistemas Multiagente.**

Existe software que facilita los recursos suficientes para implementar, ejecutar y comunicar agentes. Ejemplos de este tipo de software son: JACK Intelligents Agents, JATLite Beta, JAFMAS, ZEUS, JADE,... Este último es el que se ha utilizado en este proyecto. De esta forma, se permite programar agentes sin tener que preocuparnos de la comunicación a bajo nivel que se realizada por estos.

## **2.7. ¿Qué recursos se han utilizado para implementar este proyecto?**

Hasta ahora hemos dado una visión general de lo que es un agente y un sistema multiagente, pero en este punto es bueno recordar qué opciones hemos escogido para la implementación de este proyecto.

En este proyecto se ha utilizado el entorno de desarrollo de sistemas multiagente JADE, y los agentes se han programado en **JAVA 1.2.2\_013**. Se ha modificado Agente especialista, que implementó David Sánchez para su PFC, para permitir que pueda recibir mensajes de un agente externo (en este caso de Ag0); de esta forma, se ha conseguido una arquitectura deliberativa, para el sistema multiagente desarrollado en este proyecto. Tanto los agentes implementados por David Sánchez, como los desarrollados en este proyecto se basan en una arquitectura que tiende a ser deliberativa.

Para la comunicación entre agentes se han seguido las especificaciones que hace la **FIPA**. (Para definir protocolos de comunicación, formato del mensaje, y lenguaje en el que se escribe el contenido de los mensajes (**FIPA-SL0**)). Por este motivo, en los siguientes puntos se profundizará más en el Estándar FIPA y en el entorno **JADE**.

La base de datos se ha implementado en **Access 2000**, y en el punto 9 de este documento se especifican los parámetros con los que hay que configurar el controlador **ODBC** de Windows, para tener acceso a la base de datos desde Ag0.

### **3. Especificaciones realizadas por la FIPA.**

La FIPA (Foundation for Intelligent Physical Agents) es una asociación registrada en Geneva (suiza) sin ánimo de lucro. Su propósito es promover el éxito emergente en las aplicaciones, servicios y equipamiento basado en agentes. Este objetivo se persigue mediante especificaciones acordadas internacionalmente que maximizan la interoperabilidad entre aplicaciones, servicios y equipamiento basado en agentes. Esto se hace realidad mediante la colaboración internacional abierta de las organizaciones que son miembro, compañías y universidades activas en el campo de los agentes. ( Texto extraído y traducido del documento “FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS, FIPA 97 Specification, Version 2.0 Part 2. Agent Communication Language.”).

La FIPA define tipos de protocolos que utilizan dos ó más agentes para comunicarse:

**FIPA-request Protocol, FIPA-query Protocol, FIPA-request-when Protocol, FIPA-contract-net Protocol, FIPA-Iterated-Contract-Net Protocol, FIPA-Auction-English Protocol, FIPA-Auction-Dutch Protocol, FIPA-Brokering Protocol.**

JADE solo implementa los protocolos que están en negrita. En este proyecto se ha implementado el FIPA-Brokering combinando dos veces el protocolo FIPA-request.

**(En el punto 3.1 se muestra el funcionamiento de los protocolos que están en negrita.)**

La FIPA también especifica el formato de los mensajes que se utilizan en los diferentes protocolos (**ACL Agent Communication Language**), y define el contenido de estos mensajes y el lenguaje utilizado (en este proyecto se ha utilizado el lenguaje FIPA-SL0 que utiliza lógica de primer orden).

**(En el punto 3.2 se detalla el formato de un mensaje ACL, y los tipos de mensajes que puede haber).**

La FIPA define el siguiente entorno básico sobre el que funcionarán y se gestionarán los diferentes agentes:

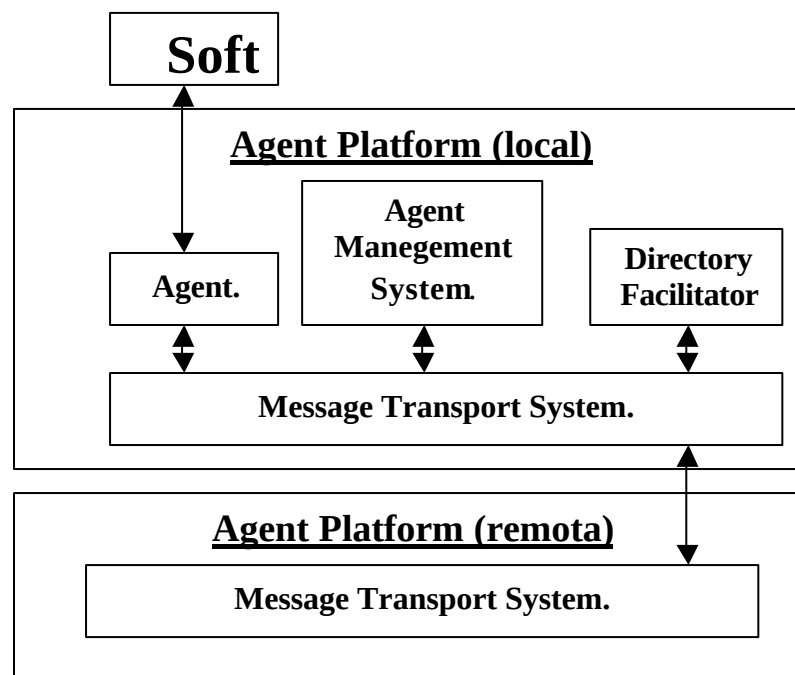


**Figura 3.1 Entorno básico para ejecutar agentes.**



Este entorno básico es el que utiliza JADE. En este proyecto sólo hay un Agent Platform, y tendremos todos los agentes en una sola máquina. Pero se puede observar que pueden ejecutarse varios agentes en diferentes plataformas; la idea sería ejecutar un agente CTH en cada hospital, y crear un Agent Platform en cada uno de ellos, ó crear un Agent Platform compartido por los equipos que haya en cada hospital.

Este entorno básico está formado por los siguientes componentes:

- Agent: unidad básica que compone la plataforma. Más arriba hemos intentado dar una idea de lo que es un agente, pero se podría decir que es un programa que ofrece unos determinados servicios. En el dibujo sólo aparece un Agent, pero pueden haber varios corriendo sobre una misma plataforma. En este proyecto todos los agentes se han ejecutado sobre una misma plataforma.
- Directory Facilitator (DF): Agente que proporciona un servicio de páginas amarillas dentro del sistema (conoce los servicios ofrecidos por los diferentes agentes que se están ejecutando en la plataforma). Es necesario que los agentes se registren en el DF para ofrecer sus servicios.
- Agent Management System (AMS): Agente que ofrece un servicio de páginas blancas. Guarda todas las direcciones de los agentes, y controla el acceso y uso de la plataforma.
- Message Transport System (MTS): Facilita la comunicación entre agentes de diferentes plataformas.
- Agent Platform (AP): Estructura básica en la que pueden crearse y ejecutarse los agentes.
- Software externo al Agent Platform: por ejemplo una base de datos puede ser gestionada por un agente, pero es un software externo a la plataforma. El agente Ag0, implementado en este proyecto, es un ejemplo.

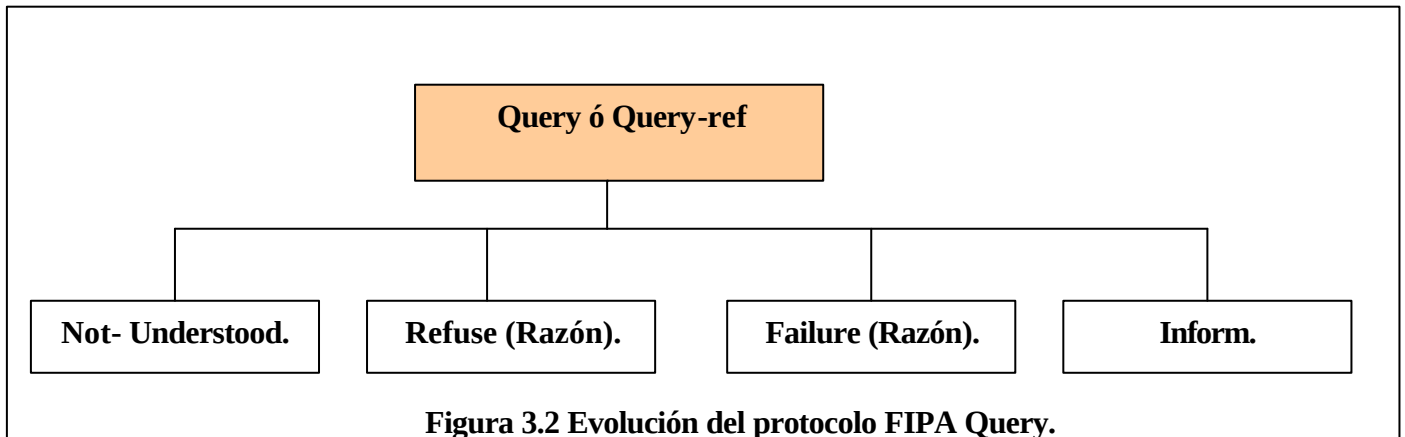
### **3.1 Descripción de los protocolos implementados en JADE.**

Normalmente, cuando dos agentes se comunican, siguen unos patrones predeterminados de conversación, igual que los tenemos los humanos. Hay patrones que se repiten a menudo, y por eso se han desarrollado unos protocolos que definen el tipo de información que se intercambian dos agentes durante el acto comunicativo. En el punto anterior se han enumerado los protocolos estandarizados por la FIPA, pero en este punto sólo veremos los implementados por JADE. En este proyecto sólo se ha utilizado el protocolo FIPA-request.

### 3.1.1 Protocolo FIPA-Query.

Es el protocolo más sencillo que define la FIPA. Un agente pide información a otro (**query** ó **query-ref**), que puede contestarle facilitándole la información (**inform**), diciéndole que no entiende el mensaje recibido (**not understood**), rechazando la solicitud indicando una razón (**refuse**), ó comunicarle que se ha intentado realizar acción, pero que no se ha podido finalizar por algún motivo (**failure**).

A continuación se describe gráficamente:



En los gráficos utilizados para describir los protocolos, se sigue el siguiente criterio:



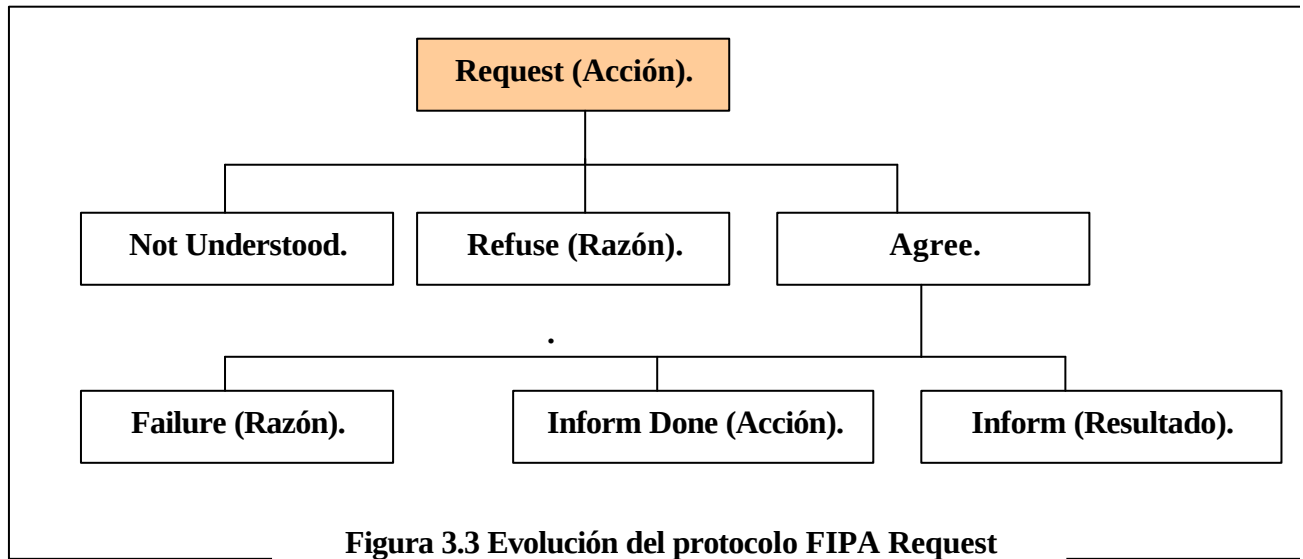
### 3.1.2 Protocolo FIPA-request.

Este protocolo lo utiliza un agente para solicitar a otro que realice una acción (**request**), y el otro agente puede contestarle:

-**Not-understood** si no entiende el mensaje.

-**Refuse** si no puede realizar acción.

-**Agree** si accede a realizar acción. Y después de este mensaje enviará un **Inform** dando el resultado ó simplemente indicando que se ha realizado acción, ó un **failure** si se ha producido cualquier problema.



### 3.1.3 Protocolo FIPA-Contract Net.

Este protocolo lo utiliza un agente cuando quiere negociar con otros agentes, para escoger finalmente el que mejor servicio le puede ofrecer.

El agente que inicia el protocolo empezará enviando un mensaje **cfp**, e indica la acción a realizar y las precondiciones que tendrá que cumplir el agente para ser escogido. Seguidamente, los agentes candidatos enviarán un mensaje **Propose** donde se especifican las condiciones que ofrecen. También pueden enviar un mensaje **Not-Understood** si no han entendido el mensaje que han recibido del agente inicial, ó bien un mensaje **Refuse** indicando la razón por la cual rechazan realizar la acción.

Si hay algún agente que ha enviado un mensaje **Propose** dentro de un tiempo límite, al agente que ha iniciado el protocolo; este último le puede contestar con un mensaje **Reject-proposal** indicando una razón del rechazo de la propuesta, ó con un mensaje **Accept-proposal** en caso de que acepte la propuesta. Aunque el agente inicial haya enviado un **Accept-Proposal**, puede cancelar la acción indicando la razón en un mensaje **Cancel**; si no es así, el agente escogido contestará con un **Failure** indicando una razón del fallo, ó con un **inform done** para que el agente iniciador del protocolo tenga constancia de que se ha realizado la acción.

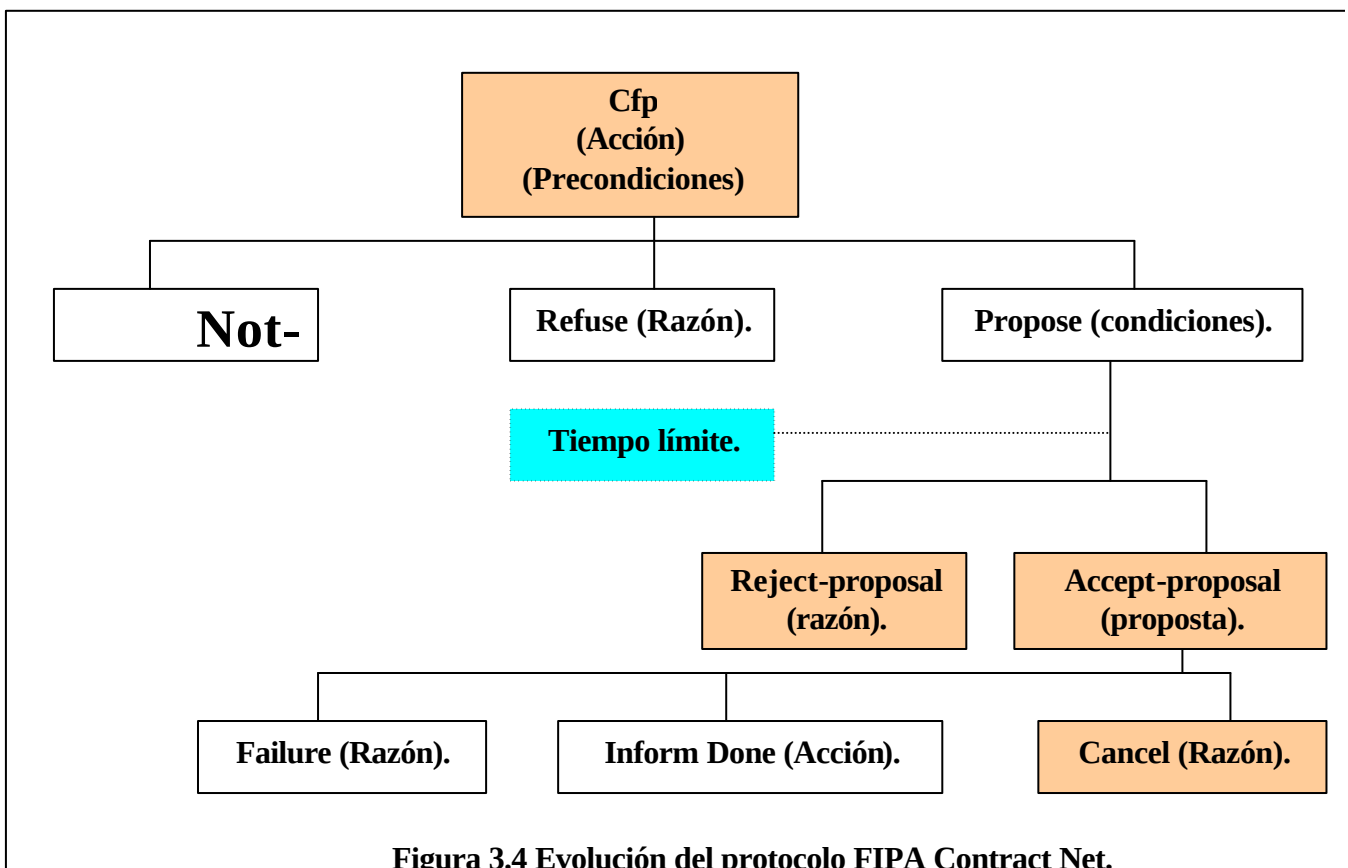


Figura 3.4 Evolución del protocolo FIPA Contract Net.

### **3.2 Formato de los mensajes ACL (Agent Communication Language).**

La FIPA ha establecido que el formato de un mensaje ACL ha de tener la siguiente estructura básica:

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <b>( Inform</b>     |                                                 |
| <b>:sender</b>      | Agente inicial.                                 |
| <b>:receiver</b>    | Agente receptor.                                |
| <b>:content</b>     | (Dar Alta (Paciente...)).                       |
| <b>:in-reply-to</b> | referencia a mensaje recibido con anterioridad. |
| <b>:reply-with</b>  | referencia de mensaje enviado.                  |
| <b>:language</b>    | FIPA-Sl0.                                       |
| <b>:ontology</b>    | Ontologia_Trasplants_GruSMA.                    |
| <b>)</b>            |                                                 |

**Figura 3.5 Mensaje ACL**

La estructura que compone el mensaje ACL empieza indicando el tipo de acto comunicativo del que se trata, en este caso es un Inform. La siguiente tabla muestra los actos comunicativos que reconoce la FIPA y se relaciona con la acción que llevan implícitas.

| <b><u>Acto comunicativo</u></b> | <b>Facilitando información.</b> | <b>Solicitando información.</b> | <b>Negociando.</b> | <b>Realizando acción.</b> | <b>Manejando error.</b> |
|---------------------------------|---------------------------------|---------------------------------|--------------------|---------------------------|-------------------------|
| <b>Accept-proposal</b>          |                                 |                                 | X                  |                           |                         |
| <b>Agree</b>                    |                                 |                                 |                    | X                         |                         |
| <b>Cancel</b>                   |                                 |                                 |                    | X                         |                         |
| <b>Cfp</b>                      |                                 |                                 | X                  |                           |                         |
| <b>Confirm</b>                  | X                               |                                 |                    |                           |                         |
| <b>Disconfirm</b>               | X                               |                                 |                    |                           |                         |
| <b>Failure</b>                  |                                 |                                 |                    |                           | X                       |
| <b>Inform</b>                   | X                               |                                 |                    |                           |                         |
| <b>Inform-if</b>                | X                               |                                 |                    |                           |                         |
| <b>Inform-ref</b>               | X                               |                                 |                    |                           |                         |
| <b>Not-understood</b>           |                                 |                                 |                    |                           | X                       |
| <b>Propose</b>                  |                                 |                                 | X                  |                           |                         |
| <b>Query-if</b>                 |                                 | X                               |                    |                           |                         |
| <b>Query-ref</b>                |                                 | X                               |                    |                           |                         |
| <b>Refuse</b>                   |                                 |                                 |                    | X                         |                         |
| <b>Reject-proposal</b>          |                                 |                                 | X                  |                           |                         |
| <b>Request</b>                  |                                 |                                 |                    | X                         |                         |
| <b>Request-when</b>             |                                 |                                 |                    | X                         |                         |
| <b>Request-whenever</b>         |                                 |                                 |                    | X                         |                         |
| <b>subscribe</b>                |                                 | X                               |                    |                           |                         |

**Figura 3.6 Tabla que relaciona acto comunicativo con función que ejerce dentro de un protocolo.**

A continuación se especifica el identificador del agente que envía el mensaje en el campo **sender**.

En el campo **receiver** contiene un conjunto de identificadores de agentes, que serán los que recibirán el mensaje.

En el campo **content** se introduce el contenido del mensaje.

En **reply-with** se da una referencia del mensaje que se acaba de enviar, para que el otro agente pueda especificarlo en **in-reply-to** de un nuevo mensaje, y así el primer agente saber relacionarlo con el mensaje inicial que ha enviado.

En **language** se especifica la codificación que se ha utilizado en el **content** del mensaje. Existen cuatro tipos de **language** predefinidos:

- **FIPA-CCL** (Constraint Choice Language): define una semántica que permite especificar predicados con satisfacción de restricciones (CSP).
- **FIPA-SL** (Semantic Language): permite formar expresiones lógicas, intercambiar conocimiento y expresar acciones a realizar. Es un lenguaje muy general y puede aplicarse a muchos dominios diferentes. Es el que se utiliza en este proyecto, concretamente FIPA-SL0.
- **FIPA-KIF** (Knowledge Interchange Format): permite expresar objetos como términos y proposiciones como sentencias.
- **FIPA-RDF** (Resource Description Framework): permite representar objetos, interacciones y acciones entre ellos.

En **ontology** se especifica el sentido que se le ha de dar a la estructura que compone la expresión de **content**.

Un mensaje ACL también contiene los campos: **envelope** , **protocol** (para especificar un protocolo que se está utilizando), y **conversation-id** (para especificar un identificador de conversación).

#### **4. Entorno de ejecución de agentes JADE** **(Java Agent Development Enviroment).**

En el punto 3 se describió el entorno básico que define la FIPA para poder ejecutar agentes. JADE es un software, compuesto por paquetes escritos en java, que implementa este entorno básico, además de ofrecer unas utilidades para poder hacer una depuración a nivel de agentes, como por ejemplo:

- Agregar, eliminar, ejecutar, parar... agentes.
- Enviar mensajes a agentes.
- Ver agentes que están registrados en el sistema (en el DF).
- Seguir evolución de mensajes que se envían los agentes. Esta acción se realiza mediante el agente Sniffer.

Los principales paquetes que componen JADE son:

-jade.core: núcleo del sistema, que proporciona clases como jade.core.Agent, que es necesaria para implementar agentes, y jade.core.behaviour que contiene las clases de comportamientos soportados y es imprescindible en la ejecución de agentes, ya que un agente tiene que tener asociado como mínimo un comportamiento.

-jade.lang: formado por los paquetes jade.lang.acl y jade.lang.sl, soporta los lenguajes de comunicación entre agentes (ACL) que define la FIPA.

-jade.proto: contiene los protocolos de comunicación entre agentes que describimos en la sección 3.1.

-jade.domain: Permite la ejecución de elementos básicos de gestión del sistema como son:

**ACC**: Elemento que permite comunicar dos plataformas de agentes (AP).

**DF**: Agente que ofrece servicio de páginas amarillas.

**AMS**: Agente que ofrece servicio de páginas blancas.

-jade.tools: Proporciona herramientas de desarrollo de las aplicaciones, y administración de la plataforma como las siguientes:

**RMA**: Agente que se ocupa de mostrar la interfaz gráfica asociada a una plataforma.

**Sniffer**: Agente que permite visualizar mensajes que se envíen los diferentes agentes.

**DummyAgent**: Permite enviar un mensaje a otros agentes.

-jade.gui: proporciona herramientas para construir interfaces gráficas asociadas a los agentes.

La versión de JADE utilizada para este proyecto es la 2.01, con software adicional que facilita el manejo de mensajes SL.

Para poder iniciar el entorno se ha de ejecutar la siguiente línea de comando:

## **java jade.Boot –platform [opciones] [lista de agentes]**

Como opción se puede poner **–gui** para mostrar el entorno gráfico asociado a la plataforma, que tiene el siguiente aspecto cuando se inicia:

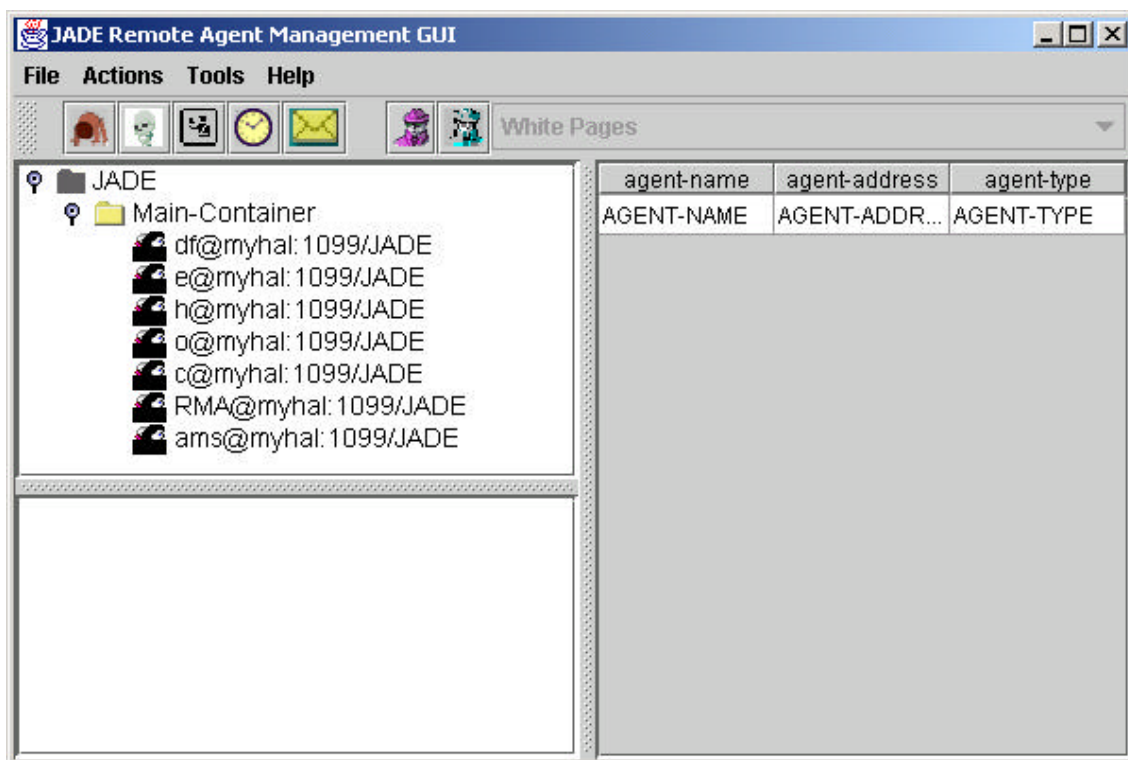


**Figura 4.1 Ventana inicial de entorno gráfico de JADE.**

La lista de agentes es una cadena de texto, donde cada agente se especifica por NombreDeAgente:ClaseJavaQueLoImplementa separados por espacios en blanco.

Cuando se ejecuta JADE como mínimo se obtiene la plataforma con los agentes AMS, DF y ACC en funcionamiento. Si hemos añadido la opción **–gui** también se cargará el agente RMA.

### **4.1 Herramientas proporcionadas por JADE.**

Además del entorno de ejecución básico de agentes, JADE proporciona un entorno gráfico (opción **–gui**), que son útiles para gestionar y monitorizar el comportamiento de los diferentes agentes. Más arriba hemos visto la pantalla inicial de este entorno gráfico.



### 4.1.1 El Menu Actions.

El Menú **Actions** ofrece las siguientes opciones que se muestran en la figura:



Figura 4.2 Ventana inicial de JADE con menú Actions seleccionado.

La opción **Start New Agent** permite cargar un nuevo agente en la plataforma a través de la siguiente ventana, en donde se especifica el nombre del agente y la clase que lo implementa.

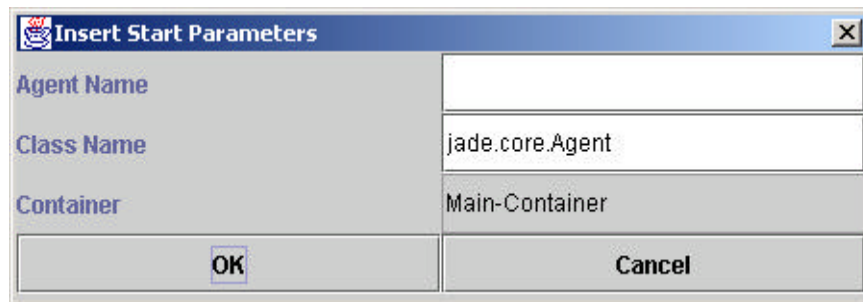


Figura 4.3 Ventana que permite iniciar un agente en JADE.

La opción **Kill** elimina un agente de la plataforma.

La opción **Suspend** para la ejecución de un agente de la plataforma.

La opción **Resume** reanuda la ejecución de un agente de la plataforma.

La opción **Send Message** envía un mensaje a los agentes seleccionados previamente mediante la siguiente ventana:

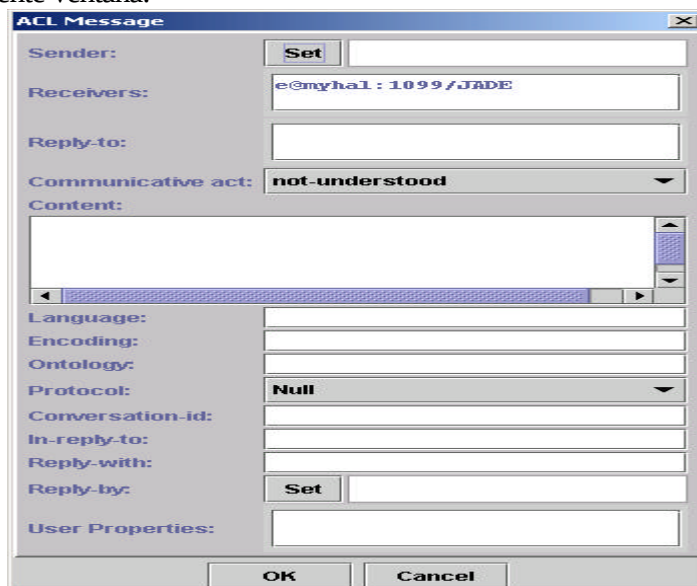


Figura 4.4 Ventana que permite enviar un mensaje a un agente.

### 4.1.2 El menú Tools.

El menú **tools** ofrece las opciones que se ven en la ventana de la siguiente figura:

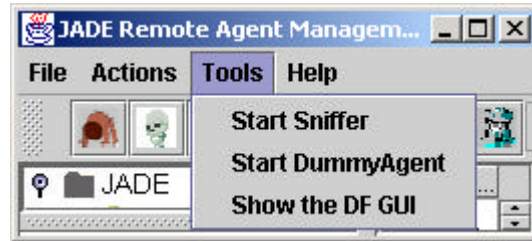


Figura 4.5 Ventana inicial de JADE con menú tools seleccionado.

La opción **Start Sniffer** abre una ventana donde se puede observar la comunicación entre los diferentes agentes seleccionados previamente de una lista, como muestra la siguiente figura:

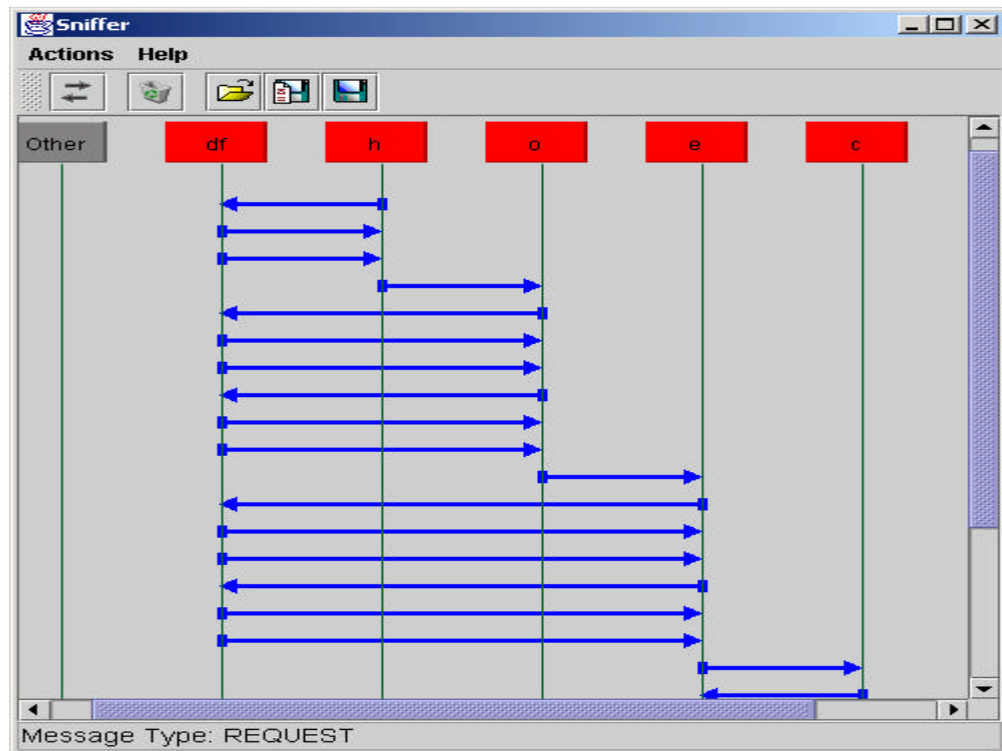


Figura 4.6 Ventana con mensajes que se envían los agentes.

Si seleccionamos una flecha de la figura anterior, se mostrará una ventana con la información que contiene el mensaje que representa la flecha.

La opción **Start DummyAgent** muestra una ventana, que permite construir un mensaje que se enviará a otros agentes.

La opción **Show the DF GUI** presenta una pantalla con los agentes registrados en el DF.

## **4.2 Construcción de agentes en JADE.**

Un agente en JADE es una extensión de la clase `jade.core.agent` con los métodos `setup` y `takeDown` sobrescritos. En el método `setup` se registra el agente en el DF y se especifican los comportamientos asociados al agente, y en el método `takeDown` se borra el agente del DF.

La funcionalidad de un agente está implementada en sus comportamientos. Hay un `scheduler` que va ejecutando todos los comportamientos del agente, y un comportamiento se obtiene instanciando alguna de las clases de la jerarquía de `Behaviours`.

La jerarquía de `behaviours` se divide en:

- `SimpleBehaviour`, que engloba a los comportamientos simples que se pueden ejecutar sin interrupción.
- `ReceiverBehaviour`, que permite poner en marcha un comportamiento que espera la recepción de un mensaje. Un ejemplo de este comportamiento sería el implementado en el archivo `RespuestaAsignacionOrgano.java` de `Ag0` de este proyecto. (El archivo se encuentra en el directorio `A0`).
- `SenderBehaviour`, que permite poner en marcha un comportamiento cuando el agente va a enviar un mensaje. Un ejemplo de este comportamiento es el implementado en el `agentCTH`, concretamente el `FIPAREQUESTINITIATORBEHAVIOUR` implementado en el archivo `AnotacionAsignacionOrganoRequest.java` de este proyecto. (El archivo se encuentra en el directorio `CTH`).
- `ComplexBehaviour`, que se componen de diversos hijos, y se pueden interrumpir entre la ejecución de dos hijos.

Otro aspecto importante en la implementación de agentes es la ontología que se le asocia para poder estructurar la información que se envía, y poder interpretar la que se recibe de otro agente. La ontología es una representación del conocimiento mediante un sistema de `Frames`. La ontología se compone de `Frames` y `Slots` básicamente, y de etiquetas para poder acceder a los `Slots`.

Un ejemplo de implementación de ontología sería la especificada en este proyecto, bajo el directorio `...A0/Ontology`. Hay un archivo llamado `OntologiaAg0.java` donde se define y estructura toda la información que compone la ontología del agente `Ag0`. También hay archivos, en este directorio, donde se implementan las clases que componen a los distintos `Frames`, con sus `Slots` y los métodos asociados `set` y `get`, que facilitan el acceso a la información.

## 5. Descripción del sistema.

El proyecto asociado a esta documentación, se encarga de gestionar toda la información relativa a pacientes, que requieren que se les trasplante un órgano de forma urgente.

Para decidir a que paciente se le asigna un determinado órgano, necesita la colaboración del sistema multiagente implementado por David Sánchez.

En este proyecto se han implementado dos agentes, el AgentCTH y el Ag0. AgentCTH es el encargado de mostrar una interfaz gráfica, para interactuar con el usuario; y Ag0 es el encargado de gestionar la base de datos, que tiene el sistema, con información de pacientes que necesitan que se le trasplante un órgano de forma urgente. Se ha modificado el agente especialista del proyecto de David Sánchez, para que pueda recibir mensajes de un agente externo (en nuestro caso de Ag0).

El siguiente dibujo da una visión general de los agentes que se han implementado en este proyecto y de su funcionalidad:

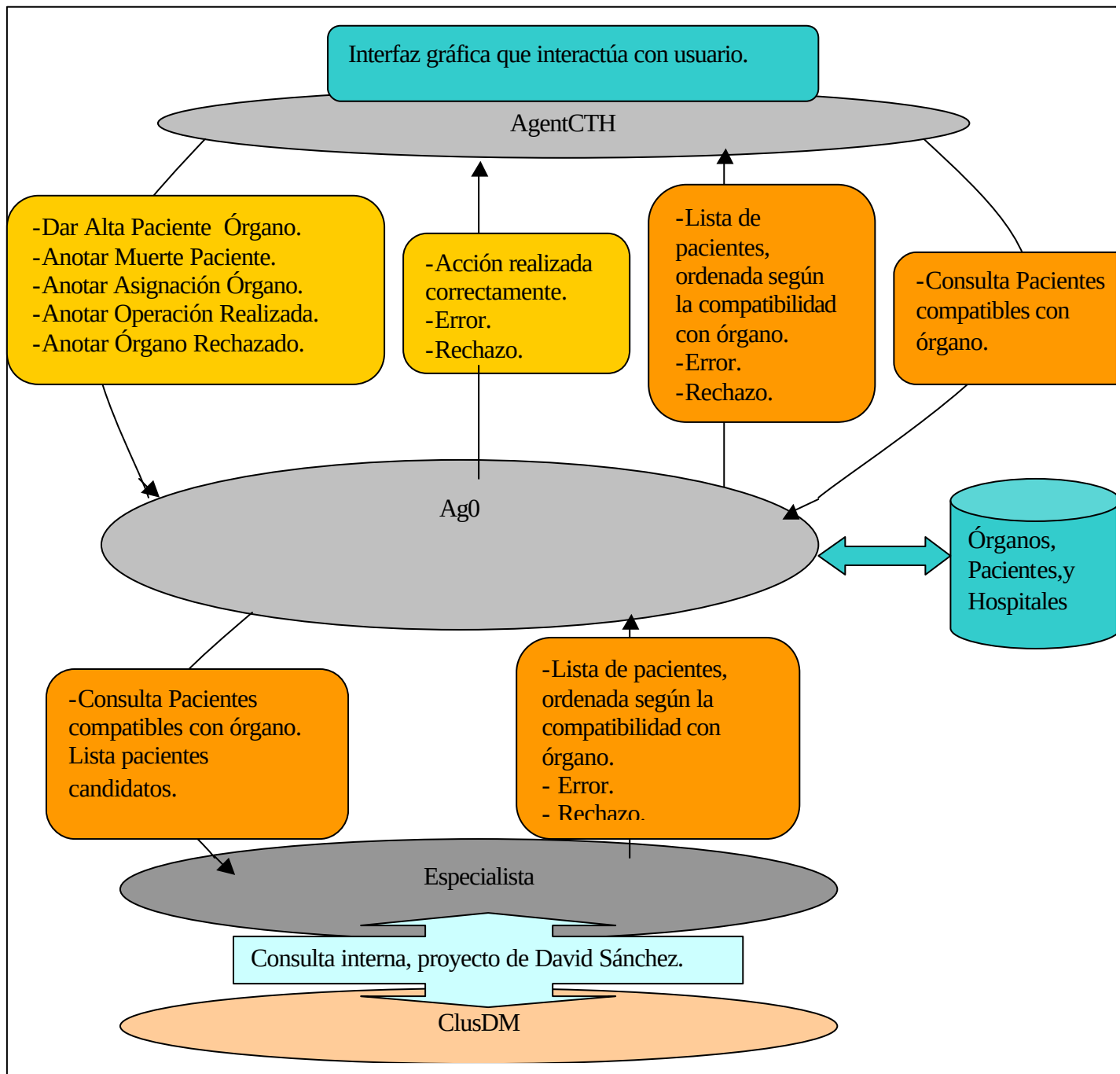


**Figura 5.1** Visión general de sistema multiagente desarrollado en este proyecto.

El dibujo anterior muestra los agentes y tipo de mensajes que se envían de una forma muy general. La acción Encontrar pacientes compatibles con un órgano se ha dibujado con un color diferente al resto de acciones, porque tiene un comportamiento diferente; en este caso, se han anidado tres comportamientos de tipo Request que involucran a los 4 agentes de la figura anterior, mientras que el resto lo componen comportamientos Request simples.

Se ha dibujado con óvalos a los agentes. AgentCTH y Ag0 tienen el mismo color para indicar que se han programado desde el principio en este proyecto. Especialista, con otro color, para indicar que se ha modificado para poder recibir mensajes desde el agente Ag0. Y ClusDM con color diferente al resto, para indicar que lo programó David Sánchez en su proyecto y no se ha tenido que modificar.

A través de la interfaz gráfica, el coordinador de trasplantes, que es el usuario final de este sistema multi-agente, podrá solicitar que se realicen las siguientes acciones:

- Introducir paciente donante, órganos donados, y si ha fallecido.
- Introducir paciente receptor y órganos que espera que se le trasplanten.
- Asignar órgano a un paciente, para que se le trasplante.
- Anotar que se le ha trasplantado un órgano a un paciente.
- Anotar que un paciente ha rechazado un órgano que se le había trasplantado.
- Anotar que un paciente ha fallecido.
- Encontrar grado de la compatibilidad de pacientes con un determinado órgano.

En el momento que el usuario solicita realizar alguna de estas acciones, AgentCTH le envía un mensaje a Agent0 para que actualice la base de datos, ó consulte a Especialista para saber el grado de compatibilidad de los pacientes con un determinado órgano.

La siguiente figura muestra los pasos por los que pasa un paciente cuando ha solicitado un órgano:

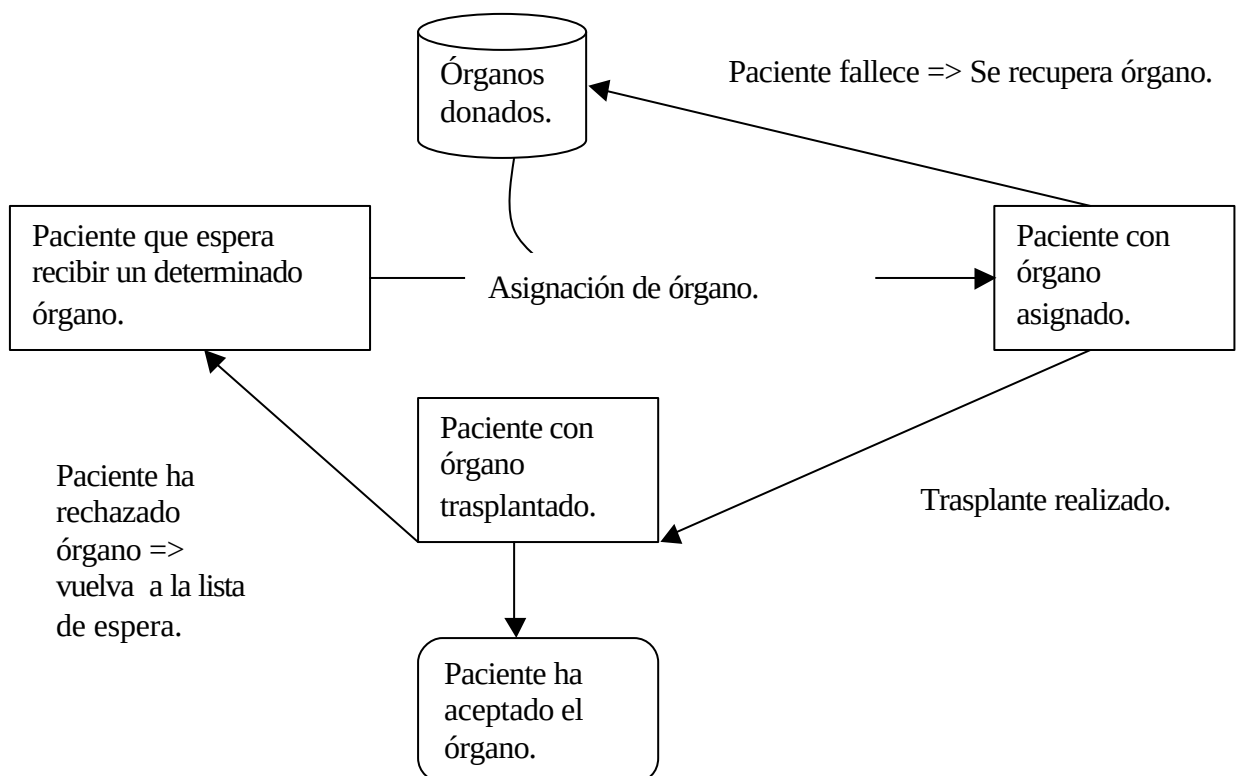


**Figura 5.2 Estados desde que paciente solicita un órgano hasta que se le trasplanta.**

## **5.1 Descripción componentes de AgentCTH**

### **5.1.1 Descripción de la interficie gráfica.**

#### **5.1.1.1 Menú Principal.**

Menú en el que se introducen todos los datos relativos al hospital, en donde el usuario se encuentra.

Si se oprime botón “Siguiente”, cuando se han rellenado todos los datos, pasará al menú opciones.

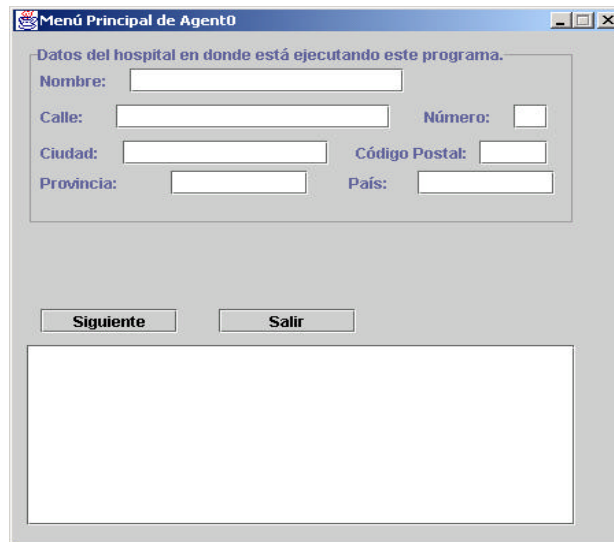


**Figura 5.3 Menú principal de la interfaz gráfica.**

#### **5.1.1.2 Menú Opciones.**

Menú en donde se muestran las acciones que el usuario le puede pedir a AgentCTH.

Entre todas las opciones escogerá una.

Si se oprime botón “Siguiente” pasará a las siguientes ventanas, según la opción escogida.

Si se oprime botón “Anterior” volverá al Menú Principal.

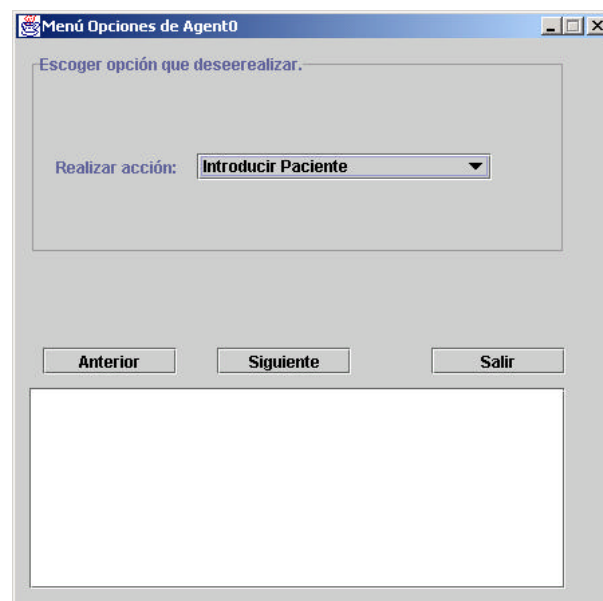


**Figura 5.4 Menú opciones de la interfaz gráfica.**

#### 5.1.1.3 Menú Introducir Paciente.

El usuario llega a esta ventana, si en el menú opciones ha escogido la opción “Introducir Paciente”.

En esta ventana se introducen todos los datos relativos a un paciente. Se especifica si el usuario va a ser Donante o Receptor.

Si se oprime el botón “Anterior” se vuelve al Menú Opciones.

Si se oprime el botón “Siguiete” se pasará al Menú Donante ó Menú Receptor, dependiendo si se trata de un paciente donante ó receptor. (Un paciente puede ser donante y receptor al mismo tiempo).

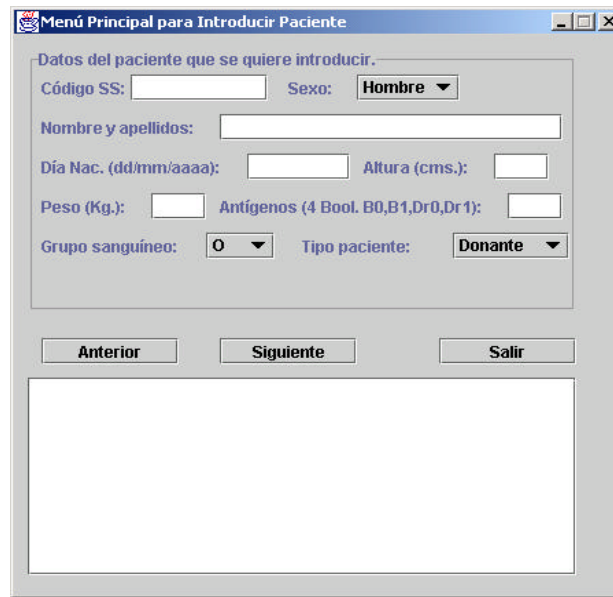


**Figura 5.5**  
**Ventana**  
**principal para**  
**registrar**  
**paciente-órgano**  
**en el sistema.**

#### 5.1.1.4 Menú Donante.

Se presenta esta ventana, cuando usuario ha especificado en el menú introducir paciente, que se trata de un paciente donante.

Se introduce en esta ventana, los datos relativos al órgano donado, y si el paciente donante está vivo ó ha fallecido.

Si donante está vivo y se oprime botón “Siguiete”, AgentCTH enviará un mensaje a Ag0 para que este intente realizar la acción correspondiente (Introducir Paciente). Si el donante ha fallecido, aún queda otra ventana, donde introducir los datos relativos a la muerte del paciente, antes de que AgentCTH envíe el mensaje a Ag0.

Se permanece en este menú, mientras no se oprima el botón “Anterior”. Esto permite que el usuario pueda modificar los datos de la última ventana, y poder introducir más órganos donados sin tener que volver a menús anteriores.

Si se oprime el botón “Anterior”, se vuelve al Menú Opciones.

Datos relativos al organo donado.

Identificador Órgano:

Tipo Órgano: Corazon ▼

Tamaño Órgano (cms.):

Estado paciente: Vivo ▼

Anterior Siguiente Salir

**Figura 5.6**  
Ventana para  
introducir órgano  
donado.

#### 5.1.1.5 Menú Muerte Paciente.

Se llega a esta ventana cuando en el Menú Donante, se ha introducido un paciente muerto, y se piden los datos relativos a la muerte del paciente.

Si se oprime el botón “Siguiente”, AgentCTH enviará un mensaje a Ag0 con acción a intentar realizar (Introducir Paciente), y se vuelve al Menú Donante por si hay más órganos que el paciente ha donado.

Si se oprime el botón “Anterior” se vuelve al Menú Opciones.

Anotar Muerte del Paciente.

Datos sobre la muerte del paciente.

Día Muerte (dd/mm/aaaa):

Hora Muerte (HH:MM):

Anterior Siguiente Salir

**Figura 5.7**  
Ventana para  
indicar momento  
muerte de  
donante.

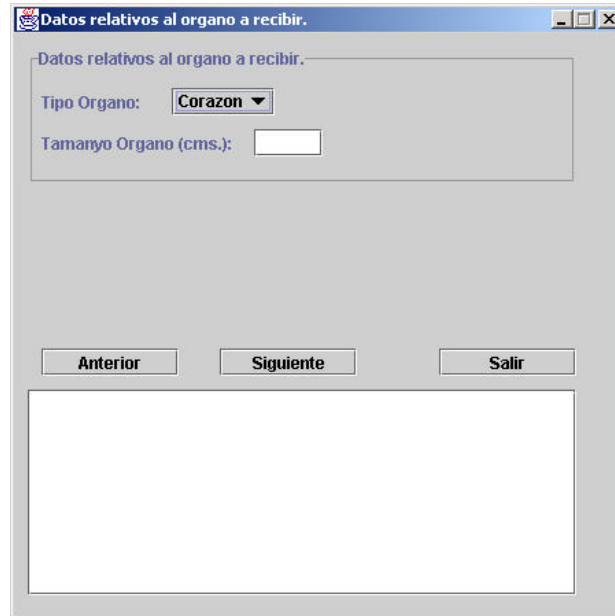


#### 5.1.1.6 Menú Receptor.

Ventana que aparece cuando en Menú Introducir Paciente se ha especificado, que se trata de un paciente que se añadirá a la lista de espera de un determinado órgano.

Se introducen los datos relativos al órgano que se espera recibir.

Si se oprime el botón “Anterior” se vuelve al Menú Opciones, y si se oprime “Siguiente” se pide a Ag0 que intente realizar la acción Introducir Paciente.



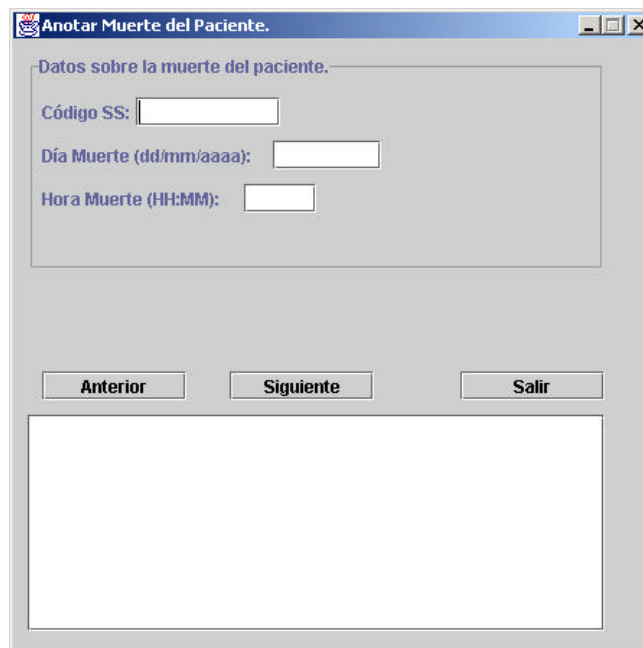
The screenshot shows a window titled "Datos relativos al organo a recibir." with a standard Windows-style title bar. Inside, there is a label "Datos relativos al organo a recibir." followed by two input fields: "Tipo Organo:" with a dropdown menu showing "Corazon" and "Tamanyo Organo (cms.):" with a text box. At the bottom, there are three buttons: "Anterior", "Siguiente", and "Salir". Below the buttons is a large empty rectangular area.

**Figura 5.8**  
Ventana para introducir datos de órgano que se espera recibir.

#### 5.1.1.7 Menú Anotar Muerte Paciente.

Aparece cuando se oprime “Siguiente” en el Menú Opciones y se ha escogido la opción “Anotar Muerte Paciente”.

Se introducen datos relativos a la muerte de un paciente y se envía mensaje a Ag0 para que intente realizar acción (Anotar Muerte Paciente).



The screenshot shows a window titled "Anotar Muerte del Paciente." with a standard Windows-style title bar. Inside, there is a label "Datos sobre la muerte del paciente." followed by three input fields: "Código SS:" with a text box, "Día Muerte (dd/mm/aaaa):" with a text box, and "Hora Muerte (HH:MM):" with a text box. At the bottom, there are three buttons: "Anterior", "Siguiente", and "Salir". Below the buttons is a large empty rectangular area.

**Figura 5.9**  
Ventana para anotar la muerte de un paciente.

#### 5.1.1.8 Menú Encontrar Pacientes Compatibles.

Se llega a esta ventana desde el menú opciones, cuando lo ha solicitado el usuario.

Se introduce identificador de órgano del cual se quieren encontrar pacientes compatibles y pesos asociados a los atributos, que Agent Especialista tendrá en cuenta a la hora de ordenar la lista de pacientes compatibles. Se elige agente para realizar ordenación y función de distancia que utilizará.

**Figura 5.10**  
Ventana para  
buscar pacientes  
compatibles con  
un órgano.

#### 5.1.1.9 Menú Anotar Asignación Órgano.

Se pide identificador de órgano y el paciente al que se le va a asignar.

Cuando se oprime el botón “Siguiente”, se envía mensaje Ag0 para que realice acción (Asignación Órgano Paciente).

**Figura 5.11**  
Ventana para  
asignar un  
órgano a un  
paciente.

#### 5.1.1.10 Menú Anotar Órgano Rechazado.

Se pide identificador de órgano rechazado.

La acción asociada a este menú, que deberá intentar realizar Ag0, es Anotación Órgano Rechazado.

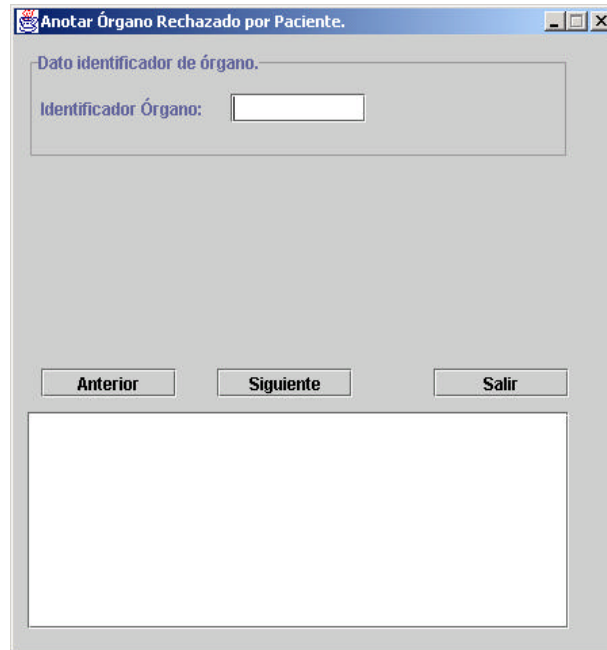


**Figura 5.12**  
**Ventana para**  
**indicar que un**  
**órgano ha sido**  
**rechazado por un**  
**paciente.**

#### 5.1.1.11 Menú Anotar Operación Realizada.

Se pide identificador de órgano, fecha y hora de la operación. La acción asociada a este menú, que deberá intentar realizar Ag0, es Anotación Operación Realizada.

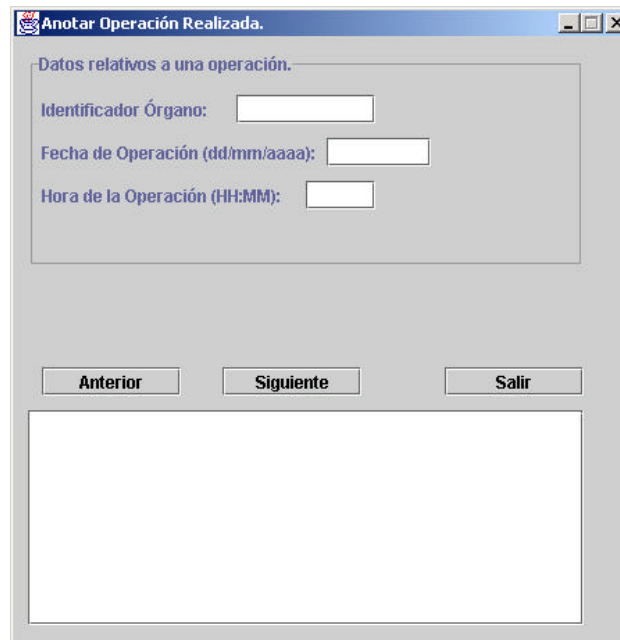


**Figura 5.13**  
**Ventana para**  
**indicar momento**  
**de un trasplante.**

Esta interfaz gráfica es propia de AgentCTH, y en este proyecto es la única parte que interactúa con el usuario. Está implementado bajo el directorio:

**Unidad:\Agent0\agentes\CTH\interficie.**

### 5.1.2 Ontología propia del agente AgentCTH.

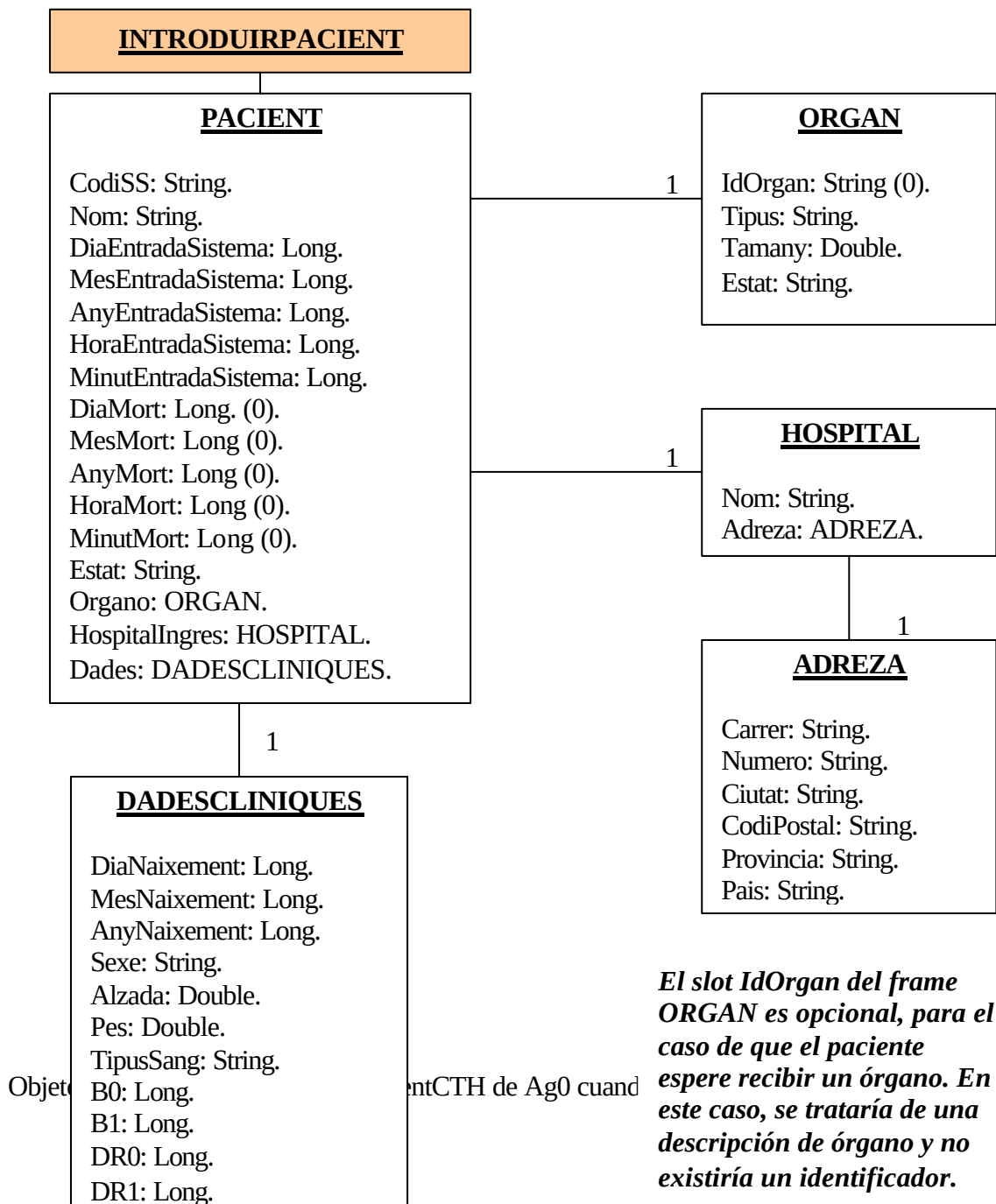
La ontología de AgentCTH se encuentra bajo el directorio:

**Unidad:\Agent0\agentes\CTH\Ontology**

Voy a mostrar la ontología que hay asociada a cada acción que solicita AgentCTH a Ag0; esto quiere decir, que pueden haber componentes repetidos, pero en realidad sólo se definen una vez en la ontología.

#### 5.1.2.1 Ontología asociada a la acción introducir pacientes.

Objetos de la ontología que envía AgentCTH a Ag0 para que introduzca un paciente en su base de datos. ( Un Slot con (0) al lado, indica que es opcional). El protocolo que utiliza AgentCTH, para comunicar a Ag0 que realice esta acción, es un Request.



-ErrorAccesBD.

Objeto de la ontología que recibe AgentCTH cuando Ag0 rechaza realizar la acción:

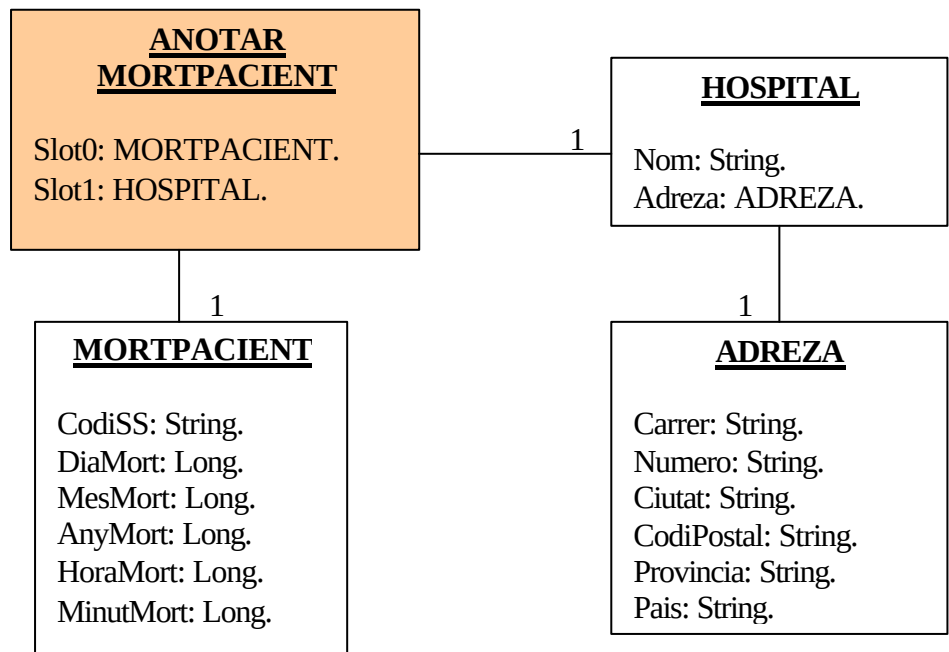
- OrganExistent.
- PatientMort.
- ErrorAccesBD.
- PatientTeOrganAssignat.

Si todo ha ido bien, Ag0 envía a AgentCTH un inform Done.

#### 5.1.2.2 Ontología asociada a la acción anotar muerte paciente.

El protocolo que utiliza AgentCTH, para comunicar a Ag0 que realice esta acción, es un Request.

Objetos de la ontología que envía AgentCTH para que Ag0 anote la muerte de un paciente en la base de datos:



Objeto de la ontología que recibe AgentCTH de Ag0 cuando se ha producido un fallo:

-ErrorAccesBD.

Objeto de la ontología que recibe AgentCTH cuando Ag0 rechaza realizar la acción:

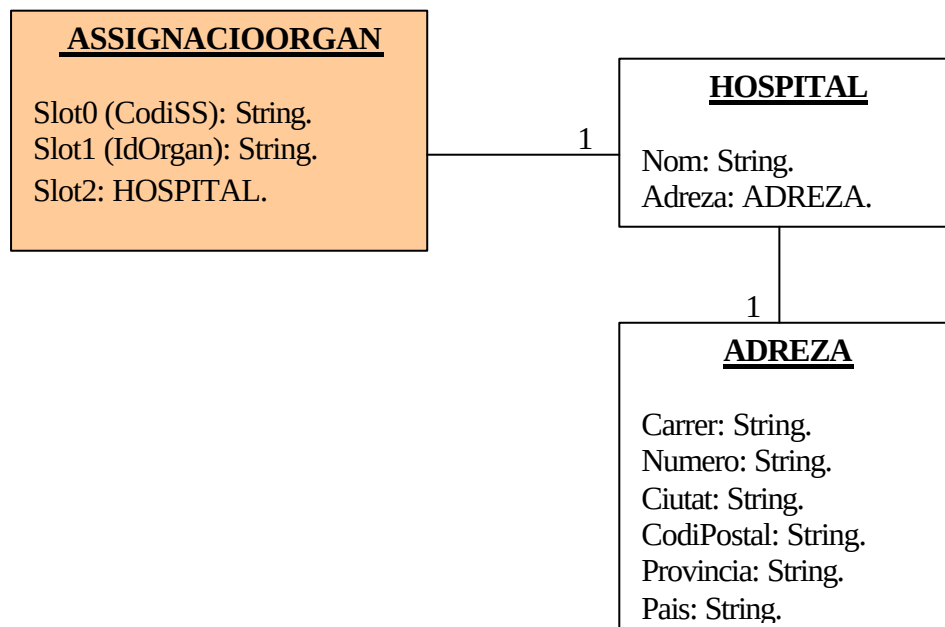
- PatientInexistent.
- PatientMort.
- ErrorAccesBD.
- HospitalSenseAutoritzar.

Si todo ha ido bien, Ag0 envía a AgentCTH un inform Done.

### 5.1.2.3 Ontología asociada a la acción anotar la asignación de un órgano a un paciente.

El protocolo que utiliza AgentCTH, para comunicar a Ag0 que realice esta acción, es un Request.

Objetos de la ontología que envía AgentCTH para que Ag0 anote la asignación de un órgano a un paciente en la base de datos:



Objeto de la ontología que recibe AgentCTH de Ag0 cuando se ha producido un fallo:  
-ErrorAccesBD.

Objeto de la ontología que recibe AgentCTH cuando Ag0 rechaza realizar la acción:

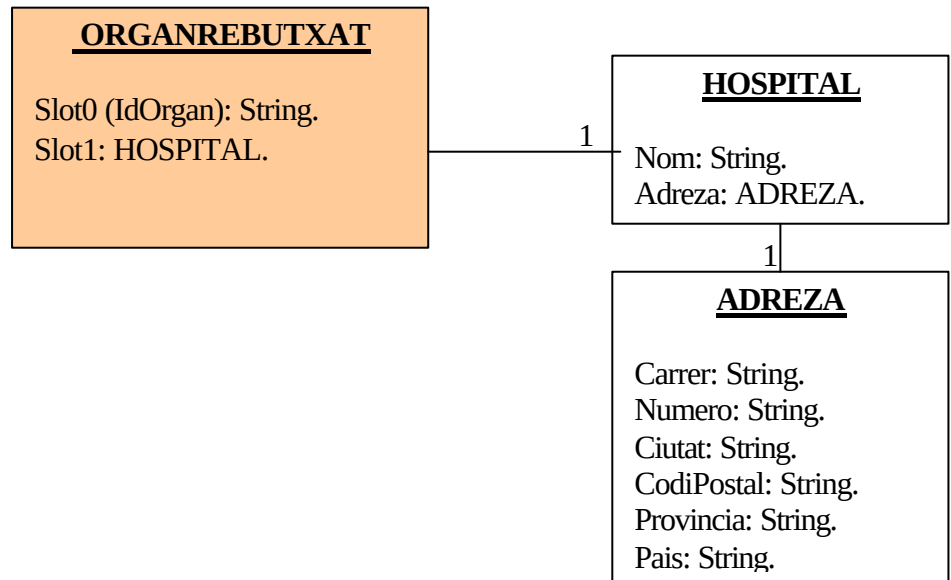
- PacientInexistent.
- PacientMort.
- ErrorAccesBD.
- HospitalSenseAutoritzar.
- OrganInexistent.
- PacientTeOrganAssignat.
- PacientOperat.
- PacientSenseEsperarOrgan.
- OrganNoDisponible.

Si todo ha ido bien, Ag0 envía a AgentCTH un inform Done.

#### 5.1.2.4 Ontología asociada a la acción anotar el rechazo de un órgano por un paciente.

El protocolo que utiliza AgentCTH, para comunicar a Ag0 que realice esta acción, es un Request.

Objetos de la ontología que envía AgentCTH para que Ag0 anote el rechazo de un órgano por un paciente en la base de datos:



Objeto de la ontología que recibe AgentCTH de Ag0 cuando se ha producido un fallo:  
-ErrorAccesBD.

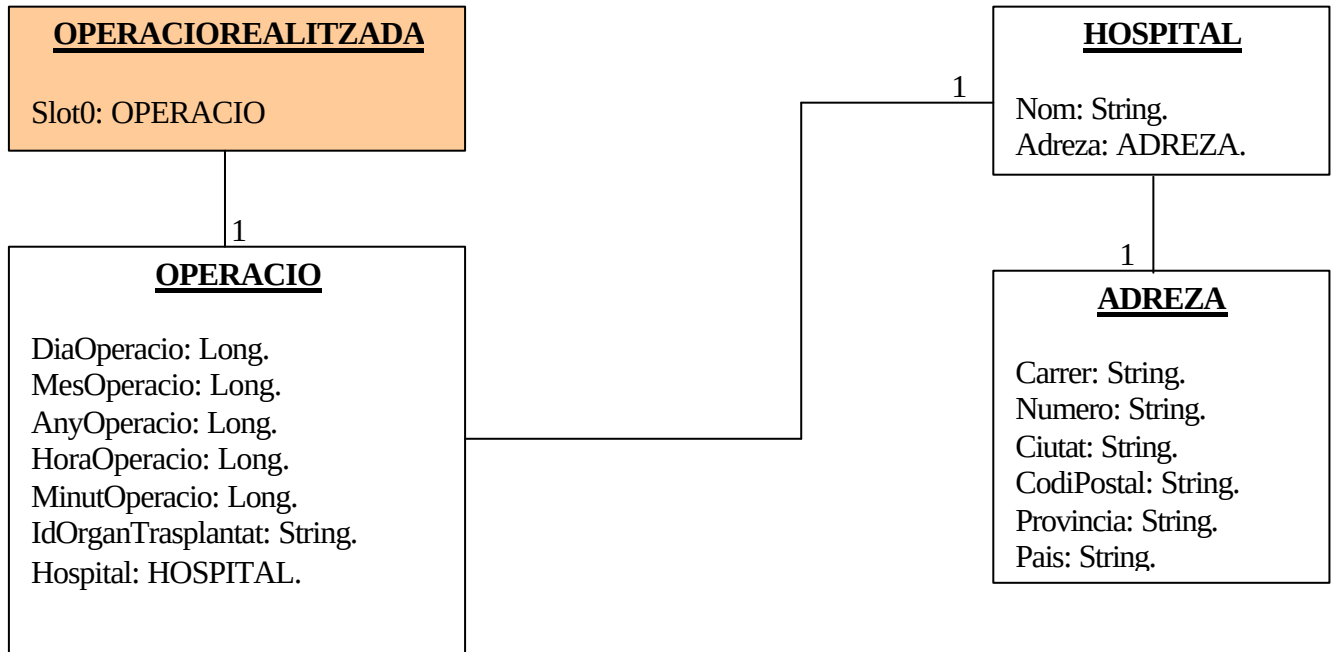
Objeto de la ontología que recibe AgentCTH cuando Ag0 rechaza realizar la acción:  
-PatientInexistent.  
-PatientMort.  
-ErrorAccesBD.  
-HospitalSenseAutoritzar.  
-OrganInexistent.  
-PatientSenseOperar.  
-OrganJaEstavaRebutxat.

Si todo ha ido bien, Ag0 envía a AgentCTH un inform Done.

### 5.1.2.5 Ontología asociada a la acción anotar trasplante realizado a un paciente.

El protocolo que utiliza AgentCTH, para comunicar a Ag0 que realice esta acción, es un Request.

Objetos de la ontología que envía AgentCTH para que Ag0 anote el trasplante de un órgano a un paciente en la base de datos:



Objeto de la ontología que recibe AgentCTH de Ag0 cuando se ha producido un fallo:  
-ErrorAccesBD.

Objeto de la ontología que recibe AgentCTH cuando Ag0 rechaza realizar la acción:

- PacientInexistent.
- PacientMort.
- ErrorAccesBD.
- HospitalSenseAutoritzar.
- OrganInexistent.
- PacientSenseOrganAssignat.
- OrganNoDisponible.

Si todo ha ido bien, Ag0 envía a AgentCTH un inform Done.

### 5.1.2.6 Ontología asociada a la acción encontrar pacientes compatibles con un determinado órgano.

Para realizar esta acción se ejecutan tres protocolos de tipo request; uno que comunica a AgentCTH con Ag0, otro que comunica Ag0 con especialista, y el último que comunica especialista con el agente que realmente realiza la ordenación de pacientes (en este caso ClusDM, pero podría ser cualquiera de los programados por David Sánchez en su proyecto, cuando se le añada el código necesario).

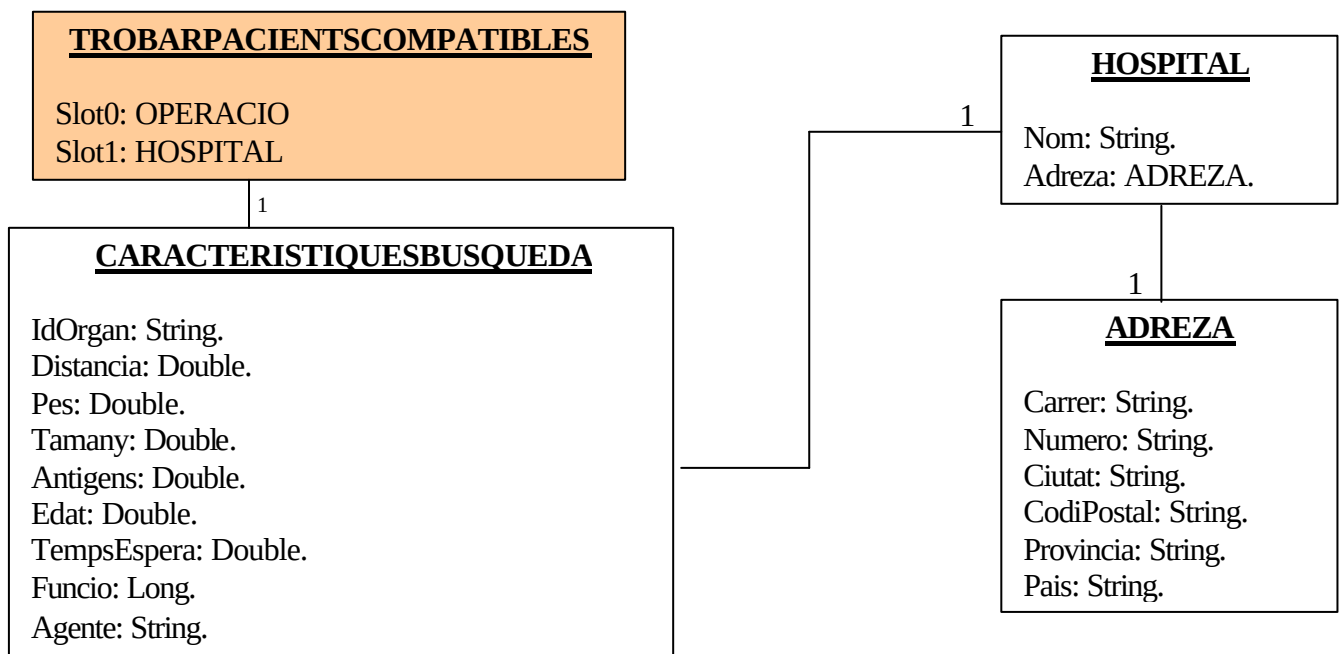
Aquí se ve parte de las ventajas de trabajar con agentes. Se ha modificado el agente especialista del proyecto de David Sánchez, cambiando su interfaz gráfica por código capaz de recibir mensajes de un agente externo, y nos hemos olvidado del resto de agentes. Después de esta modificación, cualquier agente se podrá comunicar con el



sistema multiagente (SMA) desarrollado en el proyecto de David Sánchez, si conoce su ontología (objetos que han de estar en el contenido del mensaje que se envía y se recibe). También se puede añadir funcionalidad al SMA del proyecto de David Sánchez modificando los agentes Dummy, de una forma independiente; o simplemente, añadiendo otros agentes que se comuniquen con especialista. Esta ventaja de programación que aporta el concepto de agente, es aplicable a cualquier SMA.

A continuación, se muestra los objetos de la ontología que envía AgentCTH, para que Ag0 busque en su base de datos, los pacientes candidatos a recibir un órgano del mismo tipo que el especificado en el mensaje recibido; y posteriormente, Ag0 solicitará a Especialista que ordene la lista de los pacientes candidatos según la compatibilidad con dicho órgano. Si todo va bien, la acción finalizará cuando Ag0 le envíe la lista ordenada a AgentCTH; y por este motivo, AgentCTH no recibirá un simple Inform Done, como en los casos anteriores.

Los objetos que envía AgentCTH a Ag0 para realizar esta acción son:



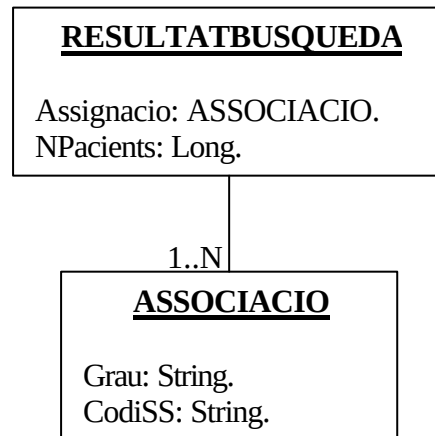
Objeto de la ontología que recibe AgentCTH de Ag0 cuando se ha producido un fallo:

- ErrorAccesBD.
- Además de los producidos por Especialista.

Objeto de la ontología que recibe AgentCTH cuando Ag0 rechaza realizar la acción:

- AgentNoRegistrat.
- OrganInexistent.
- Además de los producidos por Especialista.
- PacientInexistent.
- HospitalSenseAutoritzar.

Los siguientes objetos son los que envía Ag0 a AgentCTH como el resultado de la búsqueda y ordenación de pacientes compatibles con un órgano:



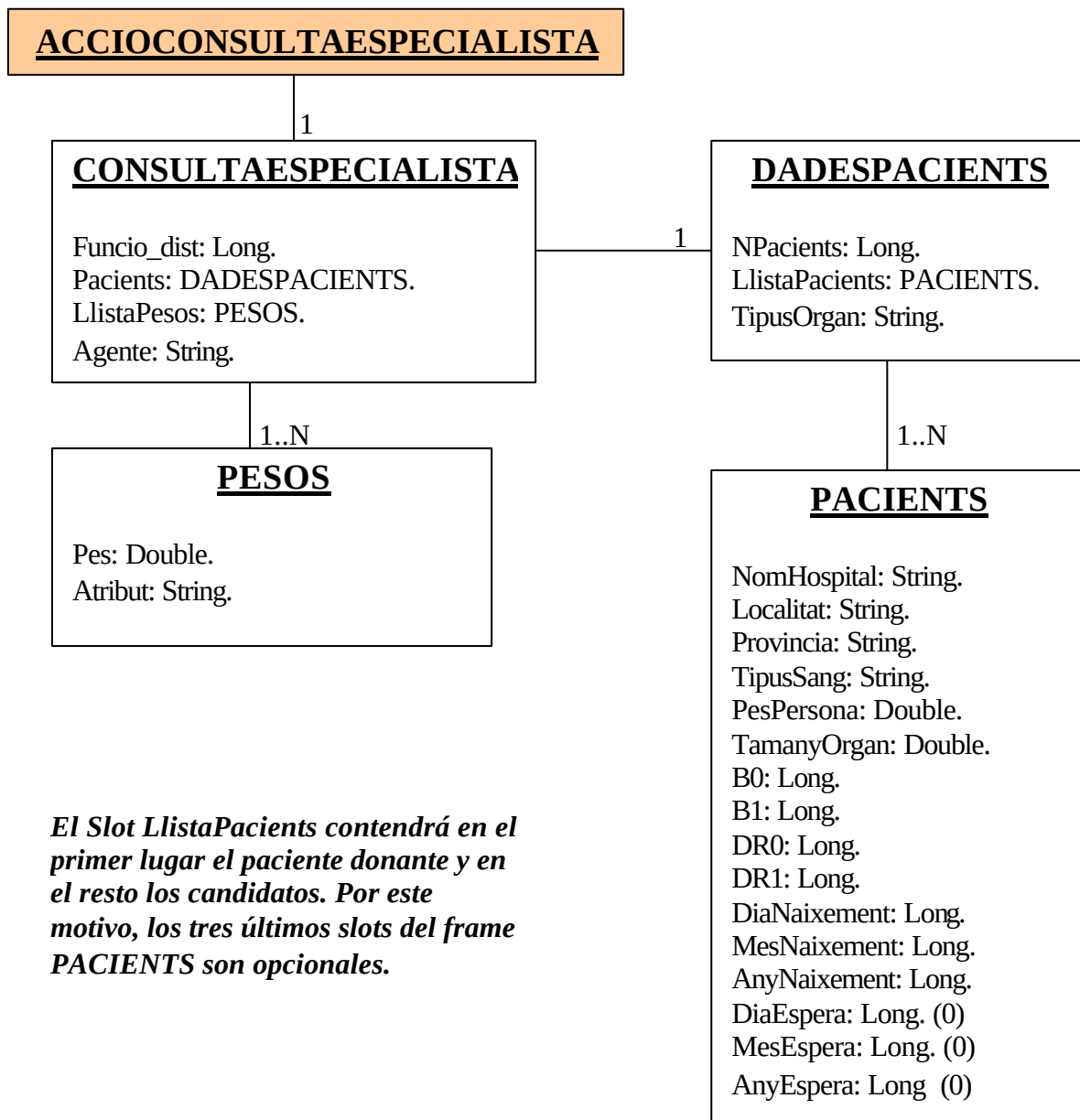
## **5.2 Descripción componentes de Ag0.**

### **5.2.1 Ontología propia del agente Ag0.**

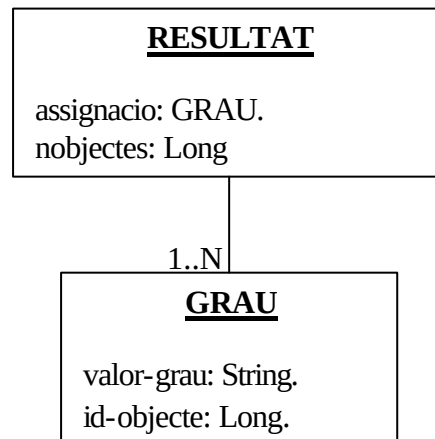
Todos los objetos propios de la ontología de AgentCTH, también han de estar definidos en la ontología de Ag0, para que pueda interpretar los mensajes que recibe y pueda acceder a la información. Es decir, toda la ontología que se expuso en el apartado 5.1.2, también debería de estar en este apartado, pero sólo presentaré en este apartado los objetos de la ontología de Ag0 que faltan; estos objetos son los necesarios para poder comunicar Ag0 con Especialista.

#### **5.2.1.1 Ontología asociada a la acción encontrar pacientes compatibles con un determinado órgano.**

Objetos de la ontología que envía Ag0 a Especialista para que realice la ordenación de los pacientes, según su compatibilidad con el órgano pasado en el mensaje:



Objetos que recibe Ag0 de Especialista, cuando solicita sus servicios y no ha habido ningún problema.



Objeto de la ontología que recibe Ag0 de Especialista cuando se ha producido un fallo, y Ag0 se lo enviará a AgentCTH:

-ClassesInsuficients: No se ha podido crear suficientes clases para realizar ordenación. (Con clases nos referimos a agrupación de pacientes, que hace el agente ClusDMResponderAgent cuando realiza el proceso de ordenación).

Objeto de la ontología que recibe Ag0 cuando Especialista rechaza realizar la acción, y Ag0 se lo enviará a AgentCTH:

- PesosNoValids: Los pesos de los atributos no suman 100%.
- MassaObjectes: Hay demasiados pacientes a ordenar (Máximo 100).
- AlgoritmeNoImplementat: El algoritmo asociado al agente que debe realizar ordenación todavía no está implementado (Es un agente Dummy).
- MassaAtributs: Demasiados atributos a tener en cuenta para realizar ordenación (Máximo 10).

## 5.2.2 Base de datos que gestiona el agente Ag0.

### 5.2.2.1 Modelo Entidad-Interrelación de la base de datos.

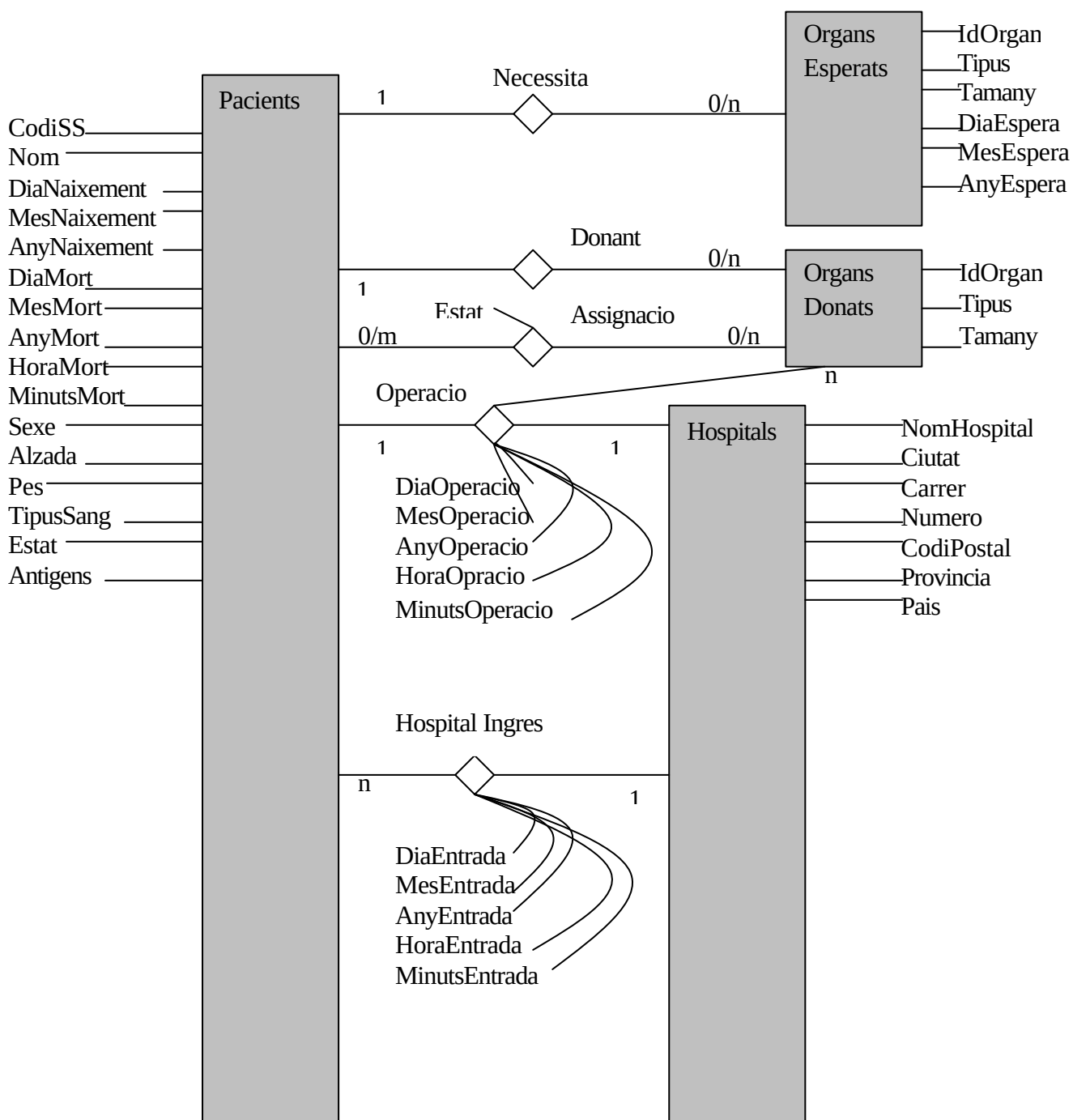


Figura 5.14 Modelo Entidad-Interrelación BD de Ag0.

### Observaciones:

- Sólo se introducen pacientes y hospitales, si al introducir un órgano, estos no existen en la base de datos. Siempre que se introduce un órgano, este lleva asociado un paciente (donante ó receptor) y un hospital de ingreso.
- Un paciente puede ser donante y receptor a la vez.
- Un paciente puede donar y recibir más de un órgano.
- A un paciente se le puede trasplantar más de un órgano del mismo tipo, en caso de que haya rechazo.
- Si en la entidad 'Pacients' el campo 'Estat' tiene el valor 'Muerto', no se modificará ningún dato asociado a ese paciente en la base de datos. De esta forma, en la base de datos figura el estado en el que estaba el paciente antes de fallecer. Esto servirá para hacer estadísticas.

### **5.2.2.2 Tablas utilizadas para implementar la base de datos.**

- Pacients ( **CodiSS**, Nom, DiaNaixement, MesNaixement, AnyNaixement, DiaMort, MesMort, AnyMort, Sexe, Alzada, Pes, TipusSang, Estat, Antigens).
- Hospitals (**NomHospital**, **Ciutat**, Carrer, Numero, CodiPostal, Provincia, Pais).
- HospitalIngres ( **CodiSS**, NomHospital, Ciutat, DiaEntrada, MesEntrada, AnyEntrada, HoraEntrada, MinutsEntrada).
- OrgansDonats (**IdOrgan**, Tipus, Tamany, CodiSSDonant (Clave extranjera Pacients));
- OrgansEsperats (**IdOrganEsperat**, Tipus, Tamany, CodiSSEsperat (Clave extranjera Pacients), DiaEspera, MesEspera, AnyEspera).
- Assignacio (**IdOrgan**, **CodiSS**, Estat).
- Operacio (**IdOrgan**, **CodiSS**, NomHospital, Ciutat, DiaOperacio, MesOperacio, AnyOperacio, HoraOperacio, MinutsOperacio).

### 5.2.2.3 ¿Cómo gestiona Ag0 la base de datos?.

#### 5.2.2.3.1 Gráfico que describe el comportamiento de Ago sobre la base de datos.

El gráfico muestra de una forma genérica, el comportamiento que tiene Ag0 sobre la base de datos, y se detalla en los siguientes puntos (5.2.2.3.X). También se recomienda volver a ojear la figura 5.2.

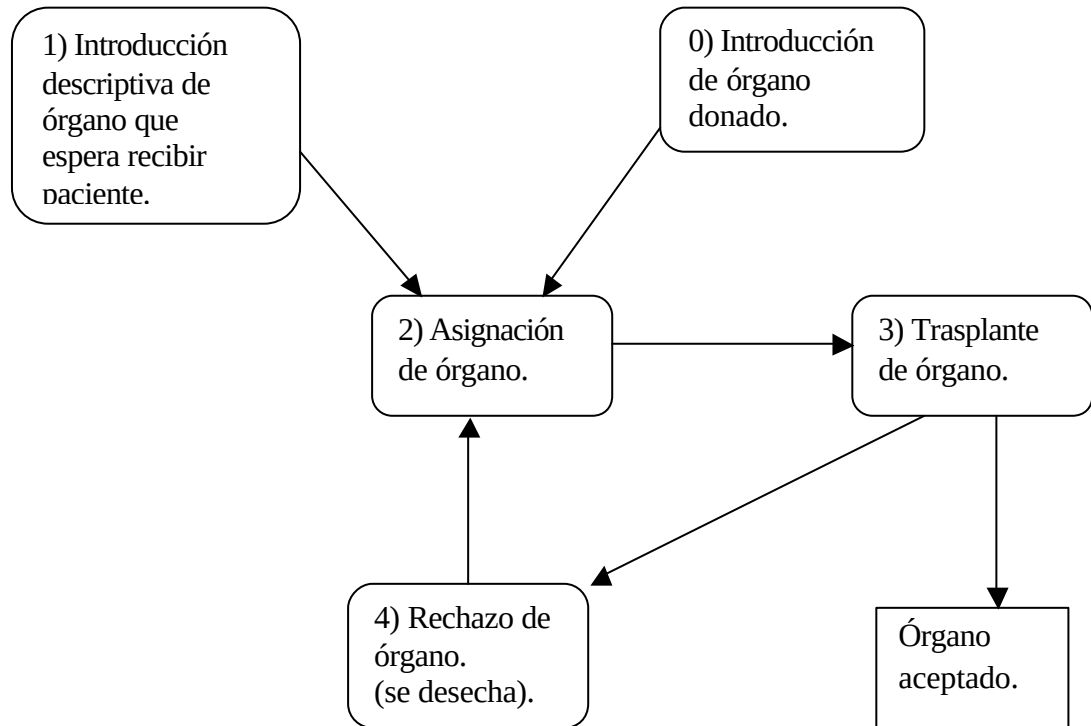


Figura 5.15 Proceso que siguen los órganos en el sistema.

#### 5.2.2.3.2 Respuesta Introducción Paciente:

##### --Si todo va bien:

Si se trata de un paciente donante, se introduce fila en tabla OrgansDonants.

Si se trata de un paciente receptor, se introduce fila en la tabla OrgansEsperats.

Si no existe paciente en BD, se introduce fila en tabla Pacients y en tabla HospitalIngres.

Si no existe el hospital donde se ingresa paciente, también se introduce fila en tabla Hospitals.

##### --Posibles inconvenientes:

-Órgano que se quiere introducir ya existe en BD.

-En caso de paciente receptor, puede ser que se intente volver a solicitar un mismo tipo de órgano por segunda vez.

-En caso de paciente receptor, puede ser que se le intente solicitar órgano, cuando este ya ha fallecido.

#### **5.2.2.3.3 Respuesta Anotación Muerte Paciente:**

##### **--Si todo va bien:**

Se actualiza fila correspondiente al paciente de la tabla Pacients. Concretamente se cambia el estado a “Muerto” y se introduce fecha y hora de muerte.

##### **--Posibles inconvenientes:**

- Paciente no existe en BD.
- Paciente ya consta como que ha fallecido en BD.
- El hospital no este autorizado a solicitar acción. (Sólo está autorizado el hospital en el que está ingresado el paciente).

#### **5.2.2.3.4 Respuesta Asignación Órgano:**

##### **--Si todo va bien:**

Dos casos:

-A paciente se le reasigna órgano después de haber rechazado otro.(No hace falta solicitar de nuevo un órgano del mismo tipo):  
Se inserta fila en tupla Assignacio con nueva relación, y estado “Asignado”.

-Paciente espera el mismo tipo de órgano y no ha rechazado ninguno con anterioridad:  
Insertamos fila en tupla Assignacio con nueva relación, y estado “Asignado”.

##### **--Posibles inconvenientes:**

- Paciente al que se quiere asignar órgano ha fallecido.
- No existe órgano en base de datos que se le quiere asignar al paciente.
- Paciente no espera recibir ese tipo de órgano.
- El órgano está siendo utilizado por otro paciente.
- El paciente no está ingresado en el hospital que solicita la acción; y por lo tanto, el hospital no está autorizado y no se realiza acción.
- Se quiere asignar órgano a paciente que no consta en base de datos.
- El paciente ya tiene órgano asignado.
- El paciente ya tiene trasplantado un órgano de ese tipo.

#### **5.2.2.3.5 Respuesta Operación Realizada:**

##### **--Si todo va bien:**

- Se cambia el estado de “Órgano Asignado” a “Operado” en la tabla Assignacio.
- Se inserta fila en tabla Operació.

##### **--Posibles inconvenientes:**

- El órgano que se quiere trasplantar no existe.
- El paciente receptor del órgano ha fallecido.
- El órgano consta como asignado en BD, pero no se le puede relacionar con ningún paciente. [Error grave de consistencia de BD].
- El órgano ya ha sido trasplantado a un paciente.
- El hospital no esta autorizado a solicitar acción, porque paciente no está ingresado en ese hospital.
- No consta que el órgano haya sido asignado a ningún paciente que esté vivo.



#### **5.2.2.3.6 Respuesta Órgano Rechazado:**

**--Si todo va bien:**

Se modifica fila en tupla Assignacio, se pasa de estado “Operado” a “Rechazado”.

**--Posibles inconvenientes:**

- El órgano no existe.
- El órgano ya consta como rechazado en BD.
- El órgano no se le ha trasplantado a nadie.
- El paciente ha fallecido, y por lo tanto no se modifica ningún dato que esté relacionado con él. (Esto servirá para extraer estadísticas y conclusiones).
- El hospital no está autorizado a solicitar acción; porque el paciente en cuestión, no está ingresado en ese hospital.
- El órgano figura como trasplantado en BD, pero no se le puede relacionar con ningún paciente. (Error grave de consistencia de base de datos).

***TODAS LAS ACCIONES VISTAS HASTA AHORA PUEDEN RECHAZARSE POR NO ESTAR Ag0 REGISTRADO EN DF.***

#### **5.2.2.3.7 Respuesta Encontrar Pacientes Compatibles:**

**--Si todo va bien:**

Se muestra al usuario los pacientes compatibles con un determinado órgano y su grado de compatibilidad.

**--Posibles inconvenientes:**

- No se ha registrado en DF algún agente que se requiere para realizar acción.
- El algoritmo asociado a un determinado agente no está implementado.
- No se ha conseguido suficientes clases de compatibilidad con organo.
- Demasiados pacientes para poder realizar acción (Error al crear tablas por agente Especialista).
- Demasiados atributos a tener en cuenta para poder realizar acción (Error al crear tablas por agente Especialista).
- El hospital no está autorizado a solicitar acción, porque no está registrado en la base de datos que gestiona Ag0.

### 5.3 Descripción del SMA que se encarga de la ordenación de pacientes.

El sistema que se encarga de la ordenación de pacientes, lo componen los agentes Especialista, ClusDM, Dummy1, Dummy2 y Dummy3. (Son agentes Dummy, porque solo se ha implementado el esqueleto del agente, y se le puede añadir el algoritmo que permite realizar la ordenación de pacientes con facilidad).

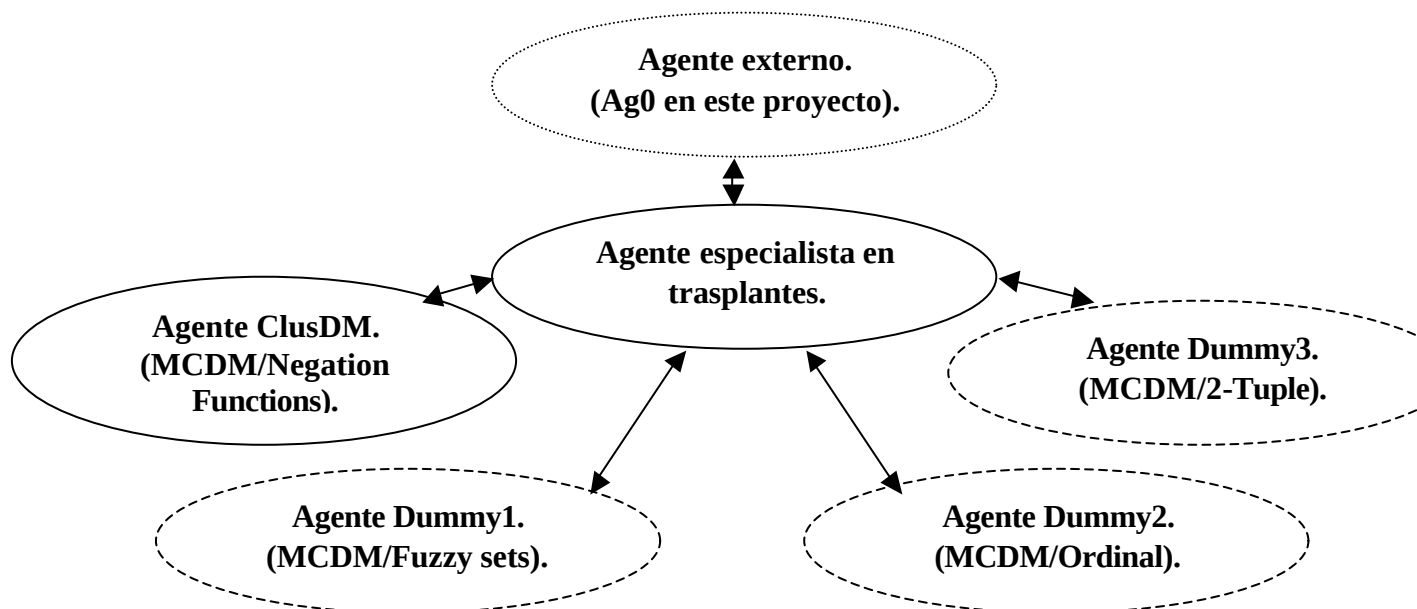


Figura 5.16 Estructura de sistema multi-agente encargado en la ordenación de pacientes.

Los agentes aplicarán métodos de decisión multi-criterio (**MCDM**), para realizar la ordenación.

El agente especialista se encarga de filtrar y preparar los datos que recibe por mensaje de un agente externo, en este caso de Ag0; y si todo es correcto, enviará los datos necesarios a ClusDM para que realice la ordenación de pacientes según la compatibilidad con el donante. Se espera, que a los agentes que se encargan de la ordenación de pacientes, se les asociará las siguientes técnicas:

- ClusDM: Técnica de funciones de negación [Torra, 1994]. (Agente utilizado en este proyecto).
- Dummy1: Técnica Fuzzy sets [Zadeh, 1968].
- Dummy2: Técnica 2-Tuple [Herrera y Martinez, 2000].
- Dummy3: Técnica Ordinal [Yager, 1998; Godo y Torra, 2000].

Estas técnicas utilizan datos cualitativos al intentar calificar el grado de similitud entre los pacientes candidatos y el paciente donante.

La ordenación se hace sobre un conjunto de criterios cualitativos, porque le resulta más fácil al usuario expresarse así. Los criterios que se han tenido en cuenta en este sistema multi-agente (SMA) son:

**-Distancia:** Expresa la distancia que existe entre el órgano donado y el receptor.  
(Los posibles valores cualitativos, en este caso, son: país, zona, provincia, ciudad, y hospital).

**-Peso:** Expresa la diferencia de peso entre el donante y el receptor.  
(Los posibles valores cualitativos, en este caso, son: floja, factible, buena, e ideal).

**-Tamaño del órgano:** Expresa la diferencia entre el tamaño del órgano donado y el del receptor.  
(Los posibles valores cualitativos, en este caso, son: floja, factible, buena, e ideal).

**-Antígenos:** Expresa la diferencia que hay entre los antígenos del donante y los del receptor.  
(Los posibles valores cualitativos, en este caso, son: correcto, adecuado, y malo).

**-Edad:** Expresa la diferencia de edad entre el donante y el receptor.  
(Los posibles valores cualitativos, en este caso, son: iguales, mas\_de\_5, mas\_de\_8, mas\_de\_11, mas\_de\_14, mas\_de\_17, y mas\_de\_20).

**-Tiempo de espera:** Expresa el tiempo que lleva el paciente esperando el órgano donado.  
(Los posibles valores cualitativos, en este caso, son: muy\_largo, largo, normal, moderado, y reciente).

El agente especialista recibe del agente externo: el tipo de órgano que corresponda, la lista de pacientes candidatos (el primer lugar de esta lista lo ocupa el paciente donante), la función de distancia utilizada por ClusDM, los pesos asociados a los diferentes atributos, y el agente que se encargará de la ordenación.

Si todo va bien, especialista enviará a Ag0 una lista ordenada de los pacientes, con el grado de afinidad que tienen con el órgano.

La función de distancia es utilizada por el agente ClusDM al agrupar pacientes con características similares. Es en este proceso, cuando el agente tiene en cuenta los pesos que ha asociado el usuario a los criterios que mencionamos anteriormente (distancia, peso, tamaño del órgano, antígenos, edad, y tiempo de espera).

Las funciones de distancia son las siguientes [Valls, 1995]:

- Diferencias.
- M.C.D (Mean Carácter Difference).
- Manhattan o City Blocks.
- Canberra Metric.
- Taxonomic Distance.

*Creo que no es necesario profundizar más en el funcionamiento de estos agentes para entender su función en este proyecto. Quien esté interesado en profundizar en la técnica que realiza la ordenación de pacientes, ó en cualquier otra cosa relacionada con este sistema multi-agente, puede consultar el proyecto de David Sánchez [David Sánchez, 2001].*

## **6. Juego de pruebas.**

### **6.1 Errores controlados por la propia interfaz gráfica:**

Desde la interfaz gráfica se controla que:

- No se supere una longitud máxima en los campos de texto. Incluido los de identificadores como IdOrgan y CodiSS.
- Todos los campos estén rellenos.
- El formato de la fecha sea: dd/mm/aaaa.
- La fecha introducida exista.
- La fecha introducida no sea futura a la actual.
- No se superen unos límites establecidos para los valores numéricos.
- El formato de la hora sea HH:MM.
- La hora introducida exista.
- La suma de los pesos asociados a los atributos, para encontrar pacientes compatibles sumen 100%.
- Los tipos de datos introducidos sean coherentes con lo que se pide desde la interfaz. Por ejemplo:
  - \*La altura y peso del paciente sean numéricos. Pueden ser decimales, representando la coma decimal por un punto.
  - \*Los pesos asociados a los atributos para encontrar pacientes compatibles con un órgano deben de ser números enteros.
  - \*Los antígenos deben de ser cuatro cifras binarias. El orden de los valores asociados a los antígenos es: B0, B1, Dr0, y Dr1 .

## **6.2 Respuesta del sistema a acciones solicitadas por usuario:**

En este apartado se intenta demostrar que el sistema funciona correctamente y hacer una presentación más detallada de las figuras 5.2, 5.15 y lo descrito en el punto 5.2.2.3. Para conseguirlo, se introducirán los datos necesarios en la BD (inicialmente vacía), y se hará un seguimiento. En el punto 7, en un apartado se mostrará cómo ha quedado el contenido de la base de datos, cuando se han realizado todas las acciones de este juego de pruebas, exceptuando la de encontrar pacientes compatibles. Para mostrar la acción encontrar pacientes compatibles, se utilizarán otros datos para mostrar su funcionamiento, y también se mostrará la base de datos con la información utilizada, en otro apartado del punto 7.

### **6.2.1 Respuesta del sistema a acciones que involucran solo al AgentCTH y Ag0.**

Se escriben en **negrita** las solicitudes que aportan algo nuevo al juego de pruebas, y en  *cursiva* las que no lo hacen.

Se hace una descripción secuencial de cómo se han introducido los datos en este juego de pruebas.

**Solicitud 1:** Introducir paciente donante, que ha fallecido en el hospital San Juan de Dios, con código de seguridad social: 0000000001. Ha donado los siguientes órganos:

- Corazón con IdOrgan: 1000000001.
- Hígado con IdOrgan: 1000000002.
- Pulmón con IdOrgan: 1000000003.
- Riñón con IdOrgan: 1000000004.

Respuesta: Al introducir los datos hasta donar el corazón, el sistema informa que los datos introducidos no modifican a los anteriores. Esto pasa siempre, a excepción del estado en el que se encuentra el órgano (asignado, operado, ó rechazado) y el estado de un paciente (si está vivo ó ha fallecido). Para hacer *modificaciones* del resto de datos, se ha de solicitar a administrador de la base de datos.

Hasta este momento, se ha introducido registro en las tablas: Patients, OrgansDonats, Hospitals, y HospitalIngres.

Al introducir órganos: hígado, pulmón, y riñón, se insertan registros en tabla OrgansDonats; pero no en las tablas: Patients, Hospitals y HospitalIngres, porque ya se han introducido los datos con la solicitud anterior, y como se ha comentado antes, para hacer una modificación de estos datos, hay que dirigirse al administrador de la base de datos.

**Solicitud 2:** Introducir paciente con CodiSS: 0000000002, que es donante de un corazón con IdOrgan: 1000000002.

Respuesta: El sistema rechaza introducir cualquier dato, porque ya existe un órgano con ese identificador.

**Solicitud 3: Pedir un corazón para el paciente con CodiSS: 0000000001.**

Respuesta: El sistema rechaza la solicitud, porque el paciente ha fallecido.

**Solicitud 4: Introducir paciente receptor, en el hospital San Juan de Dios de Barcelona, con CodiSS: 0000000002, y espera recibir un corazón.**

Respuesta: Se introduce registros en tablas: Pacients, Hospitals, HospitalIngres, y OrgansEsperats.

**Solicitud 5: Introducir paciente receptor, en el hospital San Juan de Dios de Barcelona, con CodiSS: 0000000002, y espera recibir un corazón.**

Respuesta: El sistema rechaza la solicitud, porque ya consta en base de datos, que el paciente ha solicitado el órgano.

**Solicitud 6: Anotar muerte del paciente con CodiSS: 0000000001.**

Respuesta: El paciente ya consta como fallecido en la base de datos.

**Solicitud 7: Anotar muerte del paciente con CodiSS: 0000000002, desde un hospital diferente al que está registrado el paciente.**

Respuesta: El sistema rechaza la solicitud, porque se ha hecho desde un hospital que no está autorizado.

**Solicitud 8: Anotar muerte del paciente con CodiSS: 0000000002, desde hospital en donde está registrado el paciente.**

Respuesta: Se cambia el contenido del campo estat a 'Muerto' y se anota el momento del fallecimiento, en la tabla Pacients.

**Solicitud 9: Asignar órgano, con IdOrgan: 1000000001, a paciente con CodiSS: 0000000002.**

Respuesta: El sistema rechaza la solicitud, porque el paciente al que se le quiere asignar el órgano, ha fallecido.

*Se introduce paciente, que espera recibir un corazón, con CodiSS: 0000000003, y está ingresado en el hospital San Juan de Dios.*

**Solicitud 10: Asignar órgano, con IdOrgan: 1000000002, a paciente con CodiSS: 0000000003.**

Respuesta: El sistema rechaza la solicitud, porque paciente no espera recibir un órgano de ese tipo. Recordamos que el paciente solo ha solicitado un corazón, y el IdOrgan: 1000000002 corresponde a un hígado.

**Solicitud 11: Asignar órgano, con IdOrgan: 1000000001, a paciente con CodiSS: 0000000003. La solicitud se realiza desde un hospital, en donde el paciente no está ingresado.**

Respuesta: El sistema rechaza la solicitud, porque se ha realizado desde un hospital que no está autorizado.

**Solicitud 12: Asignar órgano, con IdOrgan: 1000000011 al paciente con CodiSS: 0000000003.**

Respuesta: El sistema rechaza la solicitud, porque no existe ningún órgano, con ese IdOrgan, registrado en la base de datos.

*Se introduce paciente con CodiSS: 0000000004, que espera recibir un corazón, y está ingresado en el hospital San Juan de Dios.*

**Solicitud 13: Asignar órgano, con IdOrgan: 1000000001, a paciente con CodiSS: 0000000004.**

Respuesta: Se introduce registro en la tabla Assignacio, y el campo estat tiene el contenido 'Asignado'.

**Solicitud 14: Asignar órgano, con IdOrgan: 1000000001, a paciente con CodiSS: 0000000003.**

Respuesta: El sistema rechaza la solicitud, porque el órgano no está disponible.

**Solicitud 15: Asignar órgano, con IdOrgan: 1000000001, a paciente con CodiSS: 0000000005.**

Respuesta: El sistema rechaza la solicitud, porque se quiere asignar el órgano a un paciente, que no está registrado en la base de datos.

*Se introduce un paciente, con CodiSS: 0000000005, que dona un corazón con IdOrgan: 5000000001.*

**Solicitud 16: Asignar órgano, con IdOrgan: 5000000001, a paciente con CodiSS: 0000000004.**

Respuesta: El sistema rechaza la solicitud, porque el paciente ya tiene un órgano de ese tipo asignado.

**Solicitud 17: Anotar, que se ha realizado operación, al paciente que tiene asignado el órgano con IdOrgan: 1000000001.**

Respuesta: Se modifica el contenido del campo estat a 'operado' en la tabla Assignacio.

**Solicitud 18: Asignar órgano, con IdOrgan: 5000000001, a paciente con CodiSS: 0000000004.**

Respuesta: El sistema rechaza la solicitud, porque al paciente se le ha trasplantado un órgano de ese tipo.

**Solicitud 19: Anotar operación realizada del órgano con IdOrgan: 0000000002.**

Respuesta: El sistema rechaza la solicitud, porque el órgano no está registrado en la base de datos.

**Solicitud 20: Anotar operación realizada del órgano con IdOrgan: 5000000001.**

Respuesta: El sistema rechaza la solicitud, porque el órgano no está asignado a ningún paciente que esté vivo.

**Solicitud 21: Anotar operación realizada del órgano con IdOrgan: 1000000001.**

Respuesta: El sistema rechaza la solicitud, porque el órgano ya ha sido trasplantado.

*Se introduce paciente, con CodiSS: 0000000006, que solicita un corazón.*

*Se asigna corazón, con IdOrgan: 5000000001, a paciente con CodiSS: 0000000006.*

**Solicitud 22: Anotar operación realizada del corazón con IdOrgan: 5000000001.  
La solicitud se realiza desde un hospital diferente al que está ingresado el paciente.**

Respuesta: La solicitud se ha rechazado, porque el hospital desde el que se ha realizado, no está autorizado.

*Se anota que el paciente, con CodiSS: 0000000006, ha fallecido.*

**Solicitud 23: Anotar operación realizada de corazón con IdOrgan: 5000000001.**



Respuesta: El sistema rechaza la solicitud, porque el paciente que tenía asignado el órgano, ha fallecido.

**Solicitud 24: Anotar rechazo del corazón con IdOrgan: 1000000001.**

**La petición se realiza desde un hospital, donde el paciente no está registrado.**

Respuesta: El sistema rechaza la solicitud, porque se ha realizado desde un hospital que no está autorizado.

**Solicitud 25: Anotar rechazo del órgano con IdOrgan: 1000000002.**

Respuesta: El sistema rechaza la solicitud, porque el órgano no se le ha trasplantado a ningún paciente.

**Solicitud 26: Anotar rechazo del órgano con IdOrgan: 1000000012.**

Respuesta: El sistema rechaza la petición, porque el órgano no está registrado en la base de datos.

*Se anota que el paciente, con CodiSS: 0000000004, que tiene trasplantado el corazón con IdOrgan: 1000000001, ha fallecido.*

**Solicitud 27: Anotar rechazo del órgano con IdOrgan: 1000000001.**

Respuesta: El sistema rechaza la petición, porque el paciente, que tiene trasplantado el órgano, no está vivo.

Si un paciente muere, entonces no se modifica ningún dato relativo a él. Esto permite hacer estadísticas y sacar conclusiones.

**Solicitud 28: Asignar corazón, con IdOrgan: 5000000001, a paciente con CodiSS: 0000000003.**

Respuesta: El sistema realiza la petición. Recordamos que este órgano ya se le asignó a otro paciente, pero como ha fallecido, se puede reasignar.

*Anotar operación realizada de corazón con IdOrgan: 5000000001.*

**Solicitud 29: Anotar rechazo del corazón con IdOrgan: 5000000001.**

Respuesta: Se modifica el contenido del campo estat a 'Rechazado', en la tabla Assignacio.

**Solicitud 30: Anotar rechazo del corazón con IdOrgan: 5000000001.**

Respuesta: El sistema rechaza la petición, porque ya consta en la base de datos que el órgano se ha rechazado.

*Se introduce paciente con CodiSS: 0000000007, que dona un corazón con IdOrgan: 7000000001.*

**Solicitud 31: Asignación de corazón, con IdOrgan: 7000000001, a paciente con CodiSS: 0000000003.**

Respuesta: El sistema realiza la asignación correctamente. No ha hecho falta volver a solicitar el órgano, después de que el paciente rechazara uno del mismo tipo. Se ha añadido registro a la tabla Assignacio, con el contenido ‘Órgano Asignado’ en el campo Estat.

También se controlan errores, que no se muestran en este juego de pruebas, como los producidos al acceder a la base de datos, y algunos de los producidos por inconsistencia en los datos de la base de datos.

### **6.2.2 Respuesta del sistema en búsqueda de pacientes compatibles.**

Partiendo de los datos mostrados en el punto 7.2, se buscan candidatos que sean compatibles con órgano, que tiene IdOrgan: 1000000000.

Primero, Ag0 buscará en la base de datos los pacientes que están vivos y esperan recibir un órgano del mismo tipo que el recibido por mensaje. (En este caso, el que tiene IdOrgan: 1000000000). Cuando el agente Especialista reciba los datos que le envía Ag0, los filtrará y preparará para que ClusDM realice finalmente la ordenación.

**Solicitud 1: Se realiza búsqueda con las siguientes opciones:**

**-Agente que realizará la ordenación utilizará el algoritmo Negation\_Functions, aplicando la función de distancia Diferencias.**

**-Se asocian los siguientes valores, a los pesos de los atributos, que se tienen en cuenta a la hora de hacer la ordenación:**

**Distancia: 12  
Antígenos: 12  
Peso: 12  
Edad: 12  
Tamaño del órgano: 12  
Tiempo de espera: 40**

**-La solicitud la realiza un hospital que no está registrado en la base de datos.**

Respuesta: El sistema rechaza la solicitud, porque se ha realizado desde un hospital que no está autorizado.

**Solicitud 2: Se realiza búsqueda con las siguientes opciones:**

**-Agente que realizará la ordenación utilizará el algoritmo Negation\_Functions, aplicando la función de distancia Diferencias.**

**-Se asocian los siguientes valores, a los pesos de los atributos, que se tienen en cuenta a la hora de hacer la ordenación:**

**Distancia: 12**  
**Antígenos: 12**  
**Peso: 12**  
**Edad: 12**  
**Tamaño del órgano: 12**  
**Tiempo de espera: 40**

**Respuesta:** El sistema muestra el siguiente listado:

*Numero total de pacientes compatibles con organo: 15*

*Codigo Seguridad Social de Paciente compatible: 0000000009*  
*Con grado de compatibilidad: optimo*

*Codigo Seguridad Social de Paciente compatible: 0000000005*  
*Con grado de compatibilidad: bueno*

*Codigo Seguridad Social de Paciente compatible: 0000000017*  
*Con grado de compatibilidad: bueno*

*Codigo Seguridad Social de Paciente compatible: 0000000003*  
*Con grado de compatibilidad: bueno*

*Codigo Seguridad Social de Paciente compatible: 0000000018*  
*Con grado de compatibilidad: adecuado*

*Codigo Seguridad Social de Paciente compatible: 0000000006*  
*Con grado de compatibilidad: adecuado*

*Codigo Seguridad Social de Paciente compatible: 0000000001*  
*Con grado de compatibilidad: adecuado*

*Codigo Seguridad Social de Paciente compatible: 0000000016*  
*Con grado de compatibilidad: adecuado*

*Codigo Seguridad Social de Paciente compatible: 0000000020*  
*Con grado de compatibilidad: adecuado*

*Codigo Seguridad Social de Paciente compatible: 0000000002*  
*Con grado de compatibilidad: adecuado*

*Codigo Seguridad Social de Paciente compatible: 0000000008*  
*Con grado de compatibilidad: factible*

*Codigo Seguridad Social de Paciente compatible: 0000000004*  
*Con grado de compatibilidad: factible*

*Codigo Seguridad Social de Paciente compatible: 0000000007*  
*Con grado de compatibilidad: desfavorable*

*Codigo Seguridad Social de Paciente compatible: 0000000010*  
*Con grado de compatibilidad: malo*

*Codigo Seguridad Social de Paciente compatible: 0000000019*  
*Con grado de compatibilidad: pesimo*

Para saber, de una forma más precisa, como se comporta el sistema al ordenar los pacientes, se puede consultar el juego de pruebas de la documentación del proyecto de David Sánchez 2001. En este proyecto, lo único que se ha hecho es comunicar Ag0 con agente especialista.

**Solicitud 3: Se realiza búsqueda con las siguientes opciones:**

**-Agente que realizará la ordenación utilizará el algoritmo Two\_Tuple, aplicando cualquier función de distancia. (El sistema responde igual para los algoritmos: Fuzzy\_sets y Ordinal).**

**-Se asocian los siguientes valores, a los pesos de los atributos, que se tienen en cuenta a la hora de hacer la ordenación:**

**Distancia: 12**  
**Antígenos: 12**  
**Peso: 12**  
**Edad: 12**  
**Tamaño del órgano: 12**  
**Tiempo de espera: 40**

**Respuesta:** El sistema rechaza la solicitud, porque el agente, que ha de realizar la acción, no está registrado en el sistema. (El agente no se ha cargado previamente en la plataforma).

**Solicitud 4:** Se realiza búsqueda con las siguientes opciones:

-Agente que realizará la ordenación utilizará el algoritmo Two\_Tuple, aplicando cualquier función de distancia. (El sistema responde igual para los algoritmos: Fuzzy\_sets y Ordinal).

-Se asocian los siguientes valores, a los pesos de los atributos, que se tienen en cuenta a la hora de hacer la ordenación:

Distancia: 12  
Antígenos: 12  
Peso: 12  
Edad: 12  
Tamaño del órgano: 12  
Tiempo de espera: 40

Respuesta: Se produce un error, controlado por el sistema multi-agente, indicando que el algoritmo no está implementado.

**Solicitud 5:** Se realiza búsqueda con las siguientes opciones:

-Agente que realizará la ordenación utilizará el algoritmo Negation\_Functions, aplicando la función de distancia Diferencias.

-Se asocian los siguientes valores, a los pesos de los atributos, que se tienen en cuenta a la hora de hacer la ordenación:

Distancia: 100  
Antígenos: 0  
Peso: 0  
Edad: 0  
Tamaño del órgano: 0  
Tiempo de espera: 0

Respuesta: Se produce error, controlado por el sistema multi-agente, porque no se han conseguido suficientes clases (grupos de pacientes), en el proceso de ordenación.

***También se controlan los errores producidos por ClusDM al no poder:***

***-Tener en cuenta tantos pacientes.***

***-Tener en cuenta tantos atributos.***

***Los pesos no validos se controlan de la misma forma que para los dos anteriores, pero también son controlados a nivel de interfaz gráfica en AgentCTH.***

## **7. Contenido de la base de datos.**

En el punto 7.1 se muestra el contenido de la base de datos al realizar todos los pasos del punto 6.2.1 (Proceso que se inicia con la base de datos está vacía).

En el punto 7.2 se muestra el contenido de la base de datos al realizar todos los pasos del punto 6.2.2. En este caso, solo se realizan consultas sobre la base de datos.

### **7.1 Caso 1: Contenido de la base de datos.**

#### **Assignacio**

***IdOrgan CodigoSS Estat.***

|            |            |                 |
|------------|------------|-----------------|
| 1000000001 | 0000000004 | Operado         |
| 5000000001 | 0000000003 | Rechazado       |
| 5000000001 | 0000000006 | Organo Asignado |
| 7000000001 | 0000000003 | Organo Asignado |

#### **HospitalIngres**

***CodiSS. NomHospital.***

0000000001 San Juan de Dios

***Ciutat.***

Barcelona

***DiaEntrada MesEntrada AnyEntrada HoraEntrada MinutsEntrada.***

|    |   |      |    |   |
|----|---|------|----|---|
| 10 | 1 | 2003 | 17 | 6 |
|----|---|------|----|---|

***CodiSS. NomHospital.***

0000000002 San Juan de Dios

***Ciutat.***

Barcelona

***DiaEntrada MesEntrada AnyEntrada HoraEntrada MinutsEntrada.***

|    |   |      |    |   |
|----|---|------|----|---|
| 10 | 1 | 2003 | 18 | 7 |
|----|---|------|----|---|

***CodiSS. NomHospital.***

0000000003 San Juan de Dios

***Ciutat.***

Barcelona

***DiaEntrada MesEntrada AnyEntrada HoraEntrada MinutsEntrada.***

|    |   |      |    |    |
|----|---|------|----|----|
| 10 | 1 | 2003 | 18 | 40 |
|----|---|------|----|----|

***CodiSS. NomHospital.***

0000000004 San Juan de Dios

***Ciutat.***

Barcelona

***DiaEntrada MesEntrada AnyEntrada HoraEntrada MinutsEntrada.***

|    |   |      |    |   |
|----|---|------|----|---|
| 10 | 1 | 2003 | 20 | 5 |
|----|---|------|----|---|

***CodiSS. NomHospital.***

0000000005 San Juan de Dios

***Ciutat.***

Barcelona

***DiaEntrada MesEntrada AnyEntrada HoraEntrada MinutsEntrada.***

|    |   |      |    |    |
|----|---|------|----|----|
| 10 | 1 | 2003 | 22 | 26 |
|----|---|------|----|----|

***CodiSS. NomHospital.***

0000000006 San Juan de Dios

***Ciutat.***

Barcelona

***DiaEntrada MesEntrada AnyEntrada HoraEntrada MinutsEntrada.***

|    |   |      |    |   |
|----|---|------|----|---|
| 10 | 1 | 2003 | 23 | 1 |
|----|---|------|----|---|

**CodiSS. NomHospital.**

0000000007 San Juan de Dios

**Ciutat.**

Barcelona

**DiaEntrada MesEntrada AnyEntrada HoraEntrada MinutsEntrada.**

11

1

2003

0

23

### Hospitals

**NomHospital**

**Ciutat.**

**Carrer.**

**Nu Codi Provincia.**

**Pais.**

San Juan de Dios

Barcelona

Ronda de Dalt

20 08080 Barcelona

Espanya

### Operacio

**CodiSS. IdOrgan. NomHospital.**

0000000003 5000000001 San Juan de Dios

**Ciutat.**

Barcelona

**DiaOperacio. MesOperacio. AnyOperacio HoraOperacio. MinutsOperacio.**

10

1

2003

0

12

**CodiSS. IdOrgan. NomHospital.**

0000000004 1000000001 San Juan de Dios

**Ciutat.**

Barcelona

**DiaOperacio. MesOperacio. AnyOperacio HoraOperacio. MinutsOperacio.**

30

8

1999

13

20

### OrgansDonats

**IdOrgan. Tipus.**

**Tamany.**

**CodiSSDonant**

1000000001 Corazon

40

0000000001

1000000002 Higado

45

0000000001

1000000003 Pulmon

60

0000000001

1000000004 Rinyon

30

0000000001

5000000001 Corazon

35

0000000005

7000000001 Corazon

27

0000000007

### OrgansEsperats

**CodiSSEspe**

**Clave interna. Tipus.**

**Tamany.**

**DiaEspe MesEspe AnyEspe**

0000000002

1

Corazon

25

10

1

2003

0000000003

2

Corazon

30

10

1

2003

0000000004

3

Corazon

32

10

1

2003

0000000006

4

Corazon

35

10

1

2003

# Pacients

\* **CodiSS**

0000000001

**Nom**

Diego Pino Lopez

|                     |                     |                    |                |                |                |                 |
|---------------------|---------------------|--------------------|----------------|----------------|----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b> | <b>HoraMort</b> |
| 10                  | 7                   | 1950               | 10             | 2              | 1990           | 14              |

|                  |             |               |            |                 |              |
|------------------|-------------|---------------|------------|-----------------|--------------|
| <b>MinutsMor</b> | <b>Sexe</b> | <b>Alzada</b> | <b>Pes</b> | <b>TipusSan</b> | <b>Estat</b> |
| 53               | Hombre      | 175           | 70         | O               | Muerto       |

**Antigens**

1001

\* **CodiSS**

0000000002

**Nom**

Paco Pino Lopez

|                     |                     |                    |                |                |                |                 |
|---------------------|---------------------|--------------------|----------------|----------------|----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b> | <b>HoraMort</b> |
| 1                   | 12                  | 1956               | 1              | 1              | 1991           | 13              |

|                  |             |               |            |                 |              |
|------------------|-------------|---------------|------------|-----------------|--------------|
| <b>MinutsMor</b> | <b>Sexe</b> | <b>Alzada</b> | <b>Pes</b> | <b>TipusSan</b> | <b>Estat</b> |
| 23               | Hombre      | 178           | 70         | O               | Muerto       |

**Antigens**

1001

\* **CodiSS**

0000000003

**Nom**

Francisco Javier Macias Garcia

|                     |                     |                    |                |                |                |                 |
|---------------------|---------------------|--------------------|----------------|----------------|----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b> | <b>HoraMort</b> |
| 23                  | 6                   | 1975               |                |                |                |                 |

|                  |             |               |            |                 |              |
|------------------|-------------|---------------|------------|-----------------|--------------|
| <b>MinutsMor</b> | <b>Sexe</b> | <b>Alzada</b> | <b>Pes</b> | <b>TipusSan</b> | <b>Estat</b> |
|                  | Hombre      | 178           | 75         | O               | Vivo         |

**Antigens**

0110



\* **CodiSS**

0000000004

**Nom**

Cristobal Ledesma Martinez

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 17                  | 1                   | 1949               | 10             | 2              | 1992            | 12              |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
| 34                  | Hombre              | 172                | 70             |                | O               | Muerto          |

**Antigens**

1010

**CodiSS**

0000000005

**Nom**

Oscar Hinojosa Esquivel

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 6                   | 7                   | 1975               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 179                | 75             |                | O               | Vivo            |

**Antigens**

0101

**CodiSS**

0000000006

**Nom**

Marta Rio Freijo

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 20                  | 1                   | 1975               | 10             | 2              | 1982            | 15              |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
| 43                  | Mujer               | 165                | 55             |                | O               | Muerto          |

**Antigens**

0001

\* **CodiSS**

0000000007

**Nom**

Francisca Olias Guisado

**DiaNaixement** **MesNaixement** **AnyNaixemen** **DiaMort** **MesMort** **AnyMort** **HoraMort**

4

4

1945

**MinutsMor**

**Sexe**

**Alzada**

**Pes**

**TipusSan**

**Estat**

Mujer

160

58

O

Vivo

**Antigens**

1100

## **7.2 Caso 2: Contenido de la base de datos.**

### **Assignacio**

#### ***IdOrgan. CodiSS Estat***

2000000000 0000000023 Rechazado  
3000000000 0000000023 Organo Asignado  
3000000000 0000000026 Rechazado  
4000000000 0000000030 Organo Asignado  
5000000000 0000000027 Operado

### **HospitalIngres**

#### ***CodiSS NomHospital***

0000000001 Clinica Universitaria de Navarra

#### ***Ciutat***

Pamplona

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

4 11 1999 23 29

#### ***CodiSS NomHospital***

0000000002 Hospital Vall Hebron

#### ***Ciutat***

Barcelona

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

12 5 2001 11 55

#### ***CodiSS NomHospital***

0000000003 Hospital Ramon y Cajal

#### ***Ciutat***

Madrid

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

9 8 2000 0 8

#### ***CodiSS NomHospital***

0000000004 Hospital de Sagunto

#### ***Ciutat***

Valencia

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

12 5 2001 0 20

#### ***CodiSS NomHospital***

0000000005 Hospital Universitario de Valme

#### ***Ciutat***

Sevilla

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

31 12 1999 0 30

#### ***CodiSS NomHospital***

0000000006 Hospital de Cabuenyes

#### ***Ciutat***

Gijon

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

1 1 2001 0 42

#### ***CodiSS NomHospital***

0000000007 Hospital de Laredo

#### ***Ciutat***

Laredo

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

1 12 2000 0 47

#### ***CodiSS NomHospital***

0000000008 Hospital Universitario de la Fe

#### ***Ciutat***

Valencia

#### ***DiaEntrad MesEntrad AnyEntrad HoraEntrad MinutsEntrada***

12 5 2001 0 51

|                   |                      |                  |
|-------------------|----------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>   | <b>Ciutat</b>    |
| 000000009         | Hospital Joan XXIII  | Tarragona        |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>     | <b>AnyEntrad</b> |
| 10                | 12                   | 1999             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b> |                  |
| 0                 | 53                   |                  |

|                   |                      |                  |
|-------------------|----------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>   | <b>Ciutat</b>    |
| 000000010         | Hospital Sierrallana | Torrelavega      |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>     | <b>AnyEntrad</b> |
| 12                | 5                    | 2000             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b> |                  |
| 0                 | 57                   |                  |

|                   |                               |                  |
|-------------------|-------------------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>            | <b>Ciutat</b>    |
| 000000011         | Hospital Universitario La Paz | Madrid           |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>              | <b>AnyEntrad</b> |
| 17                | 9                             | 2000             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b>          |                  |
| 1                 | 0                             |                  |

|                   |                                    |                  |
|-------------------|------------------------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>                 | <b>Ciutat</b>    |
| 000000012         | Hospital Universitario de Canarias | La Laguna        |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>                   | <b>AnyEntrad</b> |
| 13                | 7                                  | 2001             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b>               |                  |
| 13                | 15                                 |                  |

|                   |                                |                  |
|-------------------|--------------------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>             | <b>Ciutat</b>    |
| 000000013         | Hospital Marques de Valdecilla | Santander        |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>               | <b>AnyEntrad</b> |
| 19                | 5                              | 2000             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b>           |                  |
| 13                | 18                             |                  |

|                   |                         |                  |
|-------------------|-------------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>      | <b>Ciutat</b>    |
| 000000014         | Hospital Obispo Polanco | Teruel           |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>        | <b>AnyEntrad</b> |
| 22                | 4                       | 2001             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b>    |                  |
| 13                | 21                      |                  |

|                   |                         |                  |
|-------------------|-------------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>      | <b>Ciutat</b>    |
| 000000016         | Hospital General de Vic | Vic              |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>        | <b>AnyEntrad</b> |
| 11                | 1                       | 2001             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b>    |                  |
| 14                | 2                       |                  |

|                   |                                   |                   |
|-------------------|-----------------------------------|-------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>                | <b>Ciutat</b>     |
| 000000017         | Hospital Universitario son Dureta | Palma de Mallorca |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>                  | <b>AnyEntrad</b>  |
| 14                | 5                                 | 2000              |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b>              |                   |
| 14                | 4                                 |                   |

|                   |                        |                  |
|-------------------|------------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>     | <b>Ciutat</b>    |
| 000000018         | Hospital San Sebastian | Ecija            |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>       | <b>AnyEntrad</b> |
| 1                 | 7                      | 2001             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b>   |                  |
| 14                | 8                      |                  |

|                   |                      |                  |
|-------------------|----------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>   | <b>Ciutat</b>    |
| 000000019         | Hospital San Agustin | Aviles           |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>     | <b>AnyEntrad</b> |
| 15                | 8                    | 2001             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b> |                  |
| 14                | 10                   |                  |

|                   |                      |                  |
|-------------------|----------------------|------------------|
| <b>CodiSS</b>     | <b>NomHospital</b>   | <b>Ciutat</b>    |
| 000000020         | Hospital Vall Hebron | Barcelona        |
| <b>DiaEntrad</b>  | <b>MesEntrad</b>     | <b>AnyEntrad</b> |
| 12                | 4                    | 2001             |
| <b>HoraEntrad</b> | <b>MinutsEntrada</b> |                  |
| 14                | 13                   |                  |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 0000000021       | Hospital de prueba |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 6                | 12                 | 2002             | 19                | 22                   |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 0000000022       | Hospital de prueba |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 7                | 12                 | 2002             | 13                | 31                   |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 0000000023       | Nombre de prueba   |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 7                | 12                 | 2002             | 14                | 7                    |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 0000000024       | Hospital de prueba |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 7                | 12                 | 2002             | 14                | 35                   |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 0000000025       | Hospital de prueba |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 7                | 12                 | 2002             | 14                | 38                   |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 0000000026       | Hospital de prueba |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 9                | 12                 | 2002             | 13                | 57                   |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 0000000027       | Hospital de prueba |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 11               | 12                 | 2002             | 15                | 1                    |

|                  |                     |                  |                   |                      |
|------------------|---------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b>  |                  |                   | <b>Ciutat</b>        |
| 0000000030       | Hospital de prueba3 |                  |                   | Barcelona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>    | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 14               | 12                  | 2002             | 15                | 24                   |

|                  |                    |                  |                   |                      |
|------------------|--------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b> |                  |                   | <b>Ciutat</b>        |
| 00008            | Hospital de Cruces |                  |                   | Baracaldo            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>   | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 30               | 6                  | 1999             | 13                | 56                   |

|                  |                     |                  |                   |                      |
|------------------|---------------------|------------------|-------------------|----------------------|
| <b>CodiSS</b>    | <b>NomHospital</b>  |                  |                   | <b>Ciutat</b>        |
| 1000000000       | Hospital Joan XXIII |                  |                   | Tarragona            |
| <b>DiaEntrad</b> | <b>MesEntrad</b>    | <b>AnyEntrad</b> | <b>HoraEntrad</b> | <b>MinutsEntrada</b> |
| 10               | 10                  | 1960             | 23                | 11                   |

## Hospitals

| <i>NomHospital</i>    | <i>Ciutat</i>     | <i>Carrer</i>    | <i>Nu</i> | <i>Codi</i> | <i>Provincia</i> | <i>Pais</i> |
|-----------------------|-------------------|------------------|-----------|-------------|------------------|-------------|
| Clinica Universitaria | Pamplona          | Joan Marti       | 12        | 00900       | Navarra          | Espanya     |
| Hospital de           | Gijon             | Florida          | 40        | 00400       | Asturias         | Espanya     |
| Hospital de Cruces    | Baracaldo         | La Mina          | 20        | 00008       | Vizcaya          | Espanya     |
| Hospital de Laredo    | Laredo            | Lorca            | 50        | 00001       | Cantabria        | Espanya     |
| Hospital de prueba    | Barcelona         | La mina          | 12        | 08080       | Barcelona        | Espanya     |
| Hospital de prueba3   | Barcelona         | Valencia         | 15        | 08080       | Barcelona        | Espanya     |
| Hospital de           | Valencia          | Gessami          | 14        | 00100       | Valencia         | Espanya     |
| Hospital General de   | Vic               | Jacint Verdaguer | 45        | 00010       | Barcelona        | Espanya     |
| Hospital Joan XXIII   | Tarragona         | Rafael Alberti   | 12        | 00800       | Tarragona        | Espanya     |
| Hospital Marques      | Santander         | Pau Claris       | 15        | 00007       | Cantabria        | Espanya     |
| Hospital Obispo       | Teruel            | La Plana         | 45        | 00008       | Teruel           | Espanya     |
| Hospital Ramon y      | Madrid            | Avda. de Madrid  | 12        | 08800       | Madrid           | Espanya     |
| Hospital San          | Aviles            | Roda de Vara     | 53        | 00015       | Asturias         | Espanya     |
| Hospital San          | Ecija             | Victoria         | 12        | 00012       | Sevilla          | Espanya     |
| Hospital Sierrallana  | Torrelavega       | Eusebio Guell    | 114       | 00004       | Cantabria        | Espanya     |
| Hospital              | La Laguna         | La Palma         | 15        | 00006       | Tenerife         | Espanya     |
| Hospital              | Valencia          | La Toja          | 72        | 00002       | Valencia         | Espanya     |
| Hospital              | Sevilla           | Urquinaona       | 10        | 00200       | Sevilla          | Espanya     |
| Hospital              | Madrid            | Torre vieja      | 90        | 00005       | Madrid           | Espanya     |
| Hospital              | Palma de Mallorca | Provenza         | 180       | 00011       | Baleares         | Espanya     |
| Hospital Vall         | Barcelona         | Ronda de Dalt    | 12        | 08080       | Barcelona        | Espanya     |
| Nombre de prueba      | Barcelona         | Gessamí          | 14        | 08880       | Barcelona        | Espanya     |

## Operacio

**CodiSS**   **IdOrgan**   **NomHospital**   **Ciutat**  
0000000023   2000000000   Hospital de prueba   Barcelona

**DiaOperacio**   **MesOperacio**   **AnyOperacio**   **HoraOperacio**   **MinutsOperacio**  
12   12   2000   12   0

**CodiSS**   **IdOrgan**   **NomHospital**   **Ciutat**  
0000000026   3000000000   Hospital de prueba   Barcelona

**DiaOperacio**   **MesOperacio**   **AnyOperacio**   **HoraOperacio**   **MinutsOperacio**  
12   12   2000   12   25

**CodiSS**   **IdOrgan**   **NomHospital**   **Ciutat**  
0000000027   5000000000   Hospital de prueba   Barcelona

**DiaOperacio**   **MesOperacio**   **AnyOperacio**   **HoraOperacio**   **MinutsOperacio**  
12   12   2000   12   25

### OrgansDonats

| <i><b>IdOrgan</b></i> | <i><b>Tipus</b></i> | <i><b>Tamany</b></i> | <i><b>CodiSSDonan</b></i> |
|-----------------------|---------------------|----------------------|---------------------------|
| 0000000030            | Pulmon              | 54,2                 | 0000000030                |
| 1000000000            | Corazon             | 34,2                 | 1000000000                |
| 2000000000            | Corazon             | 40                   | 0000000021                |
| 3000000000            | Corazon             | 12                   | 0000000024                |
| 4000000000            | Corazon             | 12                   | 0000000025                |
| 5000000000            | Corazon             | 35                   | 0000000026                |

### OrgansEsperats

| <i><b>CodiSSEspe</b></i> | <i><b>Clave interna. Tipus</b></i> | <i><b>Tamany</b></i> | <i><b>DiaEsper</b></i> | <i><b>MesEspe</b></i> | <i><b>AnyEsp</b></i> |
|--------------------------|------------------------------------|----------------------|------------------------|-----------------------|----------------------|
| 0000000001               | 3 Corazon                          | 30                   | 4                      | 11                    | 1999                 |
| 0000000002               | 4 Corazon                          | 31,2                 | 12                     | 5                     | 2001                 |
| 0000000003               | 5 Corazon                          | 39,64                | 9                      | 8                     | 2000                 |
| 0000000004               | 6 Corazon                          | 37,1                 | 12                     | 5                     | 2001                 |
| 0000000005               | 7 Corazon                          | 40,3                 | 31                     | 12                    | 1999                 |
| 0000000006               | 8 Corazon                          | 39,1                 | 1                      | 1                     | 2001                 |
| 0000000007               | 9 Corazon                          | 28,3                 | 1                      | 12                    | 2000                 |
| 0000000008               | 10 Corazon                         | 37                   | 12                     | 5                     | 2001                 |
| 0000000009               | 11 Corazon                         | 35,4                 | 10                     | 12                    | 1999                 |
| 0000000010               | 12 Corazon                         | 29,2                 | 12                     | 5                     | 2000                 |
| 0000000011               | 13 Corazon                         | 30                   | 17                     | 9                     | 2000                 |
| 0000000012               | 14 Corazon                         | 40,2                 | 13                     | 7                     | 2001                 |
| 0000000013               | 15 Corazon                         | 50                   | 19                     | 5                     | 2000                 |
| 0000000014               | 16 Corazon                         | 37,1                 | 22                     | 4                     | 2001                 |
| 0000000016               | 18 Corazon                         | 36,1                 | 11                     | 1                     | 2001                 |
| 0000000017               | 19 Corazon                         | 33,3                 | 14                     | 5                     | 2000                 |
| 0000000018               | 20 Corazon                         | 37                   | 1                      | 7                     | 2001                 |
| 0000000019               | 21 Corazon                         | 43,4                 | 15                     | 8                     | 2001                 |
| 0000000020               | 22 Corazon                         | 39,2                 | 12                     | 4                     | 2001                 |
| 0000000021               | 23 Higado                          | 12                   | 6                      | 12                    | 2002                 |
| 0000000021               | 24 Higado                          | 12                   | 6                      | 12                    | 2002                 |
| 0000000022               | 25 Rinyon                          | 40                   | 7                      | 12                    | 2002                 |
| 0000000023               | 26 Higado                          | 30                   | 7                      | 12                    | 2002                 |
| 0000000026               | 30 Corazon                         | 41                   | 14                     | 12                    | 2002                 |
| 0000000027               | 27 Corazon                         | 35,7                 | 11                     | 12                    | 2002                 |
| 0000000030               | 28 Corazon                         | 38                   | 14                     | 12                    | 2002                 |
| 0000000030               | 29 Higado                          | 38                   | 14                     | 12                    | 2002                 |
| 00008                    | 17 Corazon                         | 35,3                 | 30                     | 6                     | 1999                 |

## Pacients

### \* **CodiSS**

0000000001

### **Nom**

Joaquin Garcia Lopez

**DiaNaixement** **MesNaixement** **AnyNaixemen** **DiaMort** **MesMort** **AnyMort** **HoraMort**  
1 1 1956

**MinutsMor** **Sexe** **Alzada** **Pes** **TipusSan** **Estat**  
Hombre 170 60,8 A Vivo

### **Antigens**

1110

### \* **CodiSS**

0000000002

### **Nom**

Juan Ramon Jimenez Garcia

**DiaNaixement** **MesNaixement** **AnyNaixemen** **DiaMort** **MesMort** **AnyMort** **HoraMort**  
2 1 1970

**MinutsMor** **Sexe** **Alzada** **Pes** **TipusSan** **Estat**  
Hombre 170 65,8 AB Vivo

### **Antigens**

1111

### \* **CodiSS**

0000000003

### **Nom**

Fco. Javier Macias Garcia

**DiaNaixement** **MesNaixement** **AnyNaixemen** **DiaMort** **MesMort** **AnyMort** **HoraMort**  
4 7 1977

**MinutsMor** **Sexe** **Alzada** **Pes** **TipusSan** **Estat**  
Hombre 175 75 A Vivo

### **Antigens**

0010

### \* **CodiSS**

0000000004

### **Nom**

Pedro Duran Montoro

**DiaNaixement** **MesNaixement** **AnyNaixemen** **DiaMort** **MesMort** **AnyMort** **HoraMort**  
1 2 1945

**MinutsMor** **Sexe** **Alzada** **Pes** **TipusSan** **Estat**  
Hombre 182 65,9 AB Vivo

### **Antigens**

1110



\* **CodiSS**  
0000000005

**Nom**  
Javier Mellado Mellado

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 23                      | 9                   | 1950               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Hombre              | 175                | 77,2           |                | A               | Vivo            |
| <b>Antigens</b><br>0010 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000006

**Nom**  
Lorena Fernandez Moya

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 4                       | 8                   | 1965               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Mujer               | 182                | 80             |                | AB              | Vivo            |
| <b>Antigens</b><br>1111 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000007

**Nom**  
Javier Davila Quesada

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 1                       | 1                   | 1976               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Hombre              | 180                | 78             |                | A               | Vivo            |
| <b>Antigens</b><br>1010 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000008

**Nom**  
Sergio Alvarez Lora

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 2                       | 1                   | 1975               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Hombre              | 175                | 65             |                | AB              | Vivo            |
| <b>Antigens</b><br>1011 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000009

**Nom**  
Dolores Fernandez Campos

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 9                       | 9                   | 1961               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Mujer               | 175                | 70             |                | A               | Vivo            |
| <b>Antigens</b><br>0010 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000010

**Nom**  
Javier Cortes Losada

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 2                       | 1                   | 1975               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Hombre              | 169                | 65,4           |                | AB              | Vivo            |
| <b>Antigens</b><br>1101 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000011

**Nom**  
Pedro Duran Montoro

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 17                      | 9                   | 1967               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Hombre              | 170                | 60,8           |                | O               | Vivo            |
| <b>Antigens</b><br>1111 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000012

**Nom**  
David Andujar Pineda

|                         |                     |                    |                |                |                 |                 |
|-------------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b>     | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 12                      | 8                   | 1974               |                |                |                 |                 |
| <b>MinutsMor</b>        | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                         | Hombre              | 180                | 100            |                | AB              | Vivo            |
| <b>Antigens</b><br>0111 |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000013

**Nom**  
Oscar Hinojosa Esquivel

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 18                  | 9                   | 1952               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 190                | 80             |                | A               | Vivo            |

**Antigens**  
0011

\* **CodiSS**  
0000000014

**Nom**  
Silvia Costea Bosch

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 1                   | 2                   | 1990               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Mujer               | 170                | 65,9           |                | AB              | Vivo            |

**Antigens**  
1100

\* **CodiSS**  
0000000016

**Nom**  
Ivan Denia La Torre

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 4                   | 8                   | 1965               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 168                | 60             |                | AB              | Vivo            |

**Antigens**  
1110

\* **CodiSS**  
0000000017

**Nom**  
Marta La Rosa Fernandez

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 1                   | 11                  | 1966               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Mujer               | 162                | 58             |                | A               | Vivo            |

**Antigens**  
1011

\* **CodiSS**  
0000000018

**Nom**  
Damian Garcia Pereira

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 2                   | 12                  | 1955               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 190                | 81             |                | AB              | Vivo            |

**Antigens**  
0001

\* **CodiSS**  
0000000019

**Nom**  
Albert Ribes Peña

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 9                   | 9                   | 1943               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 190                | 90,1           |                | A               | Vivo            |

**Antigens**  
1101

\* **CodiSS**  
0000000020

**Nom**  
Marta Rio Freijo

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 2                   | 1                   | 1969               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Mujer               | 70                 | 63,4           |                | AB              | Vivo            |

**Antigens**  
1110

\* **CodiSS**  
0000000021

**Nom**  
Nombre de prueba

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 12                  | 12                  | 2000               | 12             | 12             | 2000            | 12              |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
| 0                   | Hombre              | 172                | 70             |                | O               | Vivo            |

**Antigens**  
1001

\* **CodiSS**  
0000000022  
**Nom**  
Nombre de prueba2

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 10                  | 12                  | 1976               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 172                | 70             |                | O               | Vivo            |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |
| 1010                |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000023  
**Nom**  
Nombre de prueba 3

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 22                  | 12                  | 1974               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 170                | 72             |                | O               | Muerto          |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |
| 1100                |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000024  
**Nom**  
Nombre de prueba 4

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 29                  | 11                  | 1975               | 12             | 12             | 2000            | 13              |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
| 45                  | Hombre              | 172                | 70             |                | O               | Muerto          |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |
| 1010                |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000025  
**Nom**  
Nombre de prueba 5

|                     |                     |                    |                |                |                 |                 |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
| 13                  | 10                  | 1980               |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Hombre              | 180                | 75             |                | O               | Vivo            |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |
| 1001                |                     |                    |                |                |                 |                 |

\* **CodiSS**  
0000000026  
**Nom**  
Diego Pino Pino

|                     |                     |                    |                |                |                 |                 |    |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|----|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |    |
| 20                  | 12                  | 1950               | 10             | 12             | 2001            |                 | 12 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |    |
| 30                  | Hombre              |                    | 178            |                | 72 O            | Vivo            |    |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |    |
| 1101                |                     |                    |                |                |                 |                 |    |

\* **CodiSS**  
0000000027  
**Nom**  
Paco Pino Pino

|                     |                     |                    |                |                |                 |                 |    |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|----|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |    |
| 12                  | 10                  | 1952               | 12             | 12             | 2000            |                 | 12 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |    |
| 25                  | Hombre              |                    | 175,2          |                | 70,3 O          | Muerto          |    |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |    |
| 1001                |                     |                    |                |                |                 |                 |    |

\* **CodiSS**  
0000000030  
**Nom**  
Luis Paracuellos

|                     |                     |                    |                |                |                 |                 |    |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|----|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |    |
| 10                  | 10                  | 1974               | 12             | 10             | 2001            |                 | 10 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |    |
| 21                  | Hombre              |                    | 175,3          |                | 72,6 O          | Muerto          |    |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |    |
| 1100                |                     |                    |                |                |                 |                 |    |

\* **CodiSS**  
00008  
**Nom**  
Eva Gonzalez Parra

|                     |                     |                    |                |                |                 |                 |  |
|---------------------|---------------------|--------------------|----------------|----------------|-----------------|-----------------|--|
| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixemen</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |  |
| 23                  | 9                   | 1935               |                |                |                 |                 |  |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>      | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |  |
|                     | Mujer               | 170                | 64,2           |                | A               | Vivo            |  |
| <b>Antigens</b>     |                     |                    |                |                |                 |                 |  |
| 1010                |                     |                    |                |                |                 |                 |  |

\* **CodiSS**  
1000000000

**Nom**  
Laura Correa Marin

| <b>DiaNaixement</b> | <b>MesNaixement</b> | <b>AnyNaixement</b> | <b>DiaMort</b> | <b>MesMort</b> | <b>AnyMort</b>  | <b>HoraMort</b> |
|---------------------|---------------------|---------------------|----------------|----------------|-----------------|-----------------|
| 10                  | 10                  | 1960                |                |                |                 |                 |
| <b>MinutsMor</b>    | <b>Sexe</b>         | <b>Alzada</b>       | <b>Pes</b>     |                | <b>TipusSan</b> | <b>Estat</b>    |
|                     | Mujer               | 180                 | 70,4           |                | A               | Vivo            |

**Antigens**  
0010

## **8. Código fuente del sistema multi-agente implementado.**

Los archivos que contienen el código fuente de este sistema multi-agente, están almacenados en la estructura de directorios que empieza con Agent0. En el directorio Agent0 se encuentran los siguientes archivos ejecutables:

**-Entorno.bat:** Actualiza las variables de entorno para poder cargar este sistema (Las variables de entorno cogen valores específicos del ordenador y de la plataforma en la que se ejecuta).

**-Compila.bat:** Compila todo el código (hay que tener en cuenta la unidad de la que cuelga el directorio Agent0).

**-Executa.bat y Executa2.bat:** Ejecuta la plataforma y cargan los agentes correspondientes.

El directorio Agent0 contiene los siguientes subdirectorios:

**agentes:** Contiene el código fuente, que corresponde a los agentes que se han utilizado en este proyecto.

**documentación:** Contiene este archivo.

El directorio agentes contiene los subdirectorios:

**CTH:** Contiene el código fuente que corresponde a *AgentCTH*, y contiene los siguientes subdirectorios: **interficie** y **ontology**. En interficie está el código fuente de la interfaz de AgentCTH y en ontology la ontología.

**A0:** Contiene el código fuente que corresponde a *Ag0*, y contiene los subdirectorios: **ontology** y **BDAgent0**. En ontology se encuentra la ontología que utiliza Ag0 y en BDAgent0 está implementada la base de datos en Access2000 que utiliza Ag0.

**Especial:** Contiene el código fuente que corresponde al proyecto de David Sánchez, pero con las modificaciones pertinentes.

En el punto 8.1 se van a enumerar los archivos que componen al sistema multi-agente, diciendo en que directorio se ha ubicado cada archivo y la función que realizan.

En el punto 8.2 se va a incluir todo el código que se utiliza en este proyecto, pero sin tener en cuenta la estructura de directorios. De esta forma no se introduce código dos veces, una vez para archivos de la ontología de AgentCTH y otra vez para los archivos de la ontología de Ag0. (No se puede tener en cuenta la instrucción package en el código fuente impreso en el apartado 8.2).



## **8.1 Distribución modular del sistema.**

Se recomienda volver a mirar la figura 5.1., para tener una visión general de los agentes y comportamientos que componen a este sistema multiagente.

### **8.1.1 AgentCTH.**

#### **8.1.1.1 Archivos contenidos en ./Agent0/agentes/CTH.**

**AgentCTH.java:** Este archivo contiene la clase AgentCTH que extiende a la clase GuiAgent, esto le permite al agente recibir eventos desde la interfaz gráfica, e iniciar el comportamiento correspondiente para comunicarse con Ag0. También contiene los métodos básicos para implementar un agente, como son: Setup, takeDown, RegistrEnEIDf, DesregistreEnEIDf....

**IntroduccionPacienteRequest.java:** Implementa el comportamiento request, que *inicia* AgentCTH, cuando el usuario ha solicitado que se introduzca los datos de un paciente-órgano. Tanto este comportamiento, como los que se enumeran a continuación en este punto, son los encargados de mostrar por la interfaz gráfica, las respuestas del sistema.

**AnotacionAsignacionOrganoRequest.java:** Implementa el comportamiento request, que *inicia* AgentCTH, cuando el usuario ha solicitado que se anote en la base de datos, la asignación de un órgano a un paciente.

**AnotacionOperacionRealizadaRequest.java:** Implementa el comportamiento request, que *inicia* AgentCTH, cuando el usuario ha solicitado que se anote en la base de datos, la operación realizada al trasplantar un órgano.

**AnotacionOrganoRechazadoRequest.java:** Implementa el comportamiento request, que *inicia* AgentCTH, cuando el usuario ha solicitado que se anote en la base de datos, el rechazo de un órgano.

**AnotacionMuertePacienteRequest.java:** Implementa el comportamiento request, que *inicia* AgentCTH, cuando el usuario ha solicitado que se anote en la base de datos, la muerte de un paciente.

**EncontrarPacientesCompatiblesRequest.java:** Implementa el comportamiento request, que *inicia* AgentCTH, cuando el usuario ha solicitado que se busquen los pacientes compatibles con un órgano.

#### 8.1.1.2 Archivos contenidos en ./Agent0/agentes/CTH/interficie.

**Utilidades.java:** Contiene un conjunto de métodos útiles para: mostrar mensajes de salida del sistema, preparar datos que se enviarán,...

A continuación se muestran los archivos, que contienen el código de las clases asociadas a cada ventana de la interfaz gráfica. Hay una relación uno a uno, entre archivo y ventana de la interfaz gráfica. (Mirar correspondencia en punto 5.1.1.)

- MenuPrincipal.java. (Punto 5.1.1.1).**
- MenuOpciones.java. (Punto 5.1.1.2).**
- MenuIntroducirPaciente.java. (Punto 5.1.1.3).**
- MenuDonate. (Punto 5.1.1.4).**
- MenuMuertePaciente.java. (Punto 5.1.1.5).**
- MenuReceptor.java. (Punto 5.1.1.6).**
- MenuAnotarMuertePaciente.java. (Punto 5.1.1.7).**
- MenuEncontrarPacientesCompatibles.java. (Punto 5.1.1.8).**
- MenuAnotarAsignacionOrgano.java. (Punto 5.1.1.9).**
- MenuAnotarOrganoRechazado.java. (Punto 5.1.1.10).**
- MenuOperacionRealizada.java. (Punto 5.1.1.11).**

#### 8.1.1.3 Archivos contenidos en ./Agent0/agentes/CTH/Ontology.

En este directorio, se encuentran todos los archivos con el código fuente de la ontología utilizada por AgentCTH.

El archivo **OntologiaCTH.java** contiene la definición y estructuración de los componentes de la ontología utilizada por AgentCTH. El **resto de archivos**, contenidos en este directorio, implementan las clases utilizadas en la ontología, y sus correspondientes métodos para facilitar el acceso a la información.

## 8.1.2 Ag0.

### 8.1.2.1 Archivos contenidos en ./Agent0/agentes/A0.

**-UtilidadesAgent0.java:** Define una clase con métodos útiles para la implementación de Ag0.

**-Ag0.java:** Contiene el código fuente del agente Ag0, como los métodos básicos (Setup, takeDown,...). Desde este archivo, se crean los comportamientos que responden a las solicitudes de AgentCTH. En el caso de la acción buscar pacientes compatibles, se inicia un protocolo request con agente Especialista.

**-RespuestaIntroduccionPaciente.java:** Implementa el comportamiento request, que *responde* a AgentCTH, cuando el usuario ha solicitado que se introduzcan los datos de un paciente-organos.

**-RespuestaAsignacionOrganos.java:** Implementa el comportamiento request, que *responde* a AgentCTH, cuando el usuario ha solicitado que se anote la asignación de un órgano a un paciente.

**-RespuestaOperacionRealizada.java:** Implementa el comportamiento request, que *responde* a AgentCTH, cuando el usuario ha solicitado que se anote la operación realizada al trasplantar un órgano.

**-RespuestaOrganosRechazado.java:** Implementa el comportamiento request, que *responde* a AgentCTH, cuando el usuario ha solicitado que se anote el rechazo de un órgano.

**-RespuestaAnotacionMuertePaciente.java:** Implementa el comportamiento request, que *responde* a AgentCTH, cuando el usuario ha solicitado que se anote la muerte de un paciente.

**-RespuestaEncontrarPacientesCompatibles.java:** Implementa el comportamiento request, que inicia Ag0 para comunicarse con Especialista, cuando el usuario ha solicitado que se busquen pacientes compatibles con un órgano. Este comportamiento crea el comportamiento EspecialistaBehaviour.

**-EspecialistaBehaviour.java:** Implementa el comportamiento request, que *responde* a AgentCTH, cuando el usuario ha solicitado que se busquen pacientes compatibles con un órgano.

#### **8.1.2.2 Archivos contenidos en ./Agent0/agentes/A0/Ontology.**

En este directorio, se encuentran todos los archivos con el código fuente de la ontología utilizada por Ag0.

El archivo **OntologiaAg0.java** contiene la definición y estructuración de los componentes de la ontología utilizada por Ag0. El **resto de archivos**, contenidos en este directorio, implementan las clases utilizadas en la ontología, y sus correspondientes métodos para facilitar el acceso a la información.

#### **8.1.2.3 Archivos contenidos en ./Agent0/agentes/A0/BDAgent0.**

En este directorio se encuentra el archivo de Access2000, que corresponde a la base de datos que gestiona Ag0.

### **8.1.3 Especialista.**

#### **8.1.3.1 Archivos contenidos en ./Agent0/agentes/especial.**

En este directorio, se encuentran todos los archivos que contienen el código fuente de los comportamientos y agentes utilizados en el proyecto de David Sánchez.

#### **8.1.3.2 Archivos contenidos en ./Agent0/agentes/especial/Ontology.**

Contiene los archivos necesarios de la ontología, que utiliza el agente Especialista para poder recibir mensajes de otro agente y para comunicarse con los agentes MCDM. El agente Especialista y los agentes MCDM utilizan los mismos archivos para la ontología.

## **8.2 Impresión del código fuente del sistema.**

### 8.2.1 Impresión de código contenido en ./Agent0/agentes/CTH.

#### **8.2.1.1 AgentCTH.java.**

```

/*****

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

AgentCTH que es el encargado de interactuar con usuario y ponerse
en contacto con Agent0 de emergencias.

Un usuario puede pedir realizar las siguientes acciones a traves de
este agente:
    -Introducir paciente (donante/receptor) de organo.
    -Anotar muerte de un paciente.
    -Anotar asignacion de un organo a un paciente.
    -Anotar operacion realizada a un paciente.
    -Anotar rechazo de organo por parte de un paciente operado.
    -Encontrar pacientes compatibles con un determinado organo.

*****/

package Agent0.agentes.CTH;

//Importacion de clases necesarias para implementar un agente.
import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;
import jade.domain.FIPAAgentManagement.*;

//Importacion de clases necesarias para que un agente siga un protocolo.
import jade.proto.*;

import jade.onto.sl.sl.*;

import java.util.*;

//Para crear objeto JTextArea de interficie.
//Servira para mensajes de salida del sistema.
import javax.swing.*;

import java.awt.event.*;
import javax.swing.event.*;

//Librerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Para poder recibir eventos de la interficie grafica.
import jade.gui.*;

//Importacion de clases donde esta implementada la interficie grafica
//asociada a este agente.
import Agent0.agentes.CTH.interficie.*;

//Importacion de clases donde esta definida la ontologia para comunicar
//AgentCTH con Agent0
import Agent0.agentes.CTH.Ontology.*;

/*****
Clase que implementa al agente AgentCTH.
*****/
```

```

*****/
public class AgentCTH extends GuiAgent{
    //Constantes para interactuar con la interficie.
    public final static int ANOTARMUERTEPACIENTE = 2001;
    public final static int ENCONTRARPACIENTESCOMPATIBLES = 2002;
    public final static int ANOTARASIGNACIONORGANO = 2003;
    public final static int ANOTARORGANORECHAZADO = 2004;
    public final static int INTRODUCIRPACIENTE = 2005;
    public final static int OPERACIONREALIZADA = 2006;

    //Servira para mostrar mensajes de salida del sistema.
    JScrollPane PanelSalida = new JScrollPane();
    JTextArea valores = new JTextArea();

    //Inicializacion para:
    // Se crea una instancia de Menu Principal de la interficie y
    // se pasa por parametro una instancia de la clase AgentCTH
    // para poder recibir eventos desde la interficie.
    public static MenuPrincipal menu;

/*****
Constructor de la clase AgentCTH.
*****/
    public AgentCTH() {
        super();
        //Se crea una instancia de MenuPrincipal, en que se pasa una instancia
        //de esta clase para poder recibir eventos desde la interficie. Y
        //un JScrollPane y un JTextArea para mostrar mensajes de salida del
        //sistema.
        menu = new MenuPrincipal(AgentCTH.this, PanelSalida, valores);

        //Para salir de la aplicacion, cuando se cierra la ventana
        //activa de la interficie.
        WindowListener l= new WindowAdapter(){
            public void windowClosing (WindowEvent e){
                System.exit(0);
            }
        };
        menu.addWindowListener(l);
    }

/*****
Especie de handler para los eventos que recibe AgentCTH desde la
interficie.
*****/
    protected void onGuiEvent(GuiEvent ev) {
        switch(ev.getType()) {
            //Inicia la anotacion de la muerte de un paciente.
            case ANOTARMUERTEPACIENTE:
                //Recibe objeto de tipo Anotar Muerte Paciente definida en ontologia.
                AnotarMortPacient accio =
                    (AnotarMortPacient) ev.getSource();

                System.out.println(getLocalName()+" : Peticion desde la"+
                    " interficie! ANOTAR MUERTE PACIENTE");

                //Consulta al DF agente de tipo Agent0 que hay en el sistema.
                java.util.List agentes = Busca_en_DF("PFCAgent0", "Agent0");
                if (!agentes.isEmpty()) { //Si lo puede haber un agent0
                    //Consigue AID de agente buscado.
                    AID aid_agent0 = new AID((String) agentes.get(0));

                    //Crea mensaje.
                    ACLMessage AMP = makeMessage(ACLMessage.REQUEST, SLCodec.NAME,
                        OntologiaCTH.NAME, "FIPA-Request", this.getAID(), aid_agent0,
                        accio);

                    //Instanciacion al BehaviourInitiatorRequest correspondiente.
                    AnotacionMuertePacienteRequest ampr = new

```

```

        AnotacionMuertePacienteRequest(AgentCTH.this,AMP);

        //Para que pueda mostrar mensajes a traves de estos objetos
        //en el comportamiento AnotacionMuertePacienteRequest.
        ampr.PanelSalida=PanelSalida;
        ampr.valores=valores;

        //Aadimos comportamiento Request.
        addBehaviour(ampr);
    }
    else{
        System.out.println("El agent0 no esta registrado en DF.");
        valores.append("\n\n Agent0 no esta registrado en DF.");
    }
}
break;

//Inicia la busqueda de pacientes compatibles con un organo.
case ENCONTRARPACIENTESCOMPATIBLES:

    //Recibe objeto con la accion a realizar.
    TrobarPacientsCompatibles accio_t =
    (TrobarPacientsCompatibles) ev.getSource();

    System.out.println(getLocalName()+" : Peticion desde"+
    " la inteficie! ENCONTRAR PACIENTES COMPATIBLES");

    //Consulta al DF agente de tipo Agent0 que hay en el sistema.
    agentes = Busca_en_DF("PFCAgent0","Agent0");
    if (!agentes.isEmpty()) { //Si lo puede haber un agent0
        //Consigue AID de agente buscado.
        AID aid_agent0 = new AID((String) agentes.get(0));

        //Crea mensaje.
        ACLMessage TPC = makeMessage(ACLMessage.REQUEST,SLCodec.NAME,
        OntologiaCTH.NAME,"FIPA-Request",this.getAID(),aid_agent0,
        accio_t);

        //Instanciacion al BehaviourInitiatorRequest correspondiente.
        EncontrarPacientesCompatiblesRequest epcr = new
        EncontrarPacientesCompatiblesRequest (AgentCTH.this,TPC);

        //Para que pueda mostrar mensajes a traves de estos objetos
        //en el comportamiento EncontrarPacientesCompatibles.
        epcr.PanelSalida=PanelSalida;
        epcr.valores=valores;

        //Aadimos comportamiento Request.
        addBehaviour(epcr);
    }
    else{
        System.out.println("El agent0 no esta registrado en DF.");
        valores.append("\n\n Agent0 no esta registrado en DF.");
    }
}
break;

//Inicia la anotacion de la asignacion de un organo a un paciente.
case ANOTARASIGNACIONORGANO:
    //Recibe accion AssignacioOrgan desde la interficie grafica.
    AssignacioOrgan accio_ao = (AssignacioOrgan) ev.getSource();

    System.out.println(getLocalName()+" : Peticion desde la inteficie!"+
    " ANOTAR ASIGNACION ORGANO");

    //Consulta al DF agente de tipo Agent0 que hay en el sistema.
    agentes = Busca_en_DF("PFCAgent0","Agent0");
    if (!agentes.isEmpty()) { //Si lo puede haber un agent0
        //Consigue AID de agente buscado.
        AID aid_agent0 = new AID((String) agentes.get(0));

```

```

//Crea mensaje.
ACLMessage AAOR = makeMessage(ACLMessage.REQUEST,
SLCodec.NAME,OntologiaCTH.NAME,
"FIPA-Request",this.getAID(),aid_agent0,accio_ao);

//Instanciacion al BehaviourInitiatorRequest correspondiente.
AnotacionAsignacionOrganoRequest aaor =
new AnotacionAsignacionOrganoRequest(AgentCTH.this,AAOR);

//Para que pueda mostrar mensajes a traves de estos objetos
//en el comportamiento AnotacionAsignacionOrganoRequest.
aaor.PanelSalida=PanelSalida;
aaor.valores=valores;

//Añadimos comportamiento Request.
addBehaviour(aaor);
}
else{
System.out.println("El agent0 no esta registrado en DF.");
valores.append("\n\n Agent0 no esta registrado en DF.");
}
}
break;

//Inicia la anotacion de organo rechazado por un paciente.
case ANOTARORGANORECHAZADO:
//Recibe accion OrganRebutxat desde interficie grafica.
OrganRebutxat OR = (OrganRebutxat) ev.getSource();
System.out.println(getLocalName()+" : Peticion desde la inteficie!" +
" ANOTARORGANORECHAZADO.");

//Consulta al DF agente de tipo Agent0 que hay en el sistema.
agentes = Busca_en_DF("PFCAgent0","Agent0");
if (!agentes.isEmpty()) { //Si lo puede haber un agent0
//Consigue AID de agente buscado.
AID aid_agent0 = new AID((String) agentes.get(0));

//Crea mensaje.
ACLMessage ORM = makeMessage(ACLMessage.REQUEST,SLCodec.NAME,
OntologiaCTH.NAME,"FIPA-Request",this.getAID(),aid_agent0,OR);

//Instanciacion al BehaviourInitiatorRequest correspondiente.
AnotacionOrganoRechazadoRequest AOr =
new AnotacionOrganoRechazadoRequest(AgentCTH.this,ORM);

//Pasamos objeto de interficie para mostrar mensajes
//de salida del sistema.
AOr.PanelSalida=PanelSalida;
AOr.valores=valores;

//Añadimos comportamiento Request.
addBehaviour(AOr);
}
else{
System.out.println("El agent0 no esta registrado en DF.");
valores.append("\n\n Agent0 no esta registrado en DF.");
}
}
break;

//Inicia la introduccion de paciente en base de datos.
case INTRODUCIRPACIENTE:
//Recibe objeto Patient definido en la ontologia.
Patient p = (Patient) ev.getSource();
System.out.println(getLocalName()+" : Peticion desde la inteficie!" +
" INTRODUCIRPACIENTE.\n");

//Se crea accion que se pasa a Agent0.
IntroducirPatient accioip = new IntroducirPatient(p);

//Consulta al DF agente de tipo Agent0 que hay en el sistema.

```



```

agentes = Busca_en_DF("PFCAgent0", "Agent0");
if (!agentes.isEmpty()) { //S lo puede haber un agent0
    //Consigue AID de agente buscado.
    AID aid_agent0 = new AID((String) agentes.get(0));

    //Crea mensaje.
    ACLMessage IP = makeMessage(ACLMessage.REQUEST, SLCodec.NAME,
    OntologiaCTH.NAME, "FIPA-Request", this.getAID(),
    aid_agent0, accioip);

    //Instanciacion al BehaviourInitiatorRequest correspondiente.
    IntroduccionPacienteRequest ipr = new
IntroduccionPacienteRequest(AgentCTH.this, IP);

    //Pasamos objeto de interficie para mostrar mensajes
    //de salida del sistema.
    ipr.PanelSalida=PanelSalida;
    ipr.valores=valores;

    //A adimos comportamiento Request.
    addBehaviour(ipr);
}
else{
    System.out.println("El agent0 no esta registrado en DF.");
    valores.append("\n\n Agent0 no esta registrado en DF.");
}
}
break;

//Inicia la anotacion de una operacion realizada.
case OPERACIONREALIZADA:
    //Recibe objeto Operacio de interficie grafica, que esta definido
    //en la ontologia.
    Operacio OPR = (Operacio) ev.getSource();
    System.out.println(getLocalName()+" : Peticion desde la inteficie!" +
    " OPERACIONREALIZADA");

    //Crea accion OperacioRealitzada que pasara a Agent0.
    OperacioRealitzada accio_OR = new OperacioRealitzada(OPR);

    //Consulta al DF agente de tipo Agent0 que hay en el sistema.
    agentes = Busca_en_DF("PFCAgent0", "Agent0");

    if (!agentes.isEmpty()) { //S lo puede haber un agent0
        //Consigue AID de agente buscado.
        AID aid_agent0 = new AID((String) agentes.get(0));

        //Crea mensaje.
        ACLMessage AOR = makeMessage(ACLMessage.REQUEST, SLCodec.NAME,
        OntologiaCTH.NAME, "FIPA-Request", this.getAID(), aid_agent0,
        accio_OR);

        //Instanciacion al BehaviourInitiatorRequest correspondiente.
        AnotacionOperacionRealizadaRequest aorr =
        new AnotacionOperacionRealizadaRequest(AgentCTH.this, AOR);

        //Se pasan instancias de estos objetos para poder mostrar
        //mensajes de salida del sistema.
        aorr.PanelSalida=PanelSalida;
        aorr.valores=valores;

        //A adimos comportamiento Request.
        addBehaviour(aorr);
    }
    else{
        System.out.println("El agent0 no esta registrado en DF.");
        valores.append("\n\n Agent0 no esta registrado en DF.");
    }
}
break;
}
}
}

```

```

/*****
Metodo que se ejecuta la primera vez que se crea el agentes.
*****/
protected void setup(){
    RegistreEnElDf();
}

/*****
Metodo que se ejecuta cuando agentes deja de funcionar.
*****/
protected void takeDown(){
    DesregistreEnElDf();
}

/*****
Metodo que registra agente en el DF.
*****/
public void RegistreEnElDf(){
    ServiceDescription sd1 = new ServiceDescription();

    //Indicamos los parametros del agente.
    sd1.setType("AgentCTH");
    sd1.setName(getLocalName());
    sd1.setOwnership("PFCAgent0");

    //Obtenemos una instancia del DF.
    DFAgentDescription dfd=new DFAgentDescription();

    //Nos añadimos como nuevo servicio.
    dfd.addServices(sd1);

    //Indicamos nuestro nombre
    AID aid=new AID();
    aid.setName(getLocalName());
    dfd.setName(aid);

    //Registramos el lenguaje utilizado.
    registerLanguage(SLCodec.NAME, new SLCodec());

    //Registramos la ontologia.
    registerOntology(OntologiaCTH.NAME,
    OntologiaCTH.instanceCTH());

    System.out.println("Hospital (" + getLocalName() + " de " +
    sd1.getOwnership() + ") on-line");

    try{
        //Nos registramos en el DF
        DFServiceCommunicator.register(this,dfd);
    }catch (FIPAException e) {
        System.err.println(getLocalName()+" : Error en el registro al DF."+
        " Error: "+e.getMessage());
        doDelete();
    }
}

/*****
Metodo que desregistra un agente del DF, utilizando el metodo takeDown.
*****/
public void DesregistreEnElDf(){
    //Obtenemos una instancia del DF
    DFAgentDescription dfd=new DFAgentDescription();
    AID aid=new AID();
    aid.setName(getLocalName());
    dfd.setName(aid);
    try{
        //Nos desregistramos
        DFServiceCommunicator.deregister(this,dfd);
        System.out.println("Agente " + getLocalName() + " eliminado del DF");
    }
}

```

```

    }catch (FIPAException e){
        System.out.println(getLocalName()+
            ": Error en el desregistro del DF. Error: "+e.getMessage());
    }
}

/*****
Metodo para la creacion de un mensaje.
*****/
public ACLMessage makeMessage (int tipus, String language,
String ontology, String protocol, AID sender, AID receiver,
Object content ) {

    //Instanciacion a un ACLMessage.
    ACLMessage message = new ACLMessage(tipus);

    //Se completa mensaje con datos pasados por parametro a esta funcion.
    message.setLanguage(language);
    message.setOntology(ontology);
    message.setProtocol(protocol);
    message.setSender(sender);
    message.addReceiver(receiver);

    //Se crea una accion de tipo SL
    SL_Action a=new SL_Action();
    a.set_0(receiver);
    a.set_1(content);
    java.util.List l = new ArrayList(1);
    l.add(a);

    try{
        //Se introduce contenido en el mensaje con la accion SL creada
        //anteriormente.
        fillContent(message,l);
    } catch (Exception e) {
        System.out.println(getLocalName() + ": Al introducir content.");
        System.out.println(getLocalName() + ": " + e.toString());
    }
    return message;
}

/*****
Metodo que busca en el DF todos los nombres de los agentes que cumplen
un cierto servicio.
*****/
public java.util.List Busca_en_DF (String propietario, String tipo)
{
    java.util.List resultado = new ArrayList();
    DFAgentDescription dfad = new DFAgentDescription();

    ServiceDescription sd = new ServiceDescription();
    java.util.List result;

    sd.setOwnership(propietario);
    sd.setType(tipo);
    dfad.addServices(sd); //Aadimos el servicio al DF, para la busqueda.

    try { //Procedemos a buscar en el df lo que queremos (dfad)
        result = jade.domain.DFServiceCommunicator.search(AgentCTH.this,dfad);
        if (result.isEmpty()) return resultado;
        for (int i=0;i<result.size();i++) {
            dfad = (DFAgentDescription) result.get(i);

            //Obtenemos el nombre del agente.
            resultado.add(dfad.getName().getName());
        }
    }
    catch (Exception exc) {
        System.out.println (exc.toString());
    }
}

```

```
        return resultado;
    }
}
```

### 8.2.1.2 AnotacioAsignacionOrganoRequest.java.

```

/*****
PFC Agent0.
Cristóbal Ledesma Olias
8/10/02

Comportamiento que AgentCTH instancia para comunicarle a Agent0 que
anote la asignación de un organo a un paciente en su base de datos.

*****/

package Agent0.agentes.CTH;

//Importacion de clases necesarias para implementar un comportamiento
//asociado a un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.util.*;

//Librerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Servira para mensajes de salida del sistema.
import javax.swing.*;

import Agent0.agentes.CTH.Ontology.*; //Importacion de la ontologia.
import Agent0.agentes.CTH.interficie.Utilidades;

/*****
Clase que implementa un comportamiento Request: Peticion de AgentCTH para
que Agent0 anote en su base de datos la asignación de un organo a un
paciente.
*****/
public class AnotacionAsignacionOrganoRequest extends
FipaRequestInitiatorBehaviour {

    //Instancia del AgentCTH que se recibe por parametro en el constructor
    //de la clase.
    Agent a;

    private Utilidades u= new Utilidades();

    //Para mostrar mensajes de salida del sistema.
    public JScrollPane PanelSalida;
    public JTextArea valores;

/*****
Constructor del comportamiento que sigue protocolo Request.
*****/
    public AnotacionAsignacionOrganoRequest (Agent agent,
ACLMessage petitionMsg)
    {
        super(agent, petitionMsg);
        a=agent;

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para anotar asignacion de organo"+
        " a paciente con ");
    }
}

```

```

    }

    /*****
    Metodo que se ejecuta cuando comportamiento recibe un mensaje Agree.
    *****/

    protected void handleAgree (ACLMessage msg)
    {
        System.out.println("\n-----
-");
        System.out.println("Requester Agent: CTH aceptado, esperando
resultado...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }

    /*****
    Metodo que se ejecuta cuando comportamiento recibe un mensaje Inform.
    *****/

    protected void handleInform(ACLMessage msg)
    {
        //Mensaje de salida
        valores.append("\n\nAgent0 (Correcto):\n"+
        "Asignacion de organo a paciente realizada correctamente.");
        u.PosicionarFinalScrollBar (PanelSalida);

        System.out.println("\n-----
-");
        System.out.println("Requester Agent: CTH informa, anotacion
realizada...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }

    /*****
    Metodo que se ejecuta cuando comportamiento recibe un mensaje
    NotUnderstood.
    *****/

    protected void handleNotUnderstood(ACLMessage msg)
    {
        //Mensaje de salida
        valores.append("\n\nAgent0 (Ininteligible):\n"+
        "Agent0 ha recibido un mensaje que no sabe interpretar.");
        u.PosicionarFinalScrollBar(PanelSalida);

        System.out.println("\n-----
-");
        System.out.println("Requester Agent: CTH No Entendido...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }

    /*****
    Metodo que se ejecuta cuando comportamiento recibe un mensaje Failure.
    *****/

    protected void handleFailure(ACLMessage msg){
        try
        {
            System.out.println("\n-----
---");
            System.out.println("Requester Agent: CTH Fallo...");
            System.out.println("Requester Agent: mensaje recibido: ");
            System.out.println(msg.toString());

            java.util.List l= a.extractContent(msg);
            Object o = l.get(1);

            //Para saber que tipo de objeto se ha recibido en el contenido de
            //mensaje.

```

```

        ErrorAccesBD eabd =new ErrorAccesBD();

        if ( o.getClass() == eabd.getClass())
        {
            valores.append("\n\nAgent0 (Error):\n"+
                "Se ha producido error al intentar acceder a BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

    }catch(FIPAException fe){
        System.out.println(a.getName()+" : Error en el manegador de missatges:");
        System.out.println(fe.getMessage().toString());
    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Refuse.
*****/
protected void handleRefuse (ACLMessage msg){
    try
    {
        System.out.println("\n-----
        ---");
        System.out.println("Requester Agent: CTH rechazado, esperando
        resultado...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        PacientInexistent pi =new PacientInexistent();
        PacientMort pm =new PacientMort();
        OrganInexistent oi=new OrganInexistent();
        PacientTeOrganAssignat ptoa=new PacientTeOrganAssignat();
        PacientOperat po=new PacientOperat();
        PacientSenseEsperarOrgan pseo=new PacientSenseEsperarOrgan();
        OrganNoDisponible ond=new OrganNoDisponible();
        ErrorAccesBD eabd= new ErrorAccesBD();
        HospitalSenseAutoritzar hsa = new HospitalSenseAutoritzar();

        if ( o.getClass() == hsa.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
                "El hospital, que est solicitando la acci0n, no est autorizado.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pi.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
                "Se quiere asignar organo a paciente que no consta en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pm.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
                "El paciente ya consta como fallecido en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == oi.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
                "El organo introducido no existe en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }
    }
}

```

```

if ( o.getClass() == ptoa.getClass())
{
    valores.append("\n\nAgent0 (Accion Rechazada):\n"+
        "El paciente ya tiene organo asignado.");
    u.PosicionarFinalScrollBar (PanelSalida);
}

if ( o.getClass() == po.getClass())
{
    valores.append("\n\nAgent0 (Accion Rechazada):\n"+
        "Al paciente ya se le ha trasplantado organo de ese tipo.");
    u.PosicionarFinalScrollBar (PanelSalida);
}

if ( o.getClass() == pseo.getClass())
{
    valores.append("\n\nAgent0 (Accion Rechazada):\n"+
        "El paciente no espera organo de ese tipo.");
    u.PosicionarFinalScrollBar (PanelSalida);
}

if ( o.getClass() == ond.getClass())
{
    valores.append("\n\nAgent0 (Accion Rechazada):\n"+
        "El organo no esta disponible.");
    u.PosicionarFinalScrollBar (PanelSalida);
}

if ( o.getClass() == eabd.getClass())
{
    valores.append("\n\nAgent0 (Accion Rechazada):\n"+
        "Error al acceder a BD.");
    u.PosicionarFinalScrollBar (PanelSalida);
}

}catch(FIPAException fe){
    System.out.println(a.getName()+" : Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}
}
}

```



### 8.2.1.3 AnotacionMuertePacienteRequest.java

```

/*****
PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Comportamiento que AgentCTH instancia para comunicarle a Agent0 que
anote la muerte de un paciente en su base de datos.

*****/

package Agent0.agentes.CTH;

//Importacion de clases necesarias para implementar un comportamiento
//asociado a un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.util.*;

//Llibrerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Servira para mensajes de salida del sistema.
import javax.swing.*;

import Agent0.agentes.CTH.Ontology.*; //Importacion de la ontologia.
import Agent0.agentes.CTH.interficie.Utilidades;

/*****
Clase que implementa un comportamiento Request: Peticion de AgentCTH para
que Agent0 anote en su base de datos la muerte de un paciente.
*****/
public class AnotacionMuertePacienteRequest extends
FipaRequestInitiatorBehaviour {

    //Instancia de AgentCTH que se recibe por parametro en el constructor
    //de la clase.
    Agent a;

    //Para mostrar mensajes de salida del sistema.
    private Utilidades u= new Utilidades();
    public JScrollPane PanelSalida;
    public JTextArea valores;

/*****
Constructor del comportamiento Request: Peticion de AgentCTH para
que Agent0 anote en su base de datos la muerte de un paciente.
*****/
    public AnotacionMuertePacienteRequest (Agent agent, ACLMessage petitionMsg)
    {
        super(agent, petitionMsg);
        a=agent;

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para anotar muerte de paciente.");
    }

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Agree.
*****/
    protected void handleAgree (ACLMessage msg)
    {

```

```

        System.out.println("\n-----
-");
        System.out.println("Requester Agent: CTH aceptado, esperando
resultado...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Inform.
*****/
protected void handleInform(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Correcto):\n"+
"Modificacion de estado de paciente realizado correctamente.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH informa, resultado...");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());

}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje NotUnderstood.
*****/
protected void handleNotUnderstood(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Ininteligible):\n"+
"Agent0 ha recibido un mensaje que no sabe interpretar.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH mensaje no entendido.");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());

}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Failure.
*****/
protected void handleFailure(ACLMessage msg){
    try
    {
        System.out.println("\n-----
---");
        System.out.println("Requester Agent: CTH Fallo...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        ErrorAccesBD eabd =new ErrorAccesBD();

        if ( o.getClass() == eabd.getClass())
        {
            valores.append("\n\nAgent0 (Error):\n"+
"Se ha producido error al intentar acceder a BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

    }catch(FIPAException fe){

```

```

        System.out.println(a.getName()+" : Error en el manegador de missatges:");
        System.out.println(fe.getMessage().toString());
    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Refuse.
*****/
protected void handleRefuse (ACLMessage msg){
    try
    {
        System.out.println("\n-----
---");
        System.out.println("Requester Agent: CTH rechazado...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        PacientInexistent pi =new PacientInexistent();
        PacientMort pm =new PacientMort();
        ErrorAccesBD eabd= new ErrorAccesBD();
        HospitalSenseAutoritzar hsa = new HospitalSenseAutoritzar();

        if ( o.getClass() == hsa.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
                "El hospital, que est solicitando la acciøn, no est autorizado.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }
        if ( o.getClass() == pi.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
                "Se quiere cambiar estado de paciente que no consta en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }
        if ( o.getClass() == pm.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
                "El paciente ya consta como fallecido en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == eabd.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
                "Error al acceder a BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }
    }
}
}catch(FIPAException fe){
    System.out.println(a.getName()+" : Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}
}
}

```

#### 8.2.1.4 AnotacionOperacionRealizadaRequest.java

```

/*****
PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Comportamiento que instancia AgentCTH para comunicarle a Agent0 que
anote operacion realizada a un determinado paciente en su BD.

*****/

package Agent0.agentes.CTH;

//Importacion de clases necesarias para implementar un comportamiento
//asociado a un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.util.*;

//Llibrerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Servira para mensajes de salida del sistema.
import javax.swing.*;

import Agent0.agentes.CTH.Ontology.*; //Importacion de la ontologia.
import Agent0.agentes.CTH.interficie.Utilidades;

/*****
Clase que implementa un comportamiento Request: Peticion de AgentCTH para
que Agent0 anote en su base de datos la operacion a un paciente.
*****/
public class AnotacionOperacionRealizadaRequest extends
FipaRequestInitiatorBehaviour {

    //Instancia de AgentCTH que se recibe por parametro en el constructor.
    Agent a;

    //Para mensajes de salida del sistema.
    Utilidades u = new Utilidades();
    public JScrollPane PanelSalida;
    public JTextArea valores;

/*****
Constructor que implementa un comportamiento Request: Peticion de
AgentCTH para que Agent0 anote en su base de datos la operacion a
un paciente.
*****/
    public AnotacionOperacionRealizadaRequest
    (Agent agent, ACLMessage petitionMsg)
    {
        super(agent, petitionMsg);
        a=agent;

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para anotar operacion en BD de Agent0.");
    }

/*****
Metodo que se ejecuta cuando comportamiento recibe mensaje Agree.
*****/
    protected void handleAgree (ACLMessage msg)
    {

```

```

        System.out.println("\n-----
-");
        System.out.println("Requester Agent: CTH aceptado, esperando
resultado...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }

/*****
Metodo que se ejecuta cuando comportamiento recibe mensaje Inform.
*****/
protected void handleInform(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Correcto):\n"+
"Anotacion de trasplante de organo se ha realizado con exito.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH informa, resultado...");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe mensaje NotUnderstood.
*****/
protected void handleNotUnderstood(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Ininteligible):\n"+
"Agent0 ha recibido un mensaje que no sabe interpretar.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH mensaje no entendido.");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe mensaje Failure.
*****/
protected void handleFailure(ACLMessage msg){
    try
    {
        System.out.println("\n-----
---");
        System.out.println("Requester Agent: CTH Fallo...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        ErrorAccesBD eabd =new ErrorAccesBD();

        if ( o.getClass() == eabd.getClass())
        {
            valores.append("\n\nAgent0 (Error):\n"+
"Se ha producido error al intentar acceder a BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }
    }catch(FIPAException fe){
        System.out.println(a.getName()+" Error en el manegador de missatges:");
        System.out.println(fe.getMessage().toString());
    }
}

```

```

    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe mensaje Refuse.
*****/
protected void handleRefuse(ACLMessage msg){
    try
    {
        System.out.println("\n-----
        ---");
        System.out.println("Requester Agent: CTH mensaje rechazado.");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        OrganInexistent oi =new OrganInexistent();
        PacientMort pm =new PacientMort();
        PacientInexistent pi = new PacientInexistent();
        ErrorAccesBD eabd= new ErrorAccesBD();
        PacientSenseOrganAssignat psoa = new PacientSenseOrganAssignat();
        OrganNoDisponibile ond = new OrganNoDisponibile();
        HospitalSenseAutoritzar hsa = new HospitalSenseAutoritzar();

        if ( o.getClass() == hsa.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
            "El hospital, que est solicitando la acci0n, no est autorizado.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == ond.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
            "El organo ya ha sido trasplantado.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pi.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
            "El organo figura como asignado en BD, pero no se puede relacionar\n"+
            "con ningun paciente.\n"+
            "ERROR GRAVE DE CONSISTENCIA EN BD:\n"+
            "El organo esta asignado a un paciente que no esta\n"+
            "registrado en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == oi.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
            "El organo no esta registrado en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pm.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
            "El paciente que tenia asignado el organo ha fallecido.\n"+
            " ATENCION: Si un paciente muere despues de ser operado,\n"+
            " primero se introducira la operacion\n"+
            " y luego se anotara la muerte.\n"+
            " En caso contrario no se podria introducir\n"+
            " operacion en BD y se tendria que dirigir\n"+
            " a Adminitrador de la BD.");
        }
    }
}

```

```

        u.PosicionarFinalScrollBar (PanelSalida);
    }

    if ( o.getClass() == eabd.getClass())
    {
        valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Error al acceder a BD.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }

    if ( o.getClass() == psoa.getClass())
    {
        valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "No consta en base de datos que el organo este asignado a ningun\n"+
            "paciente que este vivo.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }
}
}
}
}
}

```

### 8.2.1.5 AnotacionOrganoRechazadoRequest.java

```

/*****
Agent0.agentes.CTH.

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Comportamiento que instancia AgentCTH para comunicarle a Agent0 que
anote en su BD, que un paciente a rechazado un organo.

*****/

package Agent0.agentes.CTH;

//Importacion de clases necesarias para implementar un comportamiento
//asociado a un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.util.*;

//Llibrerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Servira para mensajes de salida del sistema.
import javax.swing.*;

import Agent0.agentes.CTH.Ontology.*; //Importacion de la ontologia.
import Agent0.agentes.CTH.interficie.Utilidades;

/*****
Clase que implementa un comportamiento Request: Peticion de AgentCTH
para que Agent0 anote en su base de datos el rechazo de un organo por
parte de un paciente.
*****/
public class AnotacionOrganoRechazadoRequest extends
FipaRequestInitiatorBehaviour {

    //Instancia de AgentCTH que se recibe por parametro en el constructor
    //de la clase.
    Agent a;

    //Para mensajes de salida del sistema.
    Utilidades u=new Utilidades();
    public JScrollPane PanelSalida;
    public JTextArea valores;

/*****
Constructor que implementa un comportamiento Request: Peticion de
AgentCTH para que Agent0 anote en su base de datos el rechazo de un
organos por parte de un paciente.
*****/
    public AnotacionOrganoRechazadoRequest (Agent agent, ACLMessage petitionMsg)
    {
        super(agent, petitionMsg);
        a=agent;

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para anotar rechazo de organo en BD"+
        " de Agent0.");
    }
}

```



```

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Agree.
*****/
protected void handleAgree (ACLMessage msg)
{
    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH aceptado, esperando
resultado...");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Inform.
*****/
protected void handleInform(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Correcto):\n"+
"Anotacion de rechazo de organo se ha realizado con exito.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH informa, resultado...");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje
NotUnderstood.
*****/
protected void handleNotUnderstood(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Ininteligible):\n"+

"Agent0 ha recibido un mensaje que no sabe interpretar.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH mensaje no entendido.");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Failure.
*****/
protected void handleFailure(ACLMessage msg){
    try
    {
        System.out.println("\n-----
---");
        System.out.println("Requester Agent: CTH Fallo.");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        ErrorAccesBD eabd =new ErrorAccesBD();

        if ( o.getClass() == eabd.getClass())

```

```

        {
            valores.append("\n\nAgent0 (Error):\n"+
                "Se ha producido error al intentar acceder a BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

    }catch(FIPAException fe){
        System.out.println(a.getName()+" : Error en el manegador de missatges:");
        System.out.println(fe.getMessage().toString());
    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Refuse.
*****/
protected void handleRefuse(ACLMessage msg){
    try
    {
        System.out.println("\n-----
        ---");
        System.out.println("Requester Agent: CTH rechazado.");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        OrganInexistent oi =new OrganInexistent();
        PacientMort pm =new PacientMort();
        PacientInexistent pi = new PacientInexistent();
        ErrorAccesBD eabd= new ErrorAccesBD();
        PacientSenseOperar pso = new PacientSenseOperar();
        OrganJaEstavaRebutxat ojer = new OrganJaEstavaRebutxat();
        HospitalSenseAutoritzar hsa = new HospitalSenseAutoritzar();

        if ( o.getClass() == hsa.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
                "El hospital, que est solicitando la acciøn, no est autorizado.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pi.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
                "El organo figura como trasplantado en BD, pero no se le puede\n"+
                "relacionar con ningun paciente.\n"+
                "ERROR GRAVE DE CONSISTENCIA EN BD:\n"+
                "El organo ha sido trasplantado y el paciente no esta\n"+
                "registrado en BD.");

            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == oi.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
                "El organo no esta registrado en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pm.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
                " El paciente con este organo trasplantado no esta vivo.\n"+
                " ATENCION: Si un paciente muere despues de ser operado,\n"+
                " y rechaza un organo; primero se introducira\n"+
                " el rechazo y luego se anotara la muerte.\n"+
                " En caso contrario no se podria introducir\n"+

```



### 8.2.1.6 EncontrarPacientesCompatiblesRequest.java

```

/*****
PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Comportamiento que AgentCTH instancia para comunicarle a Agent0 que
inicie la búsqueda de pacientes compatibles con un determinado organo.

*****/

package Agent0.agentes.CTH;

//Importacion de clases necesarias para implementar un comportamiento
//asociado a un agente.

//Jade Core
import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.FIPAAException;

//Comportamiento
import jade.proto.*;

import jade.domain.*;
import jade.domain.FIPAAgentManagement.*;

//Language de comunicacion SL
import jade.lang.sl.sl.*;
import jade.lang.acl.*;

//Ontologias
import jade.onto.*;
import jade.onto.sl.sl.*;

import java.io.*;
import java.util.*;

//Servira para mensajes de salida del sistema.
import javax.swing.*;

import Agent0.agentes.CTH.Ontology.*; //Importacion de la ontologia.
import Agent0.agentes.CTH.interficie.Utilidades;

/*****
Clase que implementa un comportamiento Request: Peticion de AgentCTH para
que Agent0 inicie búsqueda de pacientes compatibles con un determinado
organos.
*****/
public class EncontrarPacientesCompatiblesRequest extends
FipaRequestInitiatorBehaviour {

    //Instancia del AgentCTH que se recibe por parametro en el constructor
    //de la clase.
    Agent a;

    private Utilidades u= new Utilidades();

    //Para mostrar mensajes de salida del sistema.
    public JScrollPane PanelSalida;
    public JTextArea valores;

/*****
Constructor del comportamiento que sigue protocolo Request.
*****/
    public EncontrarPacientesCompatiblesRequest (Agent agent,
ACLMessage petitionMsg)
    {

```

```

        super(agent, peticionMsg);
        a=agent;

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para encontrar pacientes compatibles"+
        " con un determinado organo.");
    }

    /*****
    Metodo que se ejecuta cuando comportamiento recibe un mensaje Agree.
    *****/
    protected void handleAgree (ACLMessage msg)
    {
        System.out.println("\n-----
        -");
        System.out.println("Requester Agent: CTH aceptado, esperando
        resultado...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }

    /*****
    Metodo que se ejecuta cuando comportamiento recibe un mensaje Inform.
    *****/
    protected void handleInform(ACLMessage msg)
    {
        try{
            System.out.println("\n-----
            ---");
            System.out.println("Requester Agent: CTH informa, resultado...");
            System.out.println("Requester Agent: mensaje recibido: ");
            System.out.println(msg.toString());

            //Extraemos contenido del mensaje recibido.
            java.util.List l= a.extractContent(msg);
            SL_Result sl = (SL_Result) l.get(0);

            Iterator it = sl.getAll_1();

            ResultatBusqueda rb = (ResultatBusqueda) it.next();

            Long NumeroPacients = (Long) rb.getNPacients();

            valores.append("\n\n\nNumero total de pacientes compatibles con organo:
            "+
            NumeroPacients.intValue());

            //Contendra una lista de pacientes asociados a un organo.
            Iterator Resultat = rb.getAllAssignacio();

            //Contendra un paciente asociado a un organo.
            Associacio As = new Associacio();

            while (Resultat.hasNext())
            {
                As=(Associacio) Resultat.next();
                valores.append("\n\nCodigo Seguridad Social de Paciente compatible:
                ");
                valores.append(u.StringConEspaciosEnBlanco(As.getCodiSS()));

                //Se hace traduccion del catalan al castellano del contenido de grau.
                String g = u.StringConEspaciosEnBlanco(As.getGrau());
                String grau="muy bueno";
                if (g.equals("pessim")) grau="pesimo";
                if (g.equals("dolent")) grau="malo";
                if (g.equals("desfavorable")) grau="desfavorable";
                if (g.equals("fluix")) grau="flojo";
                if (g.equals("factible")) grau="factible";
                if (g.equals("adequat")) grau="adecuado";
                if (g.equals("bo")) grau="bueno";
            }
        }
    }

```

```

        if (g.equals("optim")) grau="optimo";

        valores.append("\nCon grado de compatibilidad: ");
        valores.append(grau);
        u.PosicionarFinalScrollBar (PanelSalida);
    }

    valores.append("\n\nAgent0 (Correcto):\n"+
        "Busqueda de pacientes compatibles con organo realizada
correctamente.");
    u.PosicionarFinalScrollBar (PanelSalida);

} catch (FIPAException fe){
    System.out.println(a.getName()+" : Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}

}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje
NotUnderstood.
*****/
protected void handleNotUnderstood(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Ininteligible):\n"+
        "Agent0 ha recibido un mensaje que no sabe interpretar.");
    u.PosicionarFinalScrollBar(PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH mensaje no entendido.");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Failure.
*****/
protected void handleFailure(ACLMessage msg){
    try
    {
        System.out.println("\n-----
---");
        System.out.println("Requester Agent: CTH Fallo.");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        //Extraemos el contenido del mensaje
        List l=myAgent.extractContent(msg);
        //Obtenemos el objeto enviado
        Object rao = l.get(1);

        //Buscamos en la ontologia la clase que se corresponde con objeto.
        Ontology o = myAgent.lookupOntology(msg.getOntology());
        String nomRao = o.getRoleName(rao.getClass());
        //La indicamos
        System.out.println("La razon del fallo es: " + nomRao);

        valores.append("\n\nAgent0 (Error):\n"+
            "Se ha producido error al intentar realizar accion.\n"+
            "Razon: "+nomRao);

        u.PosicionarFinalScrollBar (PanelSalida);
    } catch (FIPAException fe){
        System.out.println(a.getName()+" : Error en el manegador de missatges:");
        System.out.println(fe.getMessage().toString());
    } catch (OntologyException oe){
        System.err.println("OntologyException leyendo el fallo: " + oe);
    }
}

```

```

    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Refuse.
*****/
protected void handleRefuse (ACLMessage msg){
    try
    {
        System.out.println("\n-----
        ---");
        System.out.println("Requester Agent: CTH rechazado.");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.

        PacientInexistent pi = new PacientInexistent();
        OrganInexistent oi = new OrganInexistent();

        PesosNoValids pnv =new PesosNoValids();
        ClasesInsuficients ci =new ClasesInsuficients();
        MassaAtributs ma =new MassaAtributs();
        MassaObjectes mo =new MassaObjectes();
        AlgoritmeNoImplementat ani = new AlgoritmeNoImplementat();

        AgentNoRegistrat anor = new AgentNoRegistrat();
        ErrorAccesBD eabd= new ErrorAccesBD();

        HospitalSenseAutoritzar hsa = new HospitalSenseAutoritzar();

        if ( o.getClass() == hsa.getClass())
        {
            valores.append("\n\nAgent0 (Accion rechazada):\n"+
            "El hospital, que est solicitando la acci0n, no est autorizado.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pi.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "No hay pacientes que necesiten este organo.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == oi.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "No existe el organo con ese IdOrgan.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pnv.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Agente Especialista no puede realizar ordenacion."+
            "\nNo se ha introducido valores correctos para los pesos asociados a"+
            " los atributos.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == ci.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Agente Especialista no puede realizar ordenacion."+
            "\nAgente Especialista no ha obtenido suficientes clases en el "+

```

```

        " proceso de ordenacion de pacientes.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }

    if ( o.getClass() == ma.getClass())
    {
        valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Agente Especialista no puede realizar ordenacion."+
            "\nSe le intenta pasar demasiados atributos a agente Especialista en
"+
            "el proceso de ordenacion de pacientes.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }

    if ( o.getClass() == mo.getClass())
    {
        valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Agente Especialista no puede realizar ordenacion."+
            "\nSe le intenta pasar demasiados pacientes a agente Especialista en
"+
            "el proceso de ordenacion de pacientes.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }

    if ( o.getClass() == ani.getClass())
    {
        valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Agente Especialista no puede realizar ordenacion."+
            "\nNo esta implementado el algoritmo asociado al agente escogido.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }

    if ( o.getClass() == anor.getClass())
    {
        valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Agente Especialista no puede realizar ordenacion."+
            "\nAlgun agente necesario para realizar esta accion no esta
registrado.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }

    if ( o.getClass() == eabd.getClass())
    {
        valores.append("\n\nAgent0 (Accion Rechazada):\n"+
            "Error al acceder a BD.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }

} catch (FIPAException fe){
    System.out.println(a.getName()+" : Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}
}
}

```



### 8.1.2.7 IntroduccionPacienteRequest.java

```

/*****
Agent.agentes.CTH.

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Comportamiento que instancia AgentCTH para comunicarle a Agent0 que
introduzca los datos de un paciente en su base de datos.

*****/

package Agent0.agentes.CTH;

//Importacion de clases necesarias para implementar un comportamiento
//asociado a un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.util.*;

//Llibrerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Servira para mensajes de salida del sistema.
import javax.swing.*;

import Agent0.agentes.CTH.Ontology.*; //Importacion de la ontologia.
import Agent0.agentes.CTH.interficie.Utilidades;

/*****
Clase que implementa un comportamiento Request: Peticion de AgentCTH
para que Agent0 anote en su base de datos la muerte de un paciente.
*****/
public class IntroduccionPacienteRequest extends
FipaRequestInitiatorBehaviour {

    //Instancia de AgentCTH que se recibe por parametro en el constructor
    //de este comportamiento.
    Agent a;

    //Para mensajes de salida del sistema.
    Utilidades u = new Utilidades();
    public JScrollPane PanelSalida;
    public JTextArea valores;

/*****
Constructor del comportamiento.
*****/
    public IntroduccionPacienteRequest (Agent agent, ACLMessage peticionMsg)
    {
        super(agent, peticionMsg);
        a=agent;

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para introducir datos de paciente.");
    }

/*****
Metodo que se ejecuta cuando comportamiento recibe un Agree.
*****/
    protected void handleAgree (ACLMessage msg)

```

```

    {
        System.out.println("\n-----
-");
        System.out.println("Requester Agent: CTH aceptado, esperando
resultado...");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }

/*****
Metodo que se ejecuta cuando comportamiento recibe un Inform.
*****/
protected void handleInform(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Correcto):\n"+
        "Introduccion de paciente-organo se ha realizado con exito.");
    valores.append("\n ATENCION: La accion realizada no modifica los datos");
    valores.append("\n          registrados con anterioridad en BD.");
    valores.append("\n          Esto quiere decir que si quiere modificar
datos");
    valores.append("\n          ya registrados en BD, se ha de dirigir a
admi-");
    valores.append("\n          nistrador de BD.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH informa.");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un NotUnderstood.
*****/
protected void handleNotUnderstood(ACLMessage msg)
{
    //Mensaje de salida
    valores.append("\n\nAgent0 (Ininteligible):\n"+
        "Agent0 ha recibido un mensaje que no sabe interpretar.");
    u.PosicionarFinalScrollBar (PanelSalida);

    System.out.println("\n-----
-");
    System.out.println("Requester Agent: CTH mensaje no entendido.");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un Failure.
*****/
protected void handleFailure(ACLMessage msg){
    try
    {
        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        ErrorAccesBD eabd =new ErrorAccesBD();

        if ( o.getClass() == eabd.getClass())
        {
            valores.append("\n\nAgent0 (Error):\n"+
                "Se ha producido error al intentar acceder a BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }
    }
}

```

```

        System.out.println("\n-----
-");
        System.out.println("Requester Agent: CTH Fallo.");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());
    }catch(FIPAException fe){
        System.out.println(a.getName()+" : Error en el manegador de missatges:");
        System.out.println(fe.getMessage().toString());
    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un Refuse.
*****/
protected void handleRefuse(ACLMessage msg){
    try
    {
        java.util.List l= a.extractContent(msg);
        Object o = l.get(1);

        //Para saber que tipo de objeto se ha recibido en el contenido de
        //mensaje.
        OrganExistent oe =new OrganExistent();
        PacientMort pm =new PacientMort();
        ErrorAccesBD eabd= new ErrorAccesBD();
        PacientTeOrganAssignat ptoa = new PacientTeOrganAssignat();

        if ( o.getClass() == oe.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
                "Se intenta introducir organo que ya existe en BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == pm.getClass())
        {
            valores.append("\n\nAgent0 (Rechazada accion):\n"+
                "Se intenta solicitar organo para paciente que ha fallecido.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == eabd.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
                "Error al acceder a BD.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }

        if ( o.getClass() == ptoa.getClass())
        {
            valores.append("\n\nAgent0 (Accion Rechazada):\n"+
                "Ya consta en base de datos, que el paciente\n"+
                "ha de recibir un organo como el que se esta\n"+
                "introduciendo.");
            u.PosicionarFinalScrollBar (PanelSalida);
        }
        System.out.println("\n-----
---");
        System.out.println("Requester Agent: CTH Rechazado.");
        System.out.println("Requester Agent: mensaje recibido: ");
        System.out.println(msg.toString());

    }catch(FIPAException fe){
        System.out.println(a.getName()+" : Error en el manegador de missatges:");
        System.out.println(fe.getMessage().toString());
    }
}
}

```

## 8.2.2 Impresión de código contenido en ./Agent0/agentes/interficie.

### 8.2.2.1 MenuAnotarAsignacionOrgano.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Implementacion de menu para introducir datos relativos a la asignacion
de un organo a un paciente receptor desde interficie grafica asociada a
AgentCTH.

*****/

package Agent0.agentes.CTH.interficie;

//Importamos la clase que implementa el AgentCTH.
import Agent0.agentes.CTH.*;

//Importamos la ontologia para comunicar AgentCTH y Agent0.
import Agent0.agentes.CTH.Ontology.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

import jade.gui.*;

/*****
    Clase que muestra y controla los componentes que permiten
    al usuario introducir los datos que corresponden a la
    asignacion de un organo a un paciente.

    Se crea un objeto de esta clase cuando el usuario elige
    la opcion anotar asignacion de organo en el ComboBox del
    menu principal.

    Tiene la misma estructura de programacion que la del menu
    principal.
*****/
public class MenuAnotarAsignacionOrgano extends JFrame{

    //Recibe por parametro una instancia de la clase AgentCTH, para
    //poder enviar eventos al objeto correspondiente.
    private AgentCTH agentcth;

    //Clase que contiene metodos utiles en diferentes archivos.
    Utilidades u=new Utilidades();

    //Objeto definido en la ontologia, que sirve para recoger datos desde
    //la interficie y no perderlos mientras se navega por esta.
    Hospital hospital;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Anotar Asignación de órgano a Paciente.";

    Container contenedorprincipal;

    //Variables globales
    String idorgano;
    String codigoss;

    //Componentes Swing
    AbstractBorder borde = new TitledBorder("Dato identificador de órgano.");

```

```

AbstractBorder borde2 = new TitledBorder("Dato identificador de paciente.");

JPanel panel = new JPanel();
JPanel panel2 = new JPanel();

JLabel e_idorgano = new JLabel ("Identificador òrgano: ");
JLabel e_codigoss = new JLabel ("Código SS del paciente: ");

JTextField t_idorgano = new JTextField();
JTextField t_codigoss = new JTextField();

JButton anterior = new JButton ("Anterior");
JButton siguiente = new JButton ("Siguiente");
JButton salir = new JButton ("Salir");

//Para mostrar mensajes de salida del sistema.
//Se recibe por parametro una instancia de estos objetos, al llamar
//al constructor de esta clase. Estos objetos se crean en la clase
//AgentCTH para tener control sobre ellos desde este agente y sus
//comportamientos asociados.
private JScrollPane PanelSalida;
private JTextArea valores;

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interficie.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor de la clase que implementa el MenuAnotarAsignacionOrgano.
*****/
public MenuAnotarAsignacionOrgano(AgentCTH a, JScrollPane panelsalida,
JTextArea salida){
    super(TITLE);
    setSize(WIDTH,HEIGHT);
    setResizable(false);

    agentcth = a;
    valores = salida;
    valores.setText(""); //Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.
    PanelSalida = panelsalida;

    elementos();
    inieventos();
    show();
}

/*****
Metodo que describe elementos que aparecen en este menu y su colocacion.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);
    panel2.setLayout(null);

    e_idorgano.setBounds(10,30,140,20);
    e_codigoss.setBounds(10,30,150,20);

    t_idorgano.setBounds(150,30,100,20);
    t_codigoss.setBounds(160,30,100,20);

    anterior.setBounds(20,230,100,20);
    siguiente.setBounds(150,230,100,20);
    salir.setBounds(310,230,100,20);

```

```

panel.add(e_idorgano);
panel.add(t_idorgano);
panel.setBorder(borde);
panel.setBounds(10,10,400,75);

panel2.add(e_codigoss);
panel2.add(t_codigoss);
panel2.setBorder(borde2);
panel2.setBounds(10,100,400,75);

contenedorprincipal.add(panel);
contenedorprincipal.add(panel2);

contenedorprincipal.add(anterior);
contenedorprincipal.add(siguiete);
contenedorprincipal.add(salir);

//Para mostrar mensajes de salida del sistema.
valores.setToolTipText ("Mensajes de salida procedentes del sistema");
valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
PanelSalida.setViewPortView(valores);
PanelSalida.setBounds(10,260,400,150);

contenedorprincipal.add(PanelSalida);
}

/*****
Metodo que inicializa los eventos asociados a elementos de este menu.
*****/
void inieventos(){
    //Para eventos producidos sobre el boton anterior.
    //Para poder pulsar boton con tecla.
    anterior.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Anterior();
        }
    });

    anterior.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Anterior();
        }
    });

    //Para eventos producidos sobre el boton siguiete.
    //Para poder pulsar boton con tecla.
    siguiete.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiete();
        }
    });

    siguiete.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiete();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {

```

```

        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });

}

/*****
Metodo que se ejecuta al oprimir el usuario el boton 'Anterior'.
*****/
private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op = new MenuOpciones(agentctth, PanelSalida, valores);

    op.hospital=hospital;

    op.addWindowListener(l);

    dispose();
}

/*****
Metodo que se ejecuta al oprimir el usuario el boton 'Siguiete'.
*****/
private void Ejecutar_Siguiete()
{
    idorgano=" "+t_idorgano.getText();
    codigoss=" "+t_codigoss.getText();
    if (ControlErrores()==true)
    { //Para eliminar caracteres en blanco.
        idorgano=u.StringSinEspaciosEnBlanco(idorgano);
        codigoss=u.StringSinEspaciosEnBlanco(codigoss);

        //Envia evento a la clase AgentCTH.
        AssignacioOrgan AO = new AssignacioOrgan(codigoss,idorgano,hospital);
        agentctth.postGuiEvent(new
        jade.gui.GuiEvent(AO,agentctth.ANOTARASIGNACIONORGANO));
    }
}

/*****
Metodo que valida los datos introducidos por usuario.
*****/
boolean ControlErrores()
{
    if (t_idorgano.getText().equals(""))
    {
        valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return (false);
    }
    if ((t_idorgano.getText()).length(>10)
    {
        valores.append("\n\nERROR:\n"+
        "El identificador de organo no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    if (t_codigoss.getText().equals(""))
    {
        valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    if ((t_codigoss.getText()).length(>10)
    {

```

```
        valores.append("\n\nERROR:\n"+
        "El codigo de seguridad social no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }
    return(true);
}
```



### 8.2.2.2 MenuAnotarMuertePaciente.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Implementacion de menu para introducir datos relativos a la muerte
de un paciente desde interficie grafica asociada a AgentCTH.

*****/
package Agent0.agentes.CTH.interficie;

//Importamos la clase que implementa al AgentCTH.
import Agent0.agentes.CTH.*;
import Agent0.agentes.CTH.Ontology.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

/*****
Clase que muestra y maneja los componentes del submenu
que permite anotar la muerte de un paciente.

Se crea un objeto de esta clase cuando el usuario elige
la accion anotar muerte de un paciente en el ComboBox del
menu opciones.

Tiene la misma estructura de programacion que la vista
en el menu principal.
*****/
public class MenuAnotarMuertePaciente extends JFrame{

    //Objeto que recibe una instancia de AgentCTH por parametro en el
    //constructor de la clase.
    private AgentCTH agentcth;

    //Clase que contiene metodos utiles para diferentes archivos de la
    //interficie grafica.
    private Utilidades u = new Utilidades();

    //Objetos definidos en ontologia, que contendran valores introducidos
    //por usuario, mientras navega por interficie grafica.
    Hospital hospital;
    MortPacient mortpatient;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Anotar Muerte del Paciente.";

    Container contenedorprincipal;

    //Variables globales
    String codiss;
    String fechamuerte;
    String horamuerte;

    //Componentes Swing
    AbstractBorder borde = new TitledBorder("Datos sobre la muerte del
paciente.");

    JPanel panel = new JPanel();

    JLabel e_codiss = new JLabel ("Codigo SS: ");
    JLabel e_fechamuerte = new JLabel ("Dia Muerte (dd/mm/aaaa:");

```

```

JLabel e_horamuerte = new JLabel ("Hora Muerte (HH:MM):");

JTextField t_codiss = new JTextField();
JTextField t_fechamuerte = new JTextField();
JTextField t_horamuerte = new JTextField();

JButton anterior = new JButton ("Anterior");
JButton siguiente = new JButton ("Siguiente");
JButton salir = new JButton ("Salir");

//Para mostrar mensajes de salida del sistema.
private JScrollPane PanelSalida;
private JTextArea valores;

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interficie.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor de la clase que implementa este menu de la interficie grafica
*****/
public MenuAnotarMuertePaciente(AgentCTH a,JScrollPane panelsalida,
JTextArea salida){
    super(TITLE);
    setSize(WIDTH,HEIGHT);
    setResizable(false);

    agentctth = a;
    valores = salida;
    valores.setText("");//Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.
    PanelSalida = panelsalida;
    elementos();
    inieventos();
    show();
}

/*****
Metodo que describe elementos que apareceran en este menu y su posicion.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_codiss.setBounds(10,30,70,20);
    e_fechamuerte.setBounds(10,60,200,20);
    e_horamuerte.setBounds(10,90,200,20);

    t_codiss.setBounds(75,30,100,20);
    t_fechamuerte.setBounds(170,60,75,20);
    t_horamuerte.setBounds(150,90,50,20);

    anterior.setBounds(20,230,100,20);
    siguiente.setBounds(150,230,100,20);
    salir.setBounds(310,230,100,20);

    panel.add(e_codiss);
    panel.add(e_fechamuerte);
    panel.add(e_horamuerte);

    panel.add(t_codiss);
    panel.add(t_fechamuerte);
    panel.add(t_horamuerte);

```

```

panel.setBorder(borde);

panel.setBounds(10,10,400,150);

contenedorprincipal.add(panel);

contenedorprincipal.add(anterior);
contenedorprincipal.add(siguiete);
contenedorprincipal.add(salir);

//Para mostrar mensajes de salida del sistema.
valores.setToolTipText ("Mensajes de salida procedentes del sistema");
valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
PanelSalida.setViewPortView(valores);
PanelSalida.setBounds(10,260,400,150);
contenedorprincipal.add(PanelSalida);
}

/*****
Metodo que inicializa los eventos asociados a componentes de este menu.
*****/
void inieventos(){
    //Para eventos producidos sobre el boton anterior.
    //Para poder pulsar boton con tecla.
    anterior.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Anterior();
        }
    });

    anterior.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Anterior();
        }
    });

    //Para eventos producidos sobre el boton siguiete.
    //Para poder pulsar boton con tecla.
    siguiete.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiete();
        }
    });

    siguiete.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiete();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });
}

/*****
Metodo que se ejecutara cuando usuario oprima el boton 'Anterior'.
*****/

```

```

*****/
private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op = new MenuOpciones(agentcth, PanelSalida, valores);

    op.hospital=hospital;

    op.addWindowListener(l);

    dispose();
}

/*****
Metodo que se ejecutara cuando usuario oprima el boton 'Siguiete'.
*****/
private void Ejecutar_Siguiete()
{
    codiss=" "+t_codiss.getText();
    fechamuerte=t_fechamuerte.getText();
    horamuerte=t_horamuerte.getText();
    if (ControlErrores()==true)
    {
        codiss=u.StringSinEspaciosEnBlanco(codiss);

        //Para pasar datos a objetos definidos en ontologia.
        //Para pasar fecha de muerte de string a 3 variables Long.
        char f[] = fechamuerte.toCharArray();
        Long dmort=new
Long(Character.digit(f[0],10)*10+Character.digit(f[1],10));
        Long mmort=new
Long(Character.digit(f[3],10)*10+Character.digit(f[4],10));
        Long amort=new
Long(Character.digit(f[6],10)*1000+Character.digit(f[7],10)*100+Character.digit(f[8],10)*10+Character.digit(f[9],10));
        //Para pasar hora de muerte de string a 2 variables Long.
        char h[] = horamuerte.toCharArray();
        Long hmort=new
Long(Character.digit(h[0],10)*10+Character.digit(h[1],10));
        Long mimort=new
Long(Character.digit(h[3],10)*10+Character.digit(h[4],10));

        mortpacient=new MortPacient(codiss,dmort,mmort,amort,hmort,mimort);

        AnotarMortPacient anotarmortpacient = new AnotarMortPacient
(mortpacient,hospital);

        //Envia evento a la instancia de la clase AgentCTH.
        agentcth.postGuiEvent(new
jade.gui.GuiEvent(anotarmortpacient,agentcth.ANOTARMUERTEPACIENTE));
    }
}

/*****
Metodo que valida los datos introducidos por usuario.
*****/
boolean ControlErrores()
{
    if (t_codiss.getText().equals(""))
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    if ((t_codiss.getText()).length(>10)
    {
        valores.append("\n\nERROR:\n"+
"El codigo de seguridad social no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
    }
}

```

```

        return false;
    }

    if (fechamuerte.length()==0)
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    else
    {
        int ok=u.ControlFecha(fechamuerte);
        if (ok==1) //Formato de fecha incorrecto.
        {
            valores.append("\n\nERROR:\nFormato de la fecha introducida
incorrecto.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==2) //Fecha introducida inexistente.
        {
            valores.append("\n\nERROR:\nFecha introducida inexistente.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==3) //Error de los limites de fecha.
        {
            valores.append("\n\nERROR:\nFecha introducida fuera de limites.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
    }

    if (horamuerte.length()==0)
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    else
    {
        int ok=u.ControlHora(horamuerte);
        if (ok==1) //Formato de hora incorrecto.
        {
            valores.append("\n\nERROR:\nFormato de hora introducida incorrecto.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==2) //Error en la hora introducida.
        {
            valores.append("\n\nERROR:\nHora introducida inexistente.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
    }
    return(true);
}
}

```

### 8.2.2.3 MenuAnotarOrganoRechazado.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Implementacion de menu para introducir datos relativos al rechazo
de un organo por un paciente receptor desde interficie grafica asociada a
AgentCTH.

*****/

package Agent0.agentes.CTH.interficie;

//Importamos la clase que implementa al AgentCTH.
import Agent0.agentes.CTH.*;
import Agent0.agentes.CTH.Ontology.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

/*****
Clase que muestra y controla los componentes del submenu
que permiten al usuario introducir los datos
correspondientes al rechazo de un organo por parte de
un paciente.

Se crea un objeto de esta clase cuando el usuario elige
la opcion anotar organo rechazado en el ComboBox del menu
principal.

Tiene la misma estructura de programacion que la del menu
principal.
*****/
public class MenuAnotarOrganoRechazado extends JFrame{

    //Para recibir instancia por parametro, en el constructor de la clase,
    //de AgentCTH y poder enviarle eventos.
    private AgentCTH agentcth;

    //Clase que contiene algunos metodos utiles en diferentes archivos de
    //la interficie.
    Utilidades u= new Utilidades();

    //Objeto definido en la ontologia para guardar valores introducidos por
    //usuario mientras navega por la interficie grafica.
    Hospital hospital;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Anotar òrgano Rechazado por Paciente.";

    Container contenedorprincipal;

    //Variables globales
    String idorgano;

    //Componentes Swing
    AbstractBorder borde = new TitledBorder("Dato identificador de òrgano.");

    JPanel panel = new JPanel();

    JLabel e_idorgano = new JLabel ("Identificador òrgano: ");

```

```

    JTextField t_idorgano = new JTextField();

    JButton anterior = new JButton ("Anterior");
    JButton siguiente = new JButton ("Siguiente");
    JButton salir = new JButton ("Salir");

    //Para recibir mensajes de salida del sistema.
    private JScrollPane PanelSalida;
    private JTextArea valores;

    //Para salir de la aplicacion, cuando se cierra la ventana activa de la
    //interficie.
    WindowListener l= new WindowAdapter(){
        public void windowClosing (WindowEvent e){
            System.exit(0);
        }
    };

/*****
Constructor de la clase que implementa esta ventana en la interficie
grafica.
*****/
    public MenuAnotarOrganoRechazado(AgentCTH a,JScrollPane panelsalida,
    JTextArea salida){
        super(TITLE);
        setSize(WIDTH,HEIGHT);

        setResizable(false);

        agentcth = a;
        valores = salida;
        valores.setText(""); //Poniendo esto (salida) es inutil,
                            //pero lo dejo por si se cambia de criterio.

        PanelSalida=panelsalida;

        elementos();
        inieventos();
        show();
    }

/*****
Metodo que describe elementos que contendra esta ventana y su posicion.
*****/
    void elementos(){
        contenedorprincipal = getContentPane();
        contenedorprincipal.setLayout(null);

        panel.setLayout(null);

        e_idorgano.setBounds(10,30,140,20);

        t_idorgano.setBounds(150,30,100,20);

        anterior.setBounds(20,230,100,20);
        siguiente.setBounds(150,230,100,20);
        salir.setBounds(310,230,100,20);

        panel.add(e_idorgano);
        panel.add(t_idorgano);
        panel.setBorder(borde);
        panel.setBounds(10,10,400,75);

        contenedorprincipal.add(panel);

        contenedorprincipal.add(anterior);
        contenedorprincipal.add(siguiente);
        contenedorprincipal.add(salir);

        //Para mostrar mensajes de salida del sistema.

```

```

        valores.setToolTipText ("Mensajes de salida procedentes del sistema");
        valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
        PanelSalida.setViewportView(valores);
        PanelSalida.setBounds(10,260,400,150);

        contenedorprincipal.add(PanelSalida);
    }

/*****
Metodo que inicializa eventos asociados a algunos componentes de la
ventana.
*****/
void inieventos(){
    //Para eventos producidos sobre el boton anterior.
    //Para poder pulsar boton con tecla.
    anterior.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Anterior();
        }
    });

    anterior.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Anterior();
        }
    });

    //Para eventos producidos sobre el boton siguiente.
    //Para poder pulsar boton con tecla.
    siguiente.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiente();
        }
    });

    siguiente.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiente();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });
}

/*****
Metodo que se ejecuta cuando usuario oprime el boton 'Anterior'.
*****/
private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op = new MenuOpciones(agentcth,PanelSalida,valores);

    op.hospital=hospital;

    op.addWindowListener(l);
}

```



```

        dispose();
    }

/*****
Metodo que se ejecuta cuando usuario oprime el boton 'Siguiete'.
*****/
private void Ejecutar_Siguiete()
{
    idorgano=" "+t_idorgano.getText();
    if (ControlErrores()==true)
    {
        //Para controlar que no se introducen espacios en blanco.
        idorgano=u.StringSinEspaciosEnBlanco(idorgano);

        //Envia evento a la clase AgentCTH y correspondiente objeto
        //de ontologia para realizar accion.
        OrganRebutxat OR = new OrganRebutxat(idorgano,hospital);
        agentcth.postGuiEvent(new
jade.gui.GuiEvent(OR,agentcth.ANOTARORGANORECHAZADO));
    }
}

/*****
Metodo valida datos introducidos por usuario.
*****/
boolean ControlErrores()
{
    if (t_idorgano.getText().equals(""))
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return (false);
    }

    if ((t_idorgano.getText()).length(>10)
    {
        valores.append("\n\nERROR:\n"+
        "El identificador de organo no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }
    return(true);
}
}

```

#### 8.2.2.4 MenuDonante.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

MenuDonante para introducir datos relativos a un paciente donante desde
interficie grafica asociada a AgentCTH.

*****/
package Agent0.agentes.CTH.interficie;

//Importamos clase que implementa AgentCTH.
import Agent0.agentes.CTH.*;
//Importamos clase que implementa ontologia.
import Agent0.agentes.CTH.Ontology.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import java.util.*;

/*****
Clase que implementa la ventana MenuDonante de la interficie grafica.
*****/
public class MenuDonante extends JFrame{

    //Objeto que sirve para poder enviar eventos a AgentCTH desde interficie
    //grafica.
    private AgentCTH agentcth;

    //Clase que contiene metodos utiles en diferentes clases que implementan
    //la interficie grafica.
    private Utilidades u = new Utilidades();

    //Objetos de la ontologia que contendran valores introducidos por usuario.
    Patient patient;
    Organ organ;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Datos relativos al organo donado.";

    Container contenedorprincipal;

    //Variables globales
    String idorgano;
    String tipoorgano;
    String tamañoorgano;
    String s_estado;

    String tipos_organos []={"Corazon","Higado","Rinyon","Pulmon"};
    String estado[] = {"Vivo","Muerto"};

    //Componentes Swing
    AbstractBorder borde = new TitledBorder("Datos relativos al organo
donado.");

    JPanel panel = new JPanel();

    JLabel e_idorgano = new JLabel ("Identificador àrgano: ");
    JLabel e_tipoorgano = new JLabel ("Tipo àrgano: ");
    JLabel e_tamañoorgano = new JLabel ("Tamaño àrgano (cms.): ");
    JLabel e_estado = new JLabel ("Estado paciente: ");

```

```

JComboBox t_tipoorgano = new JComboBox(tipos_organos);
JComboBox seleccion_estado = new JComboBox(estado);

JTextField t_idorgano = new JTextField();
JTextField t_tamapooorgano = new JTextField();

JButton anterior = new JButton ("Anterior");
JButton siguiente = new JButton ("Siguiente");
JButton salir = new JButton ("Salir");

//Para mostrar mensajes de salida del sistema.
private JScrollPane PanelSalida;
private JTextArea valores;

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interficie.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor de la clase MenuDonante.
*****/
public MenuDonante(AgentCTH a, JScrollPane panelsalida, JTextArea salida){
    super(TITLE);
    setSize(WIDTH, HEIGHT);
    setResizable(false);

    agentcth = a;
    valores = salida;
    valores.setText(""); //Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.

    PanelSalida=panelsalida;

    elementos();
    inieventos();
    show();
}

/*****
Metodo que especifica que elementos mostrara el menu y su posicion.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_idorgano.setBounds(10, 30, 140, 20);
    e_tipoorgano.setBounds(10, 60, 85, 20);
    e_tamapooorgano.setBounds(10, 90, 140, 20);

    t_idorgano.setBounds(160, 30, 75, 20);
    t_tipoorgano.setBounds(105, 60, 75, 20);
    t_tamapooorgano.setBounds(155, 90, 50, 20);

    e_estado.setBounds(10, 120, 100, 20);
    seleccion_estado.setBounds(120, 120, 100, 20);

    anterior.setBounds(20, 230, 100, 20);
    siguiente.setBounds(150, 230, 100, 20);
    salir.setBounds(310, 230, 100, 20);

    panel.add(e_idorgano);
    panel.add(e_tipoorgano);
    panel.add(e_tamapooorgano);
    panel.add(e_estado);

```

```

panel.add(t_idorgano);
panel.add(t_tipoorgano);
panel.add(t_tamanoorgano);

panel.add(seleccion_estado);

panel.setBorder(borde);

panel.setBounds(10,10,400,150);

contenedorprincipal.add(panel);
contenedorprincipal.add(anterior);
contenedorprincipal.add(siguiete);
contenedorprincipal.add(salir);

//Para mostrar mensajes de salida del sistema.
valores.setToolTipText ("Mensajes de salida procedentes del sistema");
valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
PanelSalida.setViewPortView(valores);
PanelSalida.setBounds(10,260,400,150);

contenedorprincipal.add(PanelSalida);

}

/*****
Metodo que inicia los eventos asociados a componentes del menu.
*****/
void inieventos(){

//Para eventos producidos sobre el boton anterior.
//Para poder pulsar boton con tecla.
anterior.addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        //Si se ha pulsado Return.
        if (e.getKeyCode()==10) Ejecutar_Anterior();
    }
});

anterior.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e){
        Ejecutar_Anterior();
    }
});

//Para eventos producidos sobre el boton siguiete.
//Para poder pulsar boton con tecla.
siguiete.addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        //Si se ha pulsado Return.
        if (e.getKeyCode()==10) Ejecutar_Siguiete();
    }
});

siguiete.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e){
        Ejecutar_Siguiete();
    }
});

//Para eventos producidos sobre el boton salir.
salir.addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        //Si se ha pulsado Return.
        if (e.getKeyCode()==10) System.exit(0);
    }
});

salir.addMouseListener(new MouseAdapter() {

```

```

        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });
}

/*****
Metodo que se ejecuta cuando usuario oprime el boton 'Anterior'.
*****/
private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op = new MenuOpciones(agentctth, PanelSalida, valores);

    //Datos que se han de guardar mientras se navega por interficie.
    op.hospital=pacient.getHospitalIngres();

    op.addWindowListener(l);

    dispose();
}

/*****
Metodo que se ejecuta cuando usuario oprime el boton 'Siguiete'.
*****/
private void Ejecutar_Siguiete()
{
    tipoorgano=t_tipoorgano.getSelectedItem().toString();
    idorgano=" "+t_idorgano.getText();
    tamaoorgano=t_tamaoorgano.getText();

    s_estado=seleccion_estado.getSelectedItem().toString();

    if (ControlErrores()==true)
    { //Para controlar que no se introducen caracteres en blanco.
        idorgano=u.StringSinEspaciosEnBlanco(idorgano);
        s_estado=u.StringSinEspaciosEnBlanco(s_estado);

        Double tamany=new Double(tamaoorgano);

        //Guardamos valores introducidos por usuario, en objeto
        //definido en la ontologia.
        organ=new Organ(idorgano,tipoorgano,tamany,"Donat");

        //Actualitzem dades de entrada al sistema.
        GregorianCalendar c = new GregorianCalendar();

        Long diaentradasistema=new Long(c.get(Calendar.DATE));
        //Hay que aadir +1, porque los meses se cuentan desde 0.
        Long mesentradasistema=new Long(c.get(Calendar.MONTH)+1);
        Long anyentradasistema=new Long(c.get(Calendar.YEAR));
        int hesistema=c.get(Calendar.HOUR);
        if (c.get(Calendar.AM_PM)==1) hesistema+=12;
        Long horaentradasistema=new Long(hesistema);
        Long minutentradasistema=new Long(c.get(Calendar.MINUTE));

        pacient.setOrgano(organ);
        pacient.setEstat(s_estado);
        pacient.setDiaEntradaSistema(diaentradasistema);
        pacient.setMesEntradaSistema(mesentradasistema);
        pacient.setAnyEntradaSistema(anyentradasistema);
        pacient.setHoraEntradaSistema(horaentradasistema);
        pacient.setMinutEntradaSistema(minutentradasistema);

        if (seleccion_estado.getSelectedItem().equals("Muerto"))
        {
            //Se pasa Objetos creados desde AgentCTH para comunicacion
            //entre AgentCTH e interficie grafica.

```

```

        MenuMuertePaciente mmp = new
        MenuMuertePaciente(agentctch, PanelSalida, valores);

        //Se pasa objeto de la ontologia con valores introducidos por
        //usuario a la clase que implementa la ventana que se va a abrir.
        mmp.pacient=pacient;

        mmp.addWindowListener(l);

        dispose();
    } else agentctch.postGuiEvent(new
jade.gui.GuiEvent(pacient, agentctch.INTRODUCIRPACIENTE));
    }
}

/*****
Metodo valida datos introducidos por usuario.
*****/
boolean ControlErrores()
{
    double t;

    if ((t_idorgano.getText()).equals(""))
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    if ((t_idorgano.getText()).length(>10)
    {
        valores.append("\n\nERROR:\n"+
        "El identificador de organo no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }
    if (tamaoorgano.length()==0)
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    if (u.ControlDouble(tamaoorgano)==false)
    {
        valores.append("\n\nERROR:\nEl tamaño del organo ha de ser un numero.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }else t = new Double(tamaoorgano).doubleValue();

    if ((t >100.0) || (t <0.0))
    {
        valores.append("\n\nERROR:\nEl tamaño de organo ha de estar entre 0 y
100");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }
    return(true);
}

}

```

### 8.2.2.5 MenuEncontrarPacientesCompatibles.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

MenuDonante para introducir datos relativos a un paciente donante desde
interficie grafica asociada a AgentCTH.

*****/
package Agent0.agentes.CTH.interficie;

//Importamos clase que implementa AgentCTH.
import Agent0.agentes.CTH.*;
//Importamos clase que implementa ontologia.
import Agent0.agentes.CTH.Ontology.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import java.util.*;

/*****
Clase que implementa la ventana MenuDonante de la interficie grafica.
*****/
public class MenuDonante extends JFrame{

    //Objeto que sirve para poder enviar eventos a AgentCTH desde interficie
    //grafica.
    private AgentCTH agentcth;

    //Clase que contiene metodos utiles en diferentes clases que implementan
    //la interficie grafica.
    private Utilidades u = new Utilidades();

    //Objetos de la ontologia que contendran valores introducidos por usuario.
    Patient patient;
    Organ organ;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Datos relativos al organo donado.";

    Container contenedorprincipal;

    //Variables globales
    String idorgano;
    String tipoorgano;
    String tamañoorgano;
    String s_estado;

    String tipos_organos []={"Corazon","Higado","Rinyon","Pulmon"};
    String estado[] = {"Vivo","Muerto"};

    //Componentes Swing
    AbstractBorder borde = new TitledBorder("Datos relativos al organo
donado.");

    JPanel panel = new JPanel();

    JLabel e_idorgano = new JLabel ("Identificador ãrgano: ");
    JLabel e_tipoorgano = new JLabel ("Tipo ãrgano: ");
    JLabel e_tamañoorgano = new JLabel ("Tamaño ãrgano (cms.): ");
    JLabel e_estado = new JLabel ("Estado paciente: ");

```

```

JComboBox t_tipoorgano = new JComboBox(tipos_organos);
JComboBox seleccion_estado = new JComboBox(estados);

JTextField t_idorgano = new JTextField();
JTextField t_tamapooorgano = new JTextField();

JButton anterior = new JButton ("Anterior");
JButton siguiente = new JButton ("Siguiente");
JButton salir = new JButton ("Salir");

//Para mostrar mensajes de salida del sistema.
private JScrollPane PanelSalida;
private JTextArea valores;

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interficie.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor de la clase MenuDonante.
*****/
public MenuDonante(AgentCTH a, JScrollPane panelsalida, JTextArea salida){
    super(TITLE);
    setSize(WIDTH, HEIGHT);
    setResizable(false);

    agentcth = a;
    valores = salida;
    valores.setText(""); //Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.

    PanelSalida=panelsalida;

    elementos();
    inieventos();
    show();
}

/*****
Metodo que especifica que elementos mostrara el menu y su posicion.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_idorgano.setBounds(10, 30, 140, 20);
    e_tipoorgano.setBounds(10, 60, 85, 20);
    e_tamapooorgano.setBounds(10, 90, 140, 20);

    t_idorgano.setBounds(160, 30, 75, 20);
    t_tipoorgano.setBounds(105, 60, 75, 20);
    t_tamapooorgano.setBounds(155, 90, 50, 20);

    e_estado.setBounds(10, 120, 100, 20);
    seleccion_estado.setBounds(120, 120, 100, 20);

    anterior.setBounds(20, 230, 100, 20);
    siguiente.setBounds(150, 230, 100, 20);
    salir.setBounds(310, 230, 100, 20);

    panel.add(e_idorgano);
    panel.add(e_tipoorgano);
    panel.add(e_tamapooorgano);
    panel.add(e_estado);

```



```

panel.add(t_idorgano);
panel.add(t_tipoorgano);
panel.add(t_tamanoorgano);

panel.add(seleccion_estado);

panel.setBorder(borde);

panel.setBounds(10,10,400,150);

contenedorprincipal.add(panel);
contenedorprincipal.add(anterior);
contenedorprincipal.add(siguiete);
contenedorprincipal.add(salir);

//Para mostrar mensajes de salida del sistema.
valores.setToolTipText ("Mensajes de salida procedentes del sistema");
valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
PanelSalida.setViewPortView(valores);
PanelSalida.setBounds(10,260,400,150);

contenedorprincipal.add(PanelSalida);

}

/*****
Metodo que inicia los eventos asociados a componentes del menu.
*****/
void inieventos(){

//Para eventos producidos sobre el boton anterior.
//Para poder pulsar boton con tecla.
anterior.addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        //Si se ha pulsado Return.
        if (e.getKeyCode()==10) Ejecutar_Anterior();
    }
});

anterior.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e){
        Ejecutar_Anterior();
    }
});

//Para eventos producidos sobre el boton siguiete.
//Para poder pulsar boton con tecla.
siguiete.addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        //Si se ha pulsado Return.
        if (e.getKeyCode()==10) Ejecutar_Siguiete();
    }
});

siguiete.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e){
        Ejecutar_Siguiete();
    }
});

//Para eventos producidos sobre el boton salir.
salir.addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        //Si se ha pulsado Return.
        if (e.getKeyCode()==10) System.exit(0);
    }
});

salir.addMouseListener(new MouseAdapter() {

```

```

        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });
}

/*****
Metodo que se ejecuta cuando usuario oprime el boton 'Anterior'.
*****/
private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op = new MenuOpciones(agentctth, PanelSalida, valores);

    //Datos que se han de guardar mientras se navega por interficie.
    op.hospital=pacient.getHospitalIngres();

    op.addWindowListener(l);

    dispose();
}

/*****
Metodo que se ejecuta cuando usuario oprime el boton 'Siguiete'.
*****/
private void Ejecutar_Siguiete()
{
    tipoorgano=t_tipoorgano.getSelectedItem().toString();
    idorgano=" "+t_idorgano.getText();
    tamaoorgano=t_tamaoorgano.getText();

    s_estado=seleccion_estado.getSelectedItem().toString();

    if (ControlErrores()==true)
    { //Para controlar que no se introducen caracteres en blanco.
        idorgano=u.StringSinEspaciosEnBlanco(idorgano);
        s_estado=u.StringSinEspaciosEnBlanco(s_estado);

        Double tamany=new Double(tamaoorgano);

        //Guardamos valores introducidos por usuario, en objeto
        //definido en la ontologia.
        organ=new Organ(idorgano,tipoorgano,tamany,"Donat");

        //Actualitzem dades de entrada al sistema.
        GregorianCalendar c = new GregorianCalendar();

        Long diaentradasistema=new Long(c.get(Calendar.DATE));
        //Hay que aadir +1, porque los meses se cuentan desde 0.
        Long mesentradasistema=new Long(c.get(Calendar.MONTH)+1);
        Long anyentradasistema=new Long(c.get(Calendar.YEAR));
        int hesistema=c.get(Calendar.HOUR);
        if (c.get(Calendar.AM_PM)==1) hesistema+=12;
        Long horaentradasistema=new Long(hesistema);
        Long minutentradasistema=new Long(c.get(Calendar.MINUTE));

        pacient.setOrgano(organ);
        pacient.setEstat(s_estado);
        pacient.setDiaEntradaSistema(diaentradasistema);
        pacient.setMesEntradaSistema(mesentradasistema);
        pacient.setAnyEntradaSistema(anyentradasistema);
        pacient.setHoraEntradaSistema(horaentradasistema);
        pacient.setMinutEntradaSistema(minutentradasistema);

        if (seleccion_estado.getSelectedItem().equals("Muerto"))
        {
            //Se pasa Objetos creados desde AgentCTH para comunicacion
            //entre AgentCTH e interficie grafica.

```

```

        MenuMuertePaciente mmp = new
        MenuMuertePaciente(agentctch, PanelSalida, valores);

        //Se pasa objeto de la ontologia con valores introducidos por
        //usuario a la clase que implementa la ventana que se va a abrir.
        mmp.pacient=pacient;

        mmp.addWindowListener(l);

        dispose();
    } else agentctch.postGuiEvent(new
jade.gui.GuiEvent(pacient, agentctch.INTRODUCIRPACIENTE));
    }
}

/*****
Metodo valida datos introducidos por usuario.
*****/
boolean ControlErrores()
{
    double t;

    if ((t_idorgano.getText()).equals(""))
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    if ((t_idorgano.getText()).length()>10)
    {
        valores.append("\n\nERROR:\n"+
        "El identificador de organo no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }
    if (tamaoorgano.length()==0)
    {
        valores.append("\n\nERROR:\nEs obligatorio rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    if (u.ControlDouble(tamaoorgano)==false)
    {
        valores.append("\n\nERROR:\nEl tamaño del organo ha de ser un numero.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }else t = new Double(tamaoorgano).doubleValue();

    if ((t >100.0) || (t <0.0))
    {
        valores.append("\n\nERROR:\nEl tamaño de organo ha de estar entre 0 y
100");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }
    return(true);
}
}

```

### 8.2.2.6 MenuIntroducirPaciente.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Menu introducir paciente de la interficie utilizada por el agente AgentCTH.
En este menu, es el menu inicial para poder introducir datos de usuario.

*****/
package Agent0.agentes.CTH.interficie;

import java.lang.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

import java.util.*;

import Agent0.agentes.CTH.*;
import Agent0.agentes.CTH.Ontology.*;

import jade.gui.*;

/*****
Clase que muestra y maneja los componentes del submenu
que permiten introducir los datos de un paciente que se
da de alta en el sistema, ya sea un donante o un receptor
de organo. Esta clase permite crear objetos de dos clases
que corresponden a los submenus que permiten introducir los
datos relativos a la muerte de un paciente, y el otro a los
relativos al organo donado o que se espera recibir por el
paciente.
*****/
public class MenuIntroducirPaciente extends JFrame{

    //Objeto que permite enviar eventos al AgentCTH correspondiente.
    private AgentCTH agentcth;

    //Objectos que contendran valores introducidos por usuario
    //mientras navega por interficie grafica.
    Hospital hospital;
    Patient patient;
    DadesCliniques dadescliniques;

    //Objeto de la clase utilidades, utilizada para no replicar codigo.
    //Con la instanciación a esta clase, se puede tener acceso a metodos
    //que seran utilizados en diferentes clases.
    Utilidades u=new Utilidades();

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Menú Principal para Introducir Paciente";

    String sexo[] = {"Hombre","Mujer"};

    String tiposangre[] = {"O","A","B","AB"};

    String donante[]= {"Donante","Receptor"};

    Container contenedorprincipal;

```

```

//Variables globales
String s_sexo;
String s_tiposangre;
String s_donante;
String codigoss;
String nombre;
String dianacimiento;
String altura;
String peso;
String antigenos;

//Componentes Swing
AbstractBorder borde = new TitledBorder("Datos del paciente que se quiere
introducir.");

JPanel panel = new JPanel();

JLabel e_codigoss = new JLabel ("Código SS: ");
JLabel e_nombre = new JLabel ("Nombre y apellidos: ");
JLabel e_dianacimiento = new JLabel ("Día Nac. (dd/mm/aaaa):");
JLabel e_altura = new JLabel ("Altura (cms.): ");
JLabel e_peso= new JLabel ("Peso (Kg.): ");
JLabel e_antigenos = new JLabel ("Antígenos (4 Bool. B0,B1,Dr0,Dr1):");
JLabel e_sexo = new JLabel ("Sexo: ");
JLabel e_gruposangre = new JLabel ("Grupo sanguíneo: ");
JLabel e_donante = new JLabel ("Tipo paciente: ");

JTextField t_codigoss = new JTextField();
JTextField t_nombre = new JTextField();
JTextField t_dianacimiento = new JTextField();
JTextField t_altura = new JTextField();
JTextField t_peso = new JTextField();
JTextField t_antigenos = new JTextField();

JComboBox seleccion_sexo = new JComboBox(sexo);
JComboBox seleccion_tiposangre = new JComboBox(tiposangre);
JComboBox seleccion_donante = new JComboBox(donante);

JButton anterior = new JButton ("Anterior");
JButton siguiente = new JButton ("Siguiente");
JButton salir = new JButton ("Salir");

//Objetos mostrar mensajes de salida del sistema.
private JScrollPane PanelSalida;
private JTextArea valores;

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interficie.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor del MenuIntroducirPaciente.
*****/
public MenuIntroducirPaciente(AgentCTH a,JScrollPane panelsalida,JTextArea
salida){
    super(TITLE);
    setSize(WIDTH,HEIGHT);
    setResizable(false);

    agentcth = a;
    valores = salida;
    valores.setText(""); //Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.

    PanelSalida=panelsalida;

```

```

    elementos();
    inieventos();
    show();
}

/*****
Especificacion de elementos que apareceran en este menu y su situacion.
*****/

void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_codigoss.setBounds(10,20,70,20);
    e_nombre.setBounds(10,50,120,20);
    e_dianacimiento.setBounds(10,80,150,20);

    e_sexo.setBounds(190,20,40,20);
    seleccion_sexo.setBounds(240,20,75,20);

    e_altura.setBounds(245,80,85,20);
    e_peso.setBounds(10,110,75,20);

    e_gruposangre.setBounds(10,140,110,20);
    seleccion_tiposangre.setBounds(130,140,50,20);

    e_antigenos.setBounds(140,110,200,20);

    e_donante.setBounds(200,140,100,20);
    seleccion_donante.setBounds(310,140,85,20);

    t_codigoss.setBounds(75,20,100,20);
    t_nombre.setBounds(140,50,250,20);
    t_dianacimiento.setBounds(160,80,75,20);
    t_altura.setBounds(340,80,40,20);
    t_peso.setBounds(90,110,40,20);
    t_antigenos.setBounds(350,110,40,20);

    anterior.setBounds(20,230,100,20);
    siguiente.setBounds(150,230,100,20);
    salir.setBounds(310,230,100,20);

    panel.add(e_codigoss);
    panel.add(e_nombre);
    panel.add(e_dianacimiento);
    panel.add(e_altura);
    panel.add(e_peso);
    panel.add(e_antigenos);

    panel.add(e_sexo);
    panel.add(e_gruposangre);
    panel.add(e_donante);

    panel.add(t_codigoss);
    panel.add(t_nombre);
    panel.add(t_dianacimiento);
    panel.add(t_altura);
    panel.add(t_peso);
    panel.add(t_antigenos);

    panel.add(seleccion_sexo);
    panel.add(seleccion_tiposangre);
    panel.add(seleccion_donante);

    panel.setBorder(borde);

    panel.setBounds(10,10,400,200);

```

```

        contenedorprincipal.add(panel);

        contenedorprincipal.add(anterior);
        contenedorprincipal.add(siguiete);
        contenedorprincipal.add(salir);

        //Para mostrar mensajes de salida del sistema.
        valores.setToolTipText ("Mensajes de salida procedentes del sistema");
        valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
        PanelSalida.setViewportView(valores);
        PanelSalida.setBounds(10,260,400,150);
        contenedorprincipal.add(PanelSalida);
    }

    /*****
    Inicializa los eventos producidos sobre los diferentes componentes Swing.
    *****/
    void inieventos(){

        //Para eventos producidos sobre el boton anterior.
        //Para poder pulsar boton con tecla.
        anterior.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {

                //Si se ha pulsado Return.
                if (e.getKeyCode()==10) Ejecutar_Anterior();
            }
        });

        anterior.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e){
                Ejecutar_Anterior();
            }
        });

        //Para eventos producidos sobre el boton siguiete.
        //Para poder pulsar boton con tecla.
        siguiete.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                //Si se ha pulsado Return.
                if (e.getKeyCode()==10) Ejecutar_Siguiete();
            }
        });

        siguiete.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e){
                Ejecutar_Siguiete();
            }
        });

        //Para eventos producidos sobre el boton salir.
        salir.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                //Si se ha pulsado Return.
                if (e.getKeyCode()==10) System.exit(0);
            }
        });

        salir.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e){
                System.exit(0);
            }
        });

    }

    /*****
    Metodo que se ejecutara cuando usuario oprima el boton 'Anterior'.
    *****/

```

```

private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op =new MenuOpciones(agentcth,PanelSalida,valores);

    //Para no perder valores introducidos por usuario, mientras
    //navega por interficie grafica.
    op.hospital=hospital;

    op.addWindowListener(l);

    dispose();
}

/*****
Metodo que se ejecutara cuando usuario oprima el boton 'Siguiete'.
*****/
private void Ejecutar_Siguiete()
{
    codigoss=" "+t_codigoss.getText();
    nombre=" "+t_nombre.getText();
    dianacimiento=t_dianacimiento.getText();
    s_sexo=seleccion_sexo.getSelectedItem().toString();
    altura=t_altura.getText();
    peso=t_peso.getText();
    s_tiposangre=seleccion_tiposangre.getSelectedItem().toString();
    antigenos=t_antigenos.getText();
    s_donante=seleccion_donante.getSelectedItem().toString();

    if(ControlErrores()==true)
    { //Para controlar que no se introduzcan caracteres en blanco.
        codigoss=u.StringSinEspaciosEnBlanco(codigoss);
        nombre=u.StringSinEspaciosEnBlanco(nombre);

        //Para obtener fecha y hora del sistema.
        GregorianCalendar c = new GregorianCalendar();
        Long diaentradasistema=new Long(c.get(Calendar.DATE));
        //Hay que añadir +1, porque los meses se cuentan desde 0.
        Long mesentradasistema=new Long(c.get(Calendar.MONTH)+1);
        Long anyentradasistema=new Long(c.get(Calendar.YEAR));
        int hesistema=c.get(Calendar.HOUR);
        if (c.get(Calendar.AM_PM)==1) hesistema+=12;
        Long horaentradasistema=new Long(hesistema);
        Long minutentradasistema=new Long(c.get(Calendar.MINUTE));

        //Para pasar fecha de nacimiento de String a 3 variables Long.
        char f[] = dianacimiento.toCharArray();
        Long dnac=new
        Long(Character.digit(f[0],10)*10+Character.digit(f[1],10));
        Long mnac=new
        Long(Character.digit(f[3],10)*10+Character.digit(f[4],10));
        Long anac=new
        Long(Character.digit(f[6],10)*1000+Character.digit(f[7],10)*100+Character.digit(f[8],10)*10+Character.digit(f[9],10));

        Double alz = new Double(altura);
        Double pes = new Double(peso);

        //Para pasar antigenos de String a 4 valores Long que seran binarios.
        char ant[]=antigenos.toCharArray();
        Long b0 = new Long(Character.digit(ant[0],10));
        Long b1 = new Long(Character.digit(ant[1],10));
        Long dr0 = new Long(Character.digit(ant[2],10));
        Long dr1 = new Long(Character.digit(ant[3],10));

        //Para guardar valores introducidos por usuario, en objetos definidos
        //en ontologia.
        dadescliniques=new DadesCliniques(dnac,mnac,anac,s_sexo,alz,pes,
        s_tiposangre,b0,b1,dr0,dr1);
    }
}

```



```

    pacient= new Pacient(codigoss,nombre,diaentradasistema,
    mesentradasistema,anyentradasistema,horaentradasistema,
    minutentradasistema,null,null,hospital,dadescliniques);

    if (seleccion_donante.getSelectedItemAt().equals("Donante"))
    {
        //Se pasa Objetos creados en AgentCTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuDonante d = new MenuDonante(agentctth,PanelSalida,valores);

        //Se pasa objeto pacient definido en ontologia con valores
        //introducidos por usuario.
        d.pacient=pacient;

        d.addWindowListener(l);

        dispose();
    }
    else
    {
        //Se pasa Objetos creados en AgentCTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuReceptor r = new MenuReceptor(agentctth,PanelSalida,valores);

        //Se pasa el objeto Pacient definido en ontologia, con valores
        //introducidos por usuario.
        r.pacient=pacient;

        r.addWindowListener(l);

        dispose();
    }
}
}

/*****
Metodo que valida los datos introducidos por usuario.
*****/
boolean ControlErrores()
{
    double alt,pe;

    if ((t_codigoss.getText()).equals(""))
    {
        valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }

    if ((t_codigoss.getText()).length(>10)
    {
        valores.append("\n\nERROR:\n"+
        "El codigo de seguridad social no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    if ((t_nombre.getText()).equals(""))
    {
        valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }

    if ((t_nombre.getText()).length(>70)
    {
        valores.append("\n\nERROR:\n"+

```

```

        "El nombre y apellidos del paciente no puede tener mas de 70
caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    if (dianacimiento.length()==0)
    {
        valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    else
    {
        int ok=u.ControlFecha(dianacimiento);
        if (ok==1) //Error en formato de fecha.
        {
            valores.append("\n\nERROR:\nFormato de fecha introducida
incorrecto.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==2) //Error de fecha inexistente.
        {
            valores.append("\n\nERROR:\nFecha introducida inexistente.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==3) //Error de los limites de fecha.
        {
            valores.append("\n\nERROR:\nFecha introducida fuera de limites.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
    }
}

if (altura.length()==0)
{
    valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return(false);
}

if (u.ControlDouble(altura)==false)
{
    valores.append("\n\nERROR:\nEl dato altura debe de ser un numero.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return (false);
} alt=new Double(altura).doubleValue();

if ((alt<0) || (alt>249))
{
    valores.append("\n\nERROR:\nEl dato altura debe estar entre 0 y 249.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return (false);
}

if (peso.length()==0)
{
    valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return(false);
}

if (u.ControlDouble(peso)==false)
{
    valores.append("\n\nERROR:\nEl dato peso debe de ser un numero.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return (false);
} pe=new Double(peso).doubleValue();

```

```

if ((pe<0) || (pe>399))
{
    valores.append("\n\nERROR:\nEl dato peso debe estar entre 0 y 399.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return (false);
}

if (antigenos.length()==0)
{
    valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return(false);
}

if (!(u.ControlAntigenos(antigenos)))
{
    valores.append("\n\nERROR:\nFormato de datos antigenos incorrecto.");
    valores.append("\nHa de ser un numero de 4 cifras binarias (0 o 1).");
    u.PosicionarFinalScrollBar (PanelSalida);
    return(false);
}
return(true);
}
}

```

### 8.2.2.7 MenuMuertePaciente.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

MenuMuertePaciente de la interficie utilizada por el agente AgentCTH.
En este menu se introducen los datos relativos a la muerte de un paciente.

*****/
package Agent0.agentes.CTH.interficie;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

import Agent0.agentes.CTH.*;
import Agent0.agentes.CTH.Ontology.*;

/*****
Clase que implementa la ventana MenuMuertePaciente de la interficie
grafica.
*****/
public class MenuMuertePaciente extends JFrame{
    //Objeto que permite identificar al objeto al que enviar eventos desde
    //la interficie grafica.
    private AgentCTH agentcth;

    //Clase con metodos utiles en diferentes archivos de la interficie grafica.
    private Utilidades u=new Utilidades();

    //Objeto definido en ontologia que contendra valores introducidos por
    //usuario en interficie.
    Pacient pacient;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Anotar Muerte del Paciente.";

    Container contenedorprincipal;

    //Variables globales
    String estadoprevio;
    String fechamuerte;
    String horamuerte;

    //Componentes Swing

    AbstractBorder borde = new TitledBorder("Datos sobre la muerte del
paciente.");

    JPanel panel = new JPanel();

    JLabel e_fechamuerte = new JLabel ("Día Muerte (dd/mm/aaaa):");
    JLabel e_horamuerte = new JLabel ("Hora Muerte (HH:MM):");

    JTextField t_fechamuerte = new JTextField();
    JTextField t_horamuerte = new JTextField();

    JButton anterior = new JButton ("Anterior");
    JButton siguiente = new JButton ("Siguiente");
    JButton salir = new JButton ("Salir");

    //Para mostrar mensajes de salida del sistema. Los objetos los recibe por
    //parametro a la hora de instanciar a esta clase.

```

```

private JScrollPane PanelSalida;
private JTextArea valores;

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interficie.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor de la clase.
*****/
public MenuMuertePaciente(AgentCTH a,JScrollPane panelsalida,
JTextArea salida){
    super(TITLE);
    setSize(WIDTH,HEIGHT);
    setResizable(false);

    agentcth = a;
    valores = salida;
    valores.setText("");//Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.
    PanelSalida=panelsalida;

    elementos();
    inieventos();
    show();
}

/*****
Especifica que elementos contendra la ventana y su localizacion.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_fechamuerte.setBounds(10,30,200,20);
    e_horamuerte.setBounds(10,60,200,20);

    t_fechamuerte.setBounds(170,30,75,20);
    t_horamuerte.setBounds(150,60,50,20);

    anterior.setBounds(20,230,100,20);
    siguiente.setBounds(150,230,100,20);
    salir.setBounds(310,230,100,20);

    panel.add(e_fechamuerte);
    panel.add(e_horamuerte);

    panel.add(t_fechamuerte);
    panel.add(t_horamuerte);

    panel.setBorder(borde);

    panel.setBounds(10,10,400,100);

    contenedorprincipal.add(panel);

    contenedorprincipal.add(anterior);
    contenedorprincipal.add(siguiente);
    contenedorprincipal.add(salir);

    //Para mostrar mensajes de salida del sistema.
    valores.setToolTipText ("Mensajes de salida procedentes del sistema");
    valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));

```

```

        PanelSalida.setViewportView(valores);
        PanelSalida.setBounds(10,260,400,150);

        contenedorprincipal.add(PanelSalida);
    }

/*****
Inicializa eventos asociados a algunos componentes de la ventana.
*****/
void inieventos(){

    //Para eventos producidos sobre el boton anterior.
    //Para poder pulsar boton con tecla.
    anterior.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Anterior();
        }
    });

    anterior.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Anterior();
        }
    });

    //Para eventos producidos sobre el boton siguiente.
    //Para poder pulsar boton con tecla.
    siguiente.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiente();
        }
    });

    siguiente.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiente();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });

}

/*****
Metodo que se ejecutara cuando usuario oprima el boton 'Anterior'.
*****/
private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op = new MenuOpciones(agentcth,PanelSalida,valores);

    //Se pasa valores introducidos por usuario al objeto que implementa
    //el menu que se va a abrir.
    op.hospital=pacient.getHospitalIngres();
}

```

```

        op.addWindowListener(l);

        dispose();
    }

/*****
Metodo que se ejecutara cuando usuario oprima el boton 'Siguiete'.
*****/
private void Ejecutar_Siguiete()
{
    fechamuerte=t_fechamuerte.getText();
    horamuerte=t_horamuerte.getText();
    if (ControlErrores()==true)
    {
        //Para guardar valores introducidos por paciente en ontologia.
        //Para pasar fecha de muerte de string a 3 variables Long.
        char f[] = fechamuerte.toCharArray();
        Long dmort=new Long(Character.digit(f[0],10)*10+
        Character.digit(f[1],10));

        Long mmort=new Long(Character.digit(f[3],10)*10+
        Character.digit(f[4],10));

        Long amort=new Long(Character.digit(f[6],10)*1000+
        Character.digit(f[7],10)*100+Character.digit(f[8],10)*10+
        Character.digit(f[9],10));

        //Para pasar hora de muerte de string a 2 variables Long.
        char h[] = horamuerte.toCharArray();

        Long hmort=new Long(Character.digit(h[0],10)*10+
        Character.digit(h[1],10));

        Long mimort=new Long(Character.digit(h[3],10)*10+
        Character.digit(h[4],10));

        pacient.setDiaMort(dmort);
        pacient.setMesMort(mmort);
        pacient.setAnyMort(amort);
        pacient.setHoraMort(hmort);
        pacient.setMinutMort(mimort);

        //Evento para avisar a AgentCTH de que se ha de enviar informacion.
        agentcth.postGuiEvent(new jade.gui.GuiEvent
        (pacient,agentcth.INTRODUCIRPACIENTE));

        //Se pasa Objetos creados en AgentCTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuDonante d=new MenuDonante(agentcth,PanelSalida,valores);

        //Se pasa valores introducidos por usuario en un objeto definido
        //en la ontologia.
        d.pacient=pacient;

        d.addWindowListener(l);

        dispose();
    }
}

/*****
Metodo que valida valores introducidos por usuario.
*****/
boolean ControlErrores()
{
    if (fechamuerte.length()==0)
    {
        valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
}

```

```

    }
    else
    {
        int ok = u.ControlFecha(fechamuerte);
        if (ok==1) //Error en formato de fecha.
        {
            valores.append("\n\nERROR:\nFormato de fecha introducida
incorrecto.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==2) //Error de fecha inexistente.
        {
            valores.append("\n\nERROR:\nFecha introducida inexistente.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==3) //Error de los limites de fecha.
        {
            valores.append("\n\nERROR:\nFecha introducida fuera de limites.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
    }
    if (horamuerte.length()==0)
    {
        valores.append("\n\nERROR:\nSe han de rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    else
    {
        int ok=u.ControlHora(horamuerte);
        if (ok==1) //Formato de hora incorrecto.
        {
            valores.append("\n\nERROR:\nFormato de hora introducida incorrecto.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==2) //Hora introducida inexistente.
        {
            valores.append("\n\nERROR:\nHora introducida inexistente.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
    }
    return(true);
}
}

```



### 8.2.2.8 MenuOpciones.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Menu opciones de la interficie utilizada por el agente AgentCTH.
En este menu, usuario elige la accion que desea realizar.

*****/

package Agent0.agentes.CTH.interficie;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

import Agent0.agentes.CTH.*;
import Agent0.agentes.CTH.Ontology.*;

/*****
MenuOpciones es la clase que corresponde al menu en el cual se permite al
usuario escoger la accion que desea realizar.
*****/
public class MenuOpciones extends JFrame{

    //Objeto de la classe AgentCTH para recibir por parametro desde
    MenuPrincipal.
    //Y poder enviar eventos a AgentCTH desde este menu.
    private AgentCTH agentcth;

    //Datos que determinan el tamaño de la ventana y el titulo.
    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Menú Opciones de Agent0";

    //Objetos de la ontologia que contendran valores introducidos por usuario
    //en la interficie grafica.
    Hospital hospital;

    //acciones contiene las posibles acciones entre las que el usuario debera
    //escoger.
    String acciones[] = {"Introducir Paciente",
        "Anotar Asignación de órgano",
        "Anotar Operación Realizada",
        "Anotar órgano Rechazado",
        "Anotar Muerte Paciente",
        "Encontrar Pacientes Compatibles"
    };

    //Definición del contenedor principal del JFrame de MenuOpciones.
    Container contenedorprincipal;

    //Variables globales
    String s_accion;

    //Creación de un borde con titulo incorporado.
    AbstractBorder borde = new TitledBorder("Escoger opción que desee"+
    "realizar.");

    //Componentes Swing:
    //Creación de panel.
    JPanel panel = new JPanel();

    //Creación de las etiquetas.

```

```

JLabel e_realizar_accion= new JLabel("Realizar acción: ");

//Creacion de ComboBox.
JComboBox realizar_accion = new JComboBox(acciones);

//Creacion de botones.
JButton anterior = new JButton ("Anterior");
JButton siguiente = new JButton ("Siguiente");
JButton salir = new JButton ("Salir");

//Objetos para recibir mensajes del sistema.
private JScrollPane PanelSalida;
private JTextArea valores;

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interficie.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor de la ventana del menu opciones.
*****/
public MenuOpciones(AgentCTH a,JScrollPane panelsalida, JTextArea salida){
    super(TITLE);
    setSize(WIDTH,HEIGHT);
    setResizable(false);

    //Para poder pasar eventos al objeto AgentCTH correspondiente.
    agentcth = a;

    //Para mostrar mensajes de salida del sistema.
    valores = salida;
    valores.setText(""); //Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.

    PanelSalida=panelsalida;

    elementos();
    inieventos();
    show();
}

/*****
Determina tamaño y colocacion de los diferentes componentes
Swing en el menu opciones.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_realizar_accion.setBounds(20,75,115,20);
    realizar_accion.setBounds(125,75,220,20);

    anterior.setBounds(20,230,100,20);
    siguiente.setBounds(150,230,100,20);
    salir.setBounds(310,230,100,20);

    panel.setBorder(borde);

    panel.setBounds(10,10,400,150);

    panel.add(e_realizar_accion);
    panel.add(realizar_accion);

    contenedorprincipal.add(panel);

```

```

        contenedorprincipal.add(anterior);
        contenedorprincipal.add(siguiete);
        contenedorprincipal.add(salir);

        //Para mensajes de salida del sistema.
        valores.setToolTipText ("Mensajes de salida procedentes del sistema");
        valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
        PanelSalida.setViewportView(valores);
        PanelSalida.setBounds(10,260,400,150);
        contenedorprincipal.add(PanelSalida);
    }

/*****
Inicializa los eventos producidos sobre los diferentes componentes Swing.
*****/
void inieventos(){
    realizar_accion.addItemListener ( new ManejadorAccion());

    //Para eventos producidos sobre el boton anterior.
    //Para poder pulsar boton con tecla.
    anterior.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Anterior();
        }
    });

    anterior.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Anterior();
        }
    });

    //Para eventos producidos sobre el boton siguiete.
    //Para poder pulsar boton con tecla.
    siguiete.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiete();
        }
    });

    siguiete.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiete();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });
}

/*****
Se actualiza el string s_accion segun la eleccion del usuario.
*****/
class ManejadorAccion implements ItemListener {
    public void itemStateChanged(ItemEvent e){

```

```

        s_accion=realizar_accion.getSelectedItem().toString();
    }
}

/*****
Metodo que se ejecuta si se ha pulsado boton anterior de interficie
grafica, con la tecla intro o con el raton.
*****/
private void Ejecutar_Anterior()
{
    //Se pasa Objetos creados en agente CTH para comunicacion entre AgentCTH
    //e interficie grafica.
    MenuPrincipal mp = new MenuPrincipal(agentctch,PanelSalida,valores);

    //Se asocia adaptador de evento a la ventana que se va abrir.
    mp.addWindowListener(l);

    //Se cierra ventana actualmente activa de la interficie grafica.
    dispose();
}

/*****
Metodo que se ejecuta cuando se ha oprimido la tecla Siguiete, con el
raton o pulsando la tecla intro.
*****/
public void Ejecutar_Siguiete()
{
    //Se abre la ventana correspondiente segun la accion elegida por el
    //usuario.
    if (realizar_accion.getSelectedItem().equals("Anotar Muerte Paciente"))
    {
        //Se pasa Objetos creado desde agente CTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuAnotarMuertePaciente amp = new
        MenuAnotarMuertePaciente(agentctch,PanelSalida,valores);

        //Se pasa valores actuales del objeto hospital a ventana que se va a
        //abrir.
        amp.hospital=hospital;

        //Se asocia adaptador de evento a la nueva ventana que se va a abrir.
        amp.addWindowListener(l);

        //Se cierra ventana actualmente activa, sin generar evento.
        dispose();
    }

    if (realizar_accion.getSelectedItem().equals
    ("Encontrar Pacientes Compatibles"))
    {
        //Se pasa Objetos creado desde agente CTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuEncontrarPacientesCompatibles epc = new
        MenuEncontrarPacientesCompatibles(agentctch,PanelSalida,valores);

        //Se pasa a la nueva ventana los valores actuales de hospital.
        epc.hospital=hospital;

        //Se asocia adaptador a la nueva ventana que se va a abrir.
        epc.addWindowListener(l);

        //Se cierra ventana actual sin generar ningun evento.
        dispose();
    }
    if (realizar_accion.getSelectedItem().equals
    ("Anotar Asignaci n de  rgano"))
    {
        //Se pasa Objetos creado en AgentCTH para comunicacion
        //entre AgentCTH e interficie grafica.

```

```

        MenuAnotarAsignacionOrgano aao = new
MenuAnotarAsignacionOrgano(agentcth, PanelSalida, valores);
        //Se pasa valores actuales de hospital a la nueva "ventana".
        aao.hospital=hospital;

        //Se asocia adaptador a la nueva ventana que se va a abrir.
        aao.addWindowListener(l);

        //Se cierra ventana actualmente activa sin generar evento.
        dispose();
    }

    if (realizar_accion.getSelectedItem().equals
("Anotar ãrgano Rechazado"))
    {
        //Se pasa Objetos creado desde AgentCTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuAnotarOrganoRechazado aor = new
MenuAnotarOrganoRechazado(agentcth, PanelSalida, valores);

        //Se pasa a clase, que implementa ventana que se va a abrir,
        //los valores actuales de hospital.
        aor.hospital=hospital;

        //Se asocia adaptador de evento a la nueva ventana.
        aor.addWindowListener(l);

        //Se cierra ventana actualmente activa, sin generar eventos.
        dispose();
    }

    if (realizar_accion.getSelectedItem().equals
("Anotar Operaciøn Realizada"))
    {
        //Se pasa Objetos creado desde AgentCTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuOperacionRealizada or = new
MenuOperacionRealizada(agentcth, PanelSalida, valores);

        //Se pasan valores recogidos en interficie a la clase que implementa
        //la ventana que se va a abrir.
        or.hospital=hospital;

        or.addWindowListener(l);

        dispose();
    }
    if (realizar_accion.getSelectedItem().equals("Introducir Paciente"))
    {
        //Se pasa Objetos creados en AgentCTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuIntroducirPaciente ip = new
MenuIntroducirPaciente(agentcth, PanelSalida, valores);

        //Para ir guardando valores introducidos por usuario mientras
        //navega por la interficie grafica.
        ip.hospital=hospital;

        ip.addWindowListener(l);

        dispose();
    }
}
}
}

```

### 8.2.2.9 MenuOperacionRealizada.java

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Implementacion de menu para introducir datos relativos a la operacion
realizada a un paciente receptor, desde interficie grafica asociada a
AgentCTH.

*****/
package Agent0.agentes.CTH.interficie;

//Importamos clase que implementa a AgentCTH.
import Agent0.agentes.CTH.*;

//Importamos ontologia para comunicar AgentCTH y Agent0.
import Agent0.agentes.CTH.Ontology.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

/*****
Clase que implementa a este menu de la interficie grafica.
*****/
public class MenuOperacionRealizada extends JFrame{

    //Objeto que recibira por parametro una instancia de AgentCTH, en el
    //constructor de la clase. Sirve para poder enviar eventos desde
    //este menu.
    private AgentCTH agentcth;

    //Clase que contiene metodos utiles en diferentes archivos de la
    //interficie grafica.
    private Utilidades u = new Utilidades();

    //Objeto definido en la ontologia para guardar valores introducidos
    //por usuario mientras navega por la interficie grafica.
    Hospital hospital;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Anotar Operación Realizada.";

    Container contenedorprincipal;

    //Variables globales
    String idorgano;
    String fechaoperacion;
    String horaoperacion;

    //Componentes Swing
    AbstractBorder borde = new TitledBorder("Datos relativos a una
operación.");

    JPanel panel = new JPanel();

    JLabel e_idorgano = new JLabel ("Identificador òrgano: ");
    JLabel e_fechaoperacion = new JLabel ("Fecha de Operación (dd/mm/aaaa):
");
    JLabel e_horaoperacion = new JLabel ("Hora de la Operación (HH:MM): ");

    JTextField t_idorgano = new JTextField();
    JTextField t_fechaoperacion = new JTextField();

```

```

    JTextField t_horaoperacion = new JTextField();

    JButton anterior = new JButton ("Anterior");
    JButton siguiente = new JButton ("Siguiente");
    JButton salir = new JButton ("Salir");

    //Para mostrar mensajes de salida del sistema.
    private JScrollPane PanelSalida;
    private JTextArea valores;

    //Para salir de la aplicacion, cuando se cierra la ventana activa de la
    //interficie.
    WindowListener l= new WindowAdapter(){
        public void windowClosing (WindowEvent e){
            System.exit(0);
        }
    };

/*****
Constructor de la clase que implementa a este menu.
*****/
public MenuOperacionRealizada(AgentCTH a, JScrollPane panelsalida,
    JTextArea salida){
    super(TITLE);
    setSize(WIDTH,HEIGHT);
    setResizable(false);

    agentcth = a;
    valores = salida;
    valores.setText(""); //Poniendo esto (salida) es inutil,
                        //pero lo dejo por si se cambia de criterio.

    PanelSalida=panelsalida;

    elementos();
    inieventos();
    show();
}

/*****
Metodo que describe elementos que apareceran en este menu y su colocacion.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_idorgano.setBounds(10,30,140,20);
    e_fechaoperacion.setBounds(10,60,200,20);
    e_horaoperacion.setBounds(10,90,200,20);

    t_idorgano.setBounds(150,30,100,20);
    t_fechaoperacion.setBounds(215,60,75,20);
    t_horaoperacion.setBounds(200,90,50,20);

    anterior.setBounds(20,230,100,20);
    siguiente.setBounds(150,230,100,20);
    salir.setBounds(310,230,100,20);

    panel.add(e_idorgano);
    panel.add(t_idorgano);

    panel.add(e_fechaoperacion);
    panel.add(t_fechaoperacion);

    panel.add(e_horaoperacion);
    panel.add(t_horaoperacion);

    panel.setBorder(borde);

```

```

panel.setBounds(10,10,400,150);

contenedorprincipal.add(panel);

contenedorprincipal.add(anterior);
contenedorprincipal.add(siguiente);
contenedorprincipal.add(salir);

//Para mostrar mensajes de salida del sistema.
valores.setToolTipText ("Mensajes de salida procedentes del sistema");
valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
PanelSalida.setViewPortView(valores);
PanelSalida.setBounds(10,260,400,150);
contenedorprincipal.add(PanelSalida);
}

/*****
Metodo que inicializa eventos asociados a algunos componentes del menu.
*****/
void inieventos(){
    //Para eventos producidos sobre el boton anterior.
    //Para poder pulsar boton con tecla.
    anterior.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Anterior();
        }
    });

    anterior.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Anterior();
        }
    });

    //Para eventos producidos sobre el boton siguiente.
    //Para poder pulsar boton con tecla.
    siguiente.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiente();
        }
    });

    siguiente.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiente();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });
}

/*****
Metodo que se ejecuta cuando usuario oprima el boton 'Anterior' del menu.
*****/
private void Ejecutar_Anterior()

```



```

{
    //Se pasa Objetos creados en AgentCTH para comunicacion
    //entre AgentCTH e interficie grafica.
    MenuOpciones op = new MenuOpciones(agentcth, PanelSalida, valores);

    op.hospital=hospital;

    op.addWindowListener(l);

    dispose();
}

/*****
Metodo que se ejecuta cuando usuario oprima el boton 'Siguiete' del menu.
*****/
private void Ejecutar_Siguiete()
{
    idorgano=" "+t_idorgano.getText();

    fechaoperacion=t_fechaoperacion.getText();
    horaoperacion=t_horaoperacion.getText();

    if (ControlErrores()==true)
    { //Para eliminar espacios en blanco.
        idorgano=u.StringSinEspaciosEnBlanco(idorgano);

        //Envia evento a la clase agentCTH.

        //Se supone que se introduzcan los datos de una operacion,
        //cuando el paciente ya tenga organo asignado previamente.

        //Para guardar datos en objeto definido en ontologia.
        //Para pasar fecha de operacion de string a 3 variables Long.
        char f[] = fechaoperacion.toCharArray();
        Long dop = new
Long(Character.digit(f[0],10)*10+Character.digit(f[1],10));
        Long mo = new
Long(Character.digit(f[3],10)*10+Character.digit(f[4],10));
        Long ao = new
Long(Character.digit(f[6],10)*1000+Character.digit(f[7],10)*100+Character.digit(f[8],10)*10+Character.digit(f[9],10));
        //Para pasar hora de operacion de string a 2 variables Long.
        char h[] = horaoperacion.toCharArray();
        Long ho = new
Long(Character.digit(h[0],10)*10+Character.digit(h[1],10));
        Long mop = new
Long(Character.digit(h[3],10)*10+Character.digit(h[4],10));

        Operacio O = new Operacio(dop,mo,ao,ho,mop,idorgano,hospital);

        //Envia evento a la instancia de la clase AgentCTH.
        agentcth.postGuiEvent(new
jade.gui.GuiEvent(0,agentcth.OPERACIONREALIZADA));
    }
}

/*****
Metodo que valida datos introducidos por usuario.
*****/
boolean ControlErrores()
{
    if (t_idorgano.getText().equals(""))
    {
        valores.append("\n\nERROR:\nSe tienen que rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }

    if (t_idorgano.getText().length(>10)
    {

```

```

        valores.append("\n\nERROR:\n"+
        "El identificador de organo no puede tener mas de 10 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    if (fechaoperacion.length()==0)
    {
        valores.append("\n\nERROR:\nSe tienen que rellenar todos los campos.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    else
    {
        int ok=u.ControlFecha(fechaoperacion);
        if (ok==1) //Error de formato de fecha.
        {
            valores.append("\n\nERROR:\nFormato de fecha incorrecto.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==2) //Error de fecha inexistente.
        {
            valores.append("\n\nERROR:\nFecha introducida inexistente.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
        if (ok==3) //Error de los limites de fecha.
        {
            valores.append("\n\nERROR:\n Fecha introducida fuera de limites.");
            u.PosicionarFinalScrollBar (PanelSalida);
            return(false);
        }
    }
}
if (horaoperacion.length()==0)
{
    valores.append("\n\nERROR:\n Se tienen que rellenar todos los campos.");
    u.PosicionarFinalScrollBar (PanelSalida);
    return(false);
}
else
{
    int ok = u.ControlHora(horaoperacion);
    if (ok==1) //Error en formato de hora.
    {
        valores.append("\n\nERROR:\nFormato de hora introducida incorrecta.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
    if (ok==2) //Error de hora inexistente.
    {
        valores.append("\n\nERROR:\nHora introducida inexistente.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }
}
return(true);
}
}

```

### 8.2.2.10 *MenuPrincipal.java*

```

/*****

Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Menu principal de la interficie utilizada por el agente AgentCTH.

*****/

package Agent0.agentes.CTH.interficie;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

import jade.gui.*;

import Agent0.agentes.CTH.*;
import Agent0.agentes.CTH.Ontology.*;

/*****
    MenuPrincipal es la clase que corresponde al menu
    principal de la interficie lanzada por AgentCTH,
    y se piden los datos del hospital donde se encuentra el usuario.
*****/

public class MenuPrincipal extends JFrame{

    //Objeto de tipo AgentCTH que servira para comunicarse con este agente
    //desde la interficie. Para enviar eventos de la interficie a la
    //instanciacion de AgentCTH correspondiente.
    private AgentCTH agentcth;

    //Para tratar datos introducidos por usuario. En esta clase se validan
    //datos introducidos por usuario,...
    Utilidades u=new Utilidades();

    //Datos que determinan el tamaño de la ventana y el titulo.*/
    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Menú Principal de Agent0";

    //Definicion del contenedor principal del JFrame de
    //Menu principal.
    Container contenedorprincipal;

    //Variables globales
    String nombre;
    String calle;
    String numero;
    String ciudad;
    String codigopostal;
    String provincia;
    String pais;

    //Inicializamos el objeto Hospital y Adreza de la ontologia para ir
    //guardando datos, mientras el usuario navega por la interficie grafica.
    Hospital hospital;
    Adreza adreza;

    //Creacion de un borde con titulo incorporado.
    AbstractBorder borde = new
    TitledBorder("Datos del hospital en donde est  ejecutando este programa.");

```

```

//Componentes Swing:

//Creacion de panel.
JPanel panel = new JPanel();

//Creacion de etiquetas.
JLabel e_nombre = new JLabel ("Nombre: ");
JLabel e_calle = new JLabel ("Calle: ");
JLabel e_numero = new JLabel ("Número: ");
JLabel e_ciudad = new JLabel ("Ciudad: ");
JLabel e_codigopostal= new JLabel ("Código Postal: ");
JLabel e_provincia = new JLabel ("Provincia: ");
JLabel e_pais = new JLabel ("País: ");

//Creacion de campos de texto.
JTextField t_nombre = new JTextField();
JTextField t_calle = new JTextField();
JTextField t_numero = new JTextField();
JTextField t_ciudad = new JTextField();
JTextField t_codigopostal = new JTextField();
JTextField t_provincia = new JTextField();
JTextField t_pais = new JTextField();

//Estos dos objetos serviran para recibir mensajes de salida desde
//AgentCTH a interfaz grafica.
private JScrollPane PanelSalida;
private JTextArea valores;

//Creacion de botones.
JButton siguiente = new JButton ("Siguiente");
JButton salir = new JButton ("Salir");

//Para salir de la aplicacion, cuando se cierra la ventana activa de la
//interfaz.
WindowListener l= new WindowAdapter(){
    public void windowClosing (WindowEvent e){
        System.exit(0);
    }
};

/*****
Constructor de la ventana del menu principal.
*****/

public MenuPrincipal(AgentCTH a,JScrollPane panelsalida, JTextArea salida){
    super(TITLE);
    try{

        //Guardamos la instancia de AgentCTH que se nos pasa por parametro, para
        //poder enviar eventos a este agente.
        this.agentctth = a;

        //Guardamos instancia de estos dos objetos, para incorporarlos al
        //MenuPrincipal y poder mostrar mensajes de salida de AgentCTH.
        this.valores=salida;
        valores.setText(""); //Poniendo esto (salida) es inutil,
                             //pero lo dejo por si se cambia de criterio.

        PanelSalida=panelsalida;

        this.setSize(WIDTH,HEIGHT);
        this.setResizable(false);

        //Se especifican elementos y colocacion de estos en MenuPrincipal.
        this.elementos();

        //Se programan eventos producidos al interactuar con elementos de
        //la interfaz. (Ejemplo: Pulsacion tecla intro cuando se esta sobre

```

```

        //uno de los botones de la interficie).
        this.inieventos();
        this.show();
    }
    catch (Exception e){
        e.printStackTrace();
    }
}

/*****
Se especifican elementos y colocacion de estos en MenuPrincipal.
*****/
void elementos(){
    contenedorprincipal = getContentPane();
    contenedorprincipal.setLayout(null);

    panel.setLayout(null);

    e_nombre.setBounds(10,20,70,20);
    e_calle.setBounds(10,50,60,20);
    e_numero.setBounds(290,50,70,20);
    e_ciudad.setBounds(10,80,65,20);
    e_codigopostal.setBounds(240,80,90,20);
    e_provincia.setBounds(10,105,70,20);
    e_pais.setBounds(235,105,50,20);

    t_nombre.setBounds(75,20,200,20);
    t_calle.setBounds(65,50,200,20);
    t_numero.setBounds(355,50,25,20);
    t_ciudad.setBounds(70,80,150,20);
    t_codigopostal.setBounds(330,80,50,20);
    t_provincia.setBounds(105,105,100,20);
    t_pais.setBounds(285,105,100,20);

    siguiente.setBounds(20,230,100,20);
    salir.setBounds(150,230,100,20);

    panel.add(e_nombre);
    panel.add(e_calle);
    panel.add(e_numero);
    panel.add(e_ciudad);
    panel.add(e_codigopostal);
    panel.add(e_provincia);
    panel.add(e_pais);

    panel.add(t_nombre);
    panel.add(t_calle);
    panel.add(t_numero);
    panel.add(t_ciudad);
    panel.add(t_codigopostal);
    panel.add(t_provincia);
    panel.add(t_pais);

    panel.setBorder(borde);

    panel.setBounds(10,10,400,150);

    contenedorprincipal.add(panel);

    contenedorprincipal.add(siguiente);
    contenedorprincipal.add(salir);

    //Para mostrar mensajes de salida del sistema.
    valores.setToolTipText ("Mensajes de salida procedentes del sistema");
    valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
    PanelSalida.setViewPortView(valores);
    PanelSalida.setBounds(10,260,400,150);

```

```

        contenedorprincipal.add(PanelSalida);
    }

/*****
Inicializa los eventos producidos sobre los diferentes
componentes Swing.
*****/
void inieventos(){

    //Para eventos producidos sobre el boton siguiente.
    //Para poder pulsar boton con tecla.
    siguiente.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiente();
        }
    });

    //Pulsacion de boton con raton.
    siguiente.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiente();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });

}

/*****
Metodo que se ejecuta cuando usuario ha seleccionado el
boton Siguiente del menu de la interficie.
*****/
private void Ejecutar_Siguiente()
{
    //Se recogen datos introducidos por el usuario.
    nombre=" "+t_nombre.getText();
    calle=" "+t_calle.getText();
    numero=" "+t_numero.getText();
    ciudad=" "+t_ciudad.getText();
    codigopostal=" "+t_codigopostal.getText();
    provincia=" "+t_provincia.getText();
    pais=" "+t_pais.getText();

    //Se validan valores introducidos por usuario.
    if (ControlErrores())
    {
        //Se abre la ventana opciones para que el usuario elija la accion
        //a realizar.
        //Se pasa Objetos creado desde agente CTH para comunicacion
        //entre AgentCTH e interficie grafica.
        MenuOpciones mo = new MenuOpciones(agentcth,PanelSalida,valores);

        //Para controlar que no se introducen espacios en blanco,
        //y si se hace sustituirlos por la cadena '/' para no generar
        //errores con la ontologia.

```

```

nombre=u.StringSinEspaciosEnBlanco(nombre);
calle=u.StringSinEspaciosEnBlanco(calle);
numero=u.StringSinEspaciosEnBlanco(numero);
ciudad=u.StringSinEspaciosEnBlanco(ciudad);
codigopostal=u.StringSinEspaciosEnBlanco(codigopostal);
provincia=u.StringSinEspaciosEnBlanco(provincia);
pais=u.StringSinEspaciosEnBlanco(pais);

//Para ir almacenando valores en los objetos de la ontologia, que se
//han recogido desde interficie grafica.
adreza=new Adreza(calle,numero,ciudad,codigopostal,provincia,pais);
hospital=new Hospital(nombre,adreza);

//Para ir pasando valores de un menu a otro de la interficie grafica.
mo.hospital=hospital;

//Se assigna el adaptador anterior a la ventana que se va abrir
//de la interficie grafica.
mo.addWindowListener(l);

//Se cierra la ventana del MenuPrincipal, pero sin generar ningun evento.
dispose();
}
}

/*****
Metodo que sirve para controlar errores a la hora de introducir datos.
*****/
boolean ControlErrores()
{
//Entero que hace de booleano que indicara si se ha dejado algun campo
vacio.
int i=0;

if ((t_nombre.getText()).length(>55)
{
valores.append("\n\n ERROR:\nEl nombre del hospital no puede"+
"tener mas de 55 caracteres.");
u.PosicionarFinalScrollBar (PanelSalida);
return false;
}

if ((t_calle.getText()).length(>30)
{
valores.append("\n\n ERROR:\nLa calle del hospital no puede"+
"tener mas de 55 caracteres.");
u.PosicionarFinalScrollBar (PanelSalida);
return false;
}

if ((t_numero.getText()).length(>10)
{
valores.append("\n\nERROR:\nEl numero del hospital no puede"+
"tener mas de 10 caracteres.");
u.PosicionarFinalScrollBar (PanelSalida);
return false;
}

if ((t_ciudad.getText()).length(>55)
{
valores.append("\n\nERROR:\nLa ciudad del hospital no puede"+
"tener mas de 55 caracteres.");
u.PosicionarFinalScrollBar (PanelSalida);
return false;
}

if ((t_codigopostal.getText()).length(>15)
{
valores.append("\n\nERROR:\nEl codigo postal del hospital no puede"+
"tener mas de 15 caracteres.");

```

```

        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    if ((t_provincia.getText()).length()>55)
    {
        valores.append("\n\nERROR:\nLa provincia del hospital no puede"+
            "tener mas de 55 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    if ((t_pais.getText()).length()>55)
    {
        valores.append("\n\nERROR:\nEl pais del hospital no puede"+
            "tener mas de 55 caracteres.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    if (t_nombre.getText().equals("")) i = 1;
    if (t_calle.getText().equals("")) i=1;
    if (t_numero.getText().equals("")) i=1;
    if (t_ciudad.getText().equals("")) i=1;
    if (t_codigopostal.getText().equals("")) i=1;
    if (t_provincia.getText().equals("")) i=1;
    if (t_pais.getText().equals("")) i=1;

    if (i==1)
    {
        valores.append("\n\nERROR:\nNo puede haber ningun campo vacio.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }else return(true);
}
}

```



### 8.2.2.11 *MenuReceptor.java*

```

/*****
Agent0.agentes.CTH.interficie.

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

MenuReceptor para introducir datos relativos a un paciente receptor desde
interficie grafica asociada a AgentCTH.
*****/

package Agent0.agentes.CTH.interficie;

//Importamos la clase que implementa el agentCTH.
import Agent0.agentes.CTH.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

import java.util.*;

import Agent0.agentes.CTH.Ontology.*;
import jade.gui.*;

public class MenuReceptor extends JFrame{

    private AgentCTH agentcth;

    private Utilidades u = new Utilidades();

    Pacient pacient;
    Organ organ;
    Hospital hospital;

    public static int WIDTH = 450;
    public static int HEIGHT = 450;
    public static String TITLE = "Datos relativos al organo a recibir.";

    Container contenedorprincipal;

    //Variables globales
    String idorgano;
    String tipoorgano;
    String tamañoorgano;

    String tipos_organos []={"Corazon","Higado","Rinyon","Pulmon"};

    //Componentes Swing

    AbstractBorder borde = new TitledBorder("Datos relativos al organo a
recibir.");

    JPanel panel = new JPanel();

    JLabel e_tipoorgano = new JLabel ("Tipo Organ: ");
    JLabel e_tamañoorgano = new JLabel ("Tamanyo Organ (cms.): ");

    JComboBox t_tipoorgano = new JComboBox(tipos_organos);

    JTextField t_idorgano = new JTextField();

```

```

    JTextField t_tamapoorgano = new JTextField();

    JButton anterior = new JButton ("Anterior");
    JButton siguiente = new JButton ("Siguiente");
    JButton salir = new JButton ("Salir");

    /** Para recibir mensajes del sistema.*/

    private JScrollPane PanelSalida;
    private JTextArea valores;

    //Para salir de la aplicacion, cuando se cierra la ventana activa de la
    //interficie.
    WindowListener l= new WindowAdapter(){
        public void windowClosing (WindowEvent e){
            System.exit(0);
        }
    };

    /*****
    Constructor de MenuReceptor.
    *****/
    public MenuReceptor(AgentCTH a, JScrollPane panelsalida, JTextArea salida){

        super(TITLE);
        setSize(WIDTH,HEIGHT);
        setResizable(false);

        agentcth = a;
        valores = salida;
        valores.setText(""); //Poniendo esto (salida) es inutil,
                            //pero lo dejo por si se cambia de criterio.

        PanelSalida=panelsalida;

        elementos();
        inieventos();
        show();
    }

    /*****
    Descripcion de elementos que aparecen en este menu.
    *****/
    void elementos(){
        contenedorprincipal = getContentPane();
        contenedorprincipal.setLayout(null);

        panel.setLayout(null);

        e_tipoorgano.setBounds(10,30,85,20);
        e_tamapoorgano.setBounds(10,60,150,20);

        t_tipoorgano.setBounds(105,30,75,20);
        t_tamapoorgano.setBounds(165,60,50,20);

        anterior.setBounds(20,230,100,20);
        siguiente.setBounds(150,230,100,20);
        salir.setBounds(310,230,100,20);

        panel.add(e_tipoorgano);
        panel.add(e_tamapoorgano);

        panel.add(t_tipoorgano);
        panel.add(t_tamapoorgano);

        panel.setBorder(borde);

        panel.setBounds(10,10,400,100);

        contenedorprincipal.add(panel);
    }

```

```

        contenedorprincipal.add(anterior);
        contenedorprincipal.add(siguiete);
        contenedorprincipal.add(salir);

        //Para mostrar mensajes de salida del sistema.
        valores.setToolTipText ("Mensajes de salida procedentes del sistema");
        valores.setBorder( new EmptyBorder (new Insets(1, 1, 1, 1)));
        PanelSalida.setViewportView(valores);
        PanelSalida.setBounds(10,260,400,150);
        contenedorprincipal.add(PanelSalida);

    }

/*****
Inicializacion de eventos asociados a componentes de la ventana.
*****/
void inieventos(){
    //Para eventos producidos sobre el boton anterior.
    //Para poder pulsar boton con tecla.
    anterior.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Anterior();
        }
    });

    anterior.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Anterior();
        }
    });

    //Para eventos producidos sobre el boton siguiete.
    //Para poder pulsar boton con tecla.
    siguiete.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) Ejecutar_Siguiete();
        }
    });

    siguiete.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            Ejecutar_Siguiete();
        }
    });

    //Para eventos producidos sobre el boton salir.
    salir.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            //Si se ha pulsado Return.
            if (e.getKeyCode()==10) System.exit(0);
        }
    });

    salir.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e){
            System.exit(0);
        }
    });
}

/*****
Metodo que se ejecutara cuando usuario oprima boton 'Anterior' del menu.
*****/
private void Ejecutar_Anterior()
{

```

```

//Se pasa Objetos creados en AgentCTH para comunicacion
//entre AgentCTH e interficie grafica.
MenuOpciones op = new MenuOpciones(agentctth, PanelSalida, valores);

op.hospital=pacient.getHospitalIngres();

op.addWindowListener(l);

dispose();
}

/*****
Metodo que se ejecutara cuando usuario oprima boton 'Siguiete' del menu.
*****/

private void Ejecutar_Siguiete()
{
    tipoorgano=t_tipoorgano.getSelectedItem().toString();
    tamaoorgano=t_tamaoorgano.getText();

    if (ControlErrores()==true)
    {
        Double tamany=new Double(tamaoorgano);

        //No hace falta quitar espacios en blanco pero si codificar
        //los caracteres especiales 'a' y '+'.
        tipoorgano=u.StringSinEspaciosEnBlanco(tipoorgano);

        //Actualizamos datos de entrada al sistema.
        GregorianCalendar c = new GregorianCalendar();

        Long diaentradasistema=new Long(c.get(Calendar.DATE));
        //Hay que aadir +1, porque los meses se cuentan desde 0.
        Long mesentradasistema=new Long(c.get(Calendar.MONTH)+1);
        Long anyentradasistema=new Long(c.get(Calendar.YEAR));
        int hesistema=c.get(Calendar.HOUR);
        if (c.get(Calendar.AM_PM)==1) hesistema+=12;
        Long horaentradasistema=new Long(hesistema);
        Long minutentradasistema=new Long(c.get(Calendar.MINUTE));

        pacient.setOrgano(organo);
        pacient.setEstat("En/_Espera/_De/_Organo");
        pacient.setDiaEntradaSistema(diaentradasistema);
        pacient.setMesEntradaSistema(mesentradasistema);
        pacient.setAnyEntradaSistema(anyentradasistema);
        pacient.setHoraEntradaSistema(horaentradasistema);
        pacient.setMinutEntradaSistema(minutentradasistema);

        //El ultimo parametro de Organ indica si el organo es
        //donado o es una descripcion de lo que se espera
        //recibir. NO CONFUNDIR QUE ESTAT en PACIENT.
        organ=new Organ(null,tipoorgano,tamany,"Esperat");

        pacient.setOrgano(organ);

        //Envia evento a la clase agentCTH.
        agentctth.postGuiEvent(new
jade.gui.GuiEvent(pacient,agentctth.INTRODUCIRPACIENTE));
    }
}

/*****
Metodo servira para validar datos introducidos por usuario.
*****/

boolean ControlErrores()
{
    double t;

    if (tamaoorgano.length()==0)

```

```

    {
        valores.append("\n\nERROR:\nNo puede haber ningun campo vacio.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }

    if (u.ControlDouble(tamañoorgano)==false)
    {
        valores.append("\n\nERROR:\nEl tamaño del organo tiene que ser un
numero.");
        u.PosicionarFinalScrollBar (PanelSalida);
        return(false);
    }else t = new Double(tamañoorgano).doubleValue();

    if ((t >100.0) || (t<0.0))
    {
        valores.append("\n\nERROR:\nEl tamaño de organo ha de estar entre 0 y
100");
        u.PosicionarFinalScrollBar (PanelSalida);
        return false;
    }

    return(true);
}
}

```

### 8.2.2.12 Utilidades.java

```

/*****

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Clase que implementa metodos utiles en algunos archivos del AgentCTH.

*****/

package Agent0.agentes.CTH.interficie;

import java.util.*;

import java.awt.*;
import javax.swing.*;

public class Utilidades
{
/*****
Constructor de la clase.
*****/
public Utilidades(){}

/*****
Método que realiza el control de una fecha que se pasa por parámetro,
y devuelve un entero indicando un error concreto.
0 -> Todo correcto.
1 -> Error en formato de fecha.
2 -> Error por fecha inexistente.
3 -> Error por fecha fuera de límites.
*****/
public int ControlFecha(String f)
{
    int dia; /*Para guardar el día introducido por usuario.*/
    int mes; /*Para guardar el mes introducido por usuario.*/
    int año;
    int nc; /*Para indicar número de carácter no válido.*/

    char fecha[]=f.toCharArray();

    /*Tabla de enteros que guardan número de días de los meses de un año, no
    bisiesto.*/
    int[] numDias = {31,28,31,30,31,30,31,31,30,31,30,31};

    if (f.length()==10)
    {
        for (int i=0;i<10;i++)
        {
            nc=i+1;
            if ((i==2) || (i==5))
            {
                if (fecha[i]!='/')
                {
                    return(1);
                }
            }
            else
            {
                if (!Character.isDigit(fecha[i]))
                {
                    return(1);
                }
            }
        }
    }

    /*Control de valores introducidos para la fecha.*/

```

```

dia=Character.digit(fecha[0],10)*10+Character.digit(fecha[1],10);
mes=Character.digit(fecha[3],10)*10+Character.digit(fecha[4],10);
año=Character.digit(fecha[6],10)*1000+Character.digit(fecha[7],10)*100;
año=año+Character.digit(fecha[8],10)*10+Character.digit(fecha[9],10);

//Tratamos caso de que se introduzca fecha fuera de los límites.
//Para crear una fecha actual.
GregorianCalendar c = new GregorianCalendar();
Date avui = new Date();

int diasistema=c.get(Calendar.DATE);
//Hay que añadir +1, porque los meses se cuentan desde 0.
int messistema=c.get(Calendar.MONTH)+1;
int anysistema=c.get(Calendar.YEAR);

//Comprobamos que fecha introducida no sea posterior a la actual.
if (año>anysistema) return 3;
if ((año==anysistema) && (mes>messistema)) return 3;
if ((año==anysistema) && (mes==messistema) && (dia>diasistema)) return 3;

/*Tratamos caso particular de mes de febrero en año bisiesto.*/
if ((mes==2) && (año%4==0) && (dia>29))
{
    return(2);
}
else /*Tratamos los otros casos.*/
{
    if (((mes>12) || (dia>numDias[mes-1])) && !((mes==2) && (dia==29) &&
(año%4==0)))
    {
        return(2);
    }
    return(0);
}
else
{
    return(1);
}
}

/*****
Metodo que convierte un String en un Long. Previamente hace falta
comprobar de que se trata de un numero el String.
*****/
public Long DeStringALong(String s)
{
    Long resultat;
    int intermedio=0;
    char cadena[]=s.toCharArray();

    for (int i=0;i<s.length();i++)
    {
        intermedio+=Character.digit(cadena[i],10);
        if (i<(s.length()-1) intermedio=intermedio*10;
    }
    resultat=new Long(intermedio);
    return resultat;
}

/*****
Metodo que comprueba si el String es un entero.
*****/
public boolean ControlEntero (String e)
{
    char entero[]=e.toCharArray();
    int l=e.length();

```

```

    for (int i=0;i<l;i++)
    {
        if (!Character.isDigit(entero[i]))
        {
            return(false);
        }
    }
    return(true);
}

/*****
Metodo que comprueba si el String es un Double.
*****/

public boolean ControlDouble (String e)
{
    char d[]=e.toCharArray();
    int l=e.length();
    for (int i=0;i<l;i++)
    {
        if ((!Character.isDigit(d[i])) && (d[i]!='.') // && (d[i]!=','))
        {
            return(false);
        }
    }
    return(true);
}

/*****
Metodo que comprueba si el String son 4 booleanos relativos a los
antigenos de una persona.
*****/

public boolean ControlAntigenos (String a)
{ int nc; /*Para indicar numero de caracter no valido.*/
  char antig[]=a.toCharArray();

  if (a.length()==4)
  {
      for (int i=0;i<4;i++)
      {
          if ((antig[i]!='0') && (antig[i]!='1'))
          {
              nc=i+1;
              return (false);
          }
      }
      return(true);
  }
  else
  {
      return(false);
  }
}

/*****
M,todo que mira si la hora introducida por par metro es correcta.
Devuelve lo siguiente:
0 -> Todo es correcto.
1 -> Formato incorrecto.
2 -> Hora inexistente.
*****/

public int ControlHora(String h)
{
    int hora; /*Para guardar la hora introducido por usuario.*/
    int minuto; /*Para guardar el minuto introducido por usuario.*/
    int nc;
    char hora_int[]=h.toCharArray();

    if (h.length()==5)

```



```

{
    for (int i=0;i<5;i++)
    {
        nc=i+1;
        if ((i==2) && (hora_int[2]!=':'))
        {
            return(1);
        }
        else
        {
            if ((i!=2) && (!Character.isDigit(hora_int[i])))
            {
                return(1);
            }
        }
    }
}

/*Control de valores introducidos para la hora.*/

hora=Character.digit(hora_int[0],10)*10+Character.digit(hora_int[1],10);
minuto=Character.digit(hora_int[3],10)*10+Character.digit(hora_int[4],10);

if ((hora>23) || (minuto>59))
{
    return(2);
}
return(0);
}
else
{
    return(1);
}
}

/*****
Metodo que pasa un String a entero. Antes se ha de comprobar si el
String es un numero.
*****/
public int Enter (String t)
{ int resultat=0;
  int n=0;
  char entero[]=t.toCharArray();
  for (int i=0;i<t.length();i++)
  {
      n+=Character.digit(entero[i],10);
      if (i<(t.length()-1) n=n*10;
  }
  resultat+=n;
  return(resultat);
}

/*****
Metodo que codifica los caracteres en blanco recogidos en la interficie
grafica, para poder almacenarlos en la ontologia.
*****/
public String StringSinEspaciosEnBlanco (String s)
{
    String prueba ="
"+
        "
";
    char c[] = s.toCharArray();
    char t[] = prueba.toCharArray();
    String resultado="";
    int j=0;
    int AlgunCasAnterior = 0;
    for (int i=0;i<s.length();i++)
    {
        AlgunCasAnterior = 0;
        //Para sustituir espacios en blanco.

```

```

if (c[i]==' ')
{ //Substituim caracters en blanc per la cadena '/_'.
  t[j]='/';
  t[j+1]='_';
  j+=2;
  AlgunCasAnterior=1;
}

//Para substituir caracters ¢, ¥, ¤, y € y no tener
//problema con Sniffer.
if (c[i]=='¢')
{
  t[j]='/';
  t[j+1]='ú';
  j+=2;
  AlgunCasAnterior=1;
}

if (c[i]=='¥')
{
  t[j]='/';
  t[j+1]='$';
  j+=2;
  AlgunCasAnterior=1;
}

if (c[i]=='¤')
{
  t[j]='%';
  t[j+1]='/';
  j+=2;
  AlgunCasAnterior=1;
}

if (c[i]=='€')
{
  t[j]='&';
  t[j+1]='/';
  j+=2;
  AlgunCasAnterior=1;
}

if (AlgunCasAnterior==0)
{ //Insertamos caracter no conflictivo.
  t[j]=c[i];
  j+=1;
}
}
for (int h=0;h<j;h++)
{
  resultado+=t[h];
}
return (resultado);
}

/*****
Hace falta este metodo, ya que los Strings definidos en la ontologia
no pueden contener caracteres en blanco, y a este agente les llega
codificados y este metodo los decodificca.
*****/
public String StringConEspaciosEnBlanco (String s)
{
  String prueba ="
+
          "
";
  char c[] = s.toCharArray();
  char t[] = prueba.toCharArray();
  String resultado="";

```

```

int AlgunCasAnterior = 0;

int j=0;

for (int i=0;i<s.length();i++)
{
    AlgunCasAnterior = 0;
    //Para recuperar espacios en blanco.

    //Para quitar el caracter en blanco inicial,
    //que introducimos antes para no tener problemas
    //si el usuario introducía un número en
    //lugar de un String en la interficie grafica. Daria problemas
    //en la ontologia, según como se haya definido el slot correspondiente.
    if ((c[i]=='/') && (c[i+1]=='_') && (i==0)) i=2;

    if ((c[i]=='/') && (c[i+1]=='_'))
    {
        t[j]=' ';
        j++;

        //Se sustituyen dos caracteres del string inicial por uno del
        //String final, por eso se incrementa la i.
        i++;

        AlgunCasAnterior = 1;
    }

    if ((c[i]=='/') && (c[i+1]=='ú'))
    {
        t[j]='u';
        j++;

        //Se sustituyen dos caracteres del string inicial por uno del
        //String final, por eso se incrementa la i.
        i++;

        AlgunCasAnterior = 1;
    }

    if ((c[i]=='/') && (c[i+1]=='$'))
    {
        t[j]='¥';
        j++;

        //Se sustituyen dos caracteres del string inicial por uno del
        //String final, por eso se incrementa la i.
        i++;

        AlgunCasAnterior = 1;
    }

    if ((c[i]=='%') && (c[i+1]=='/'))
    {
        t[j]='÷';
        j++;

        //Se sustituyen dos caracteres del string inicial por uno del
        //String final, por eso se incrementa la i.
        i++;

        AlgunCasAnterior = 1;
    }

    if ((c[i]=='&') && (c[i+1]=='/'))
    {
        t[j]='€';
        j++;
    }
}

```

```

        //Se sustituyen dos caracteres del string inicial por uno del
        //String final, por eso se incrementa la i.
        i++;

        AlgunCasAnterior = 1;
    }

    if (AlgunCasAnterior == 0)
    {
        t[j]=c[i];
        j+=1;
    }
}
for (int h=0;h<j;h++)
{
    resultado+=t[h];
}
return (resultado);
}

/*****
Metodo que coloca el ScrollBar vertical de JScrollPane, pasado
por parametro, en el punto maximo.
*****/
public void PosicionarFinalScrollBar (JScrollPane jsp)
{
    Point point= new Point();
    Dimension dim =jsp.getViewPort().getViewSize();
    point.y=dim.height;
    jsp.getViewPort().setViewPosition(point);
}

//Final classe.
}

```

## 8.2.3 Impresión de código contenido en ./Agent0/agentes/CTH/Ontology.

### 8.2.3.1 *OntologiaCTH.java*

```

/*****
    Agent0.agentes.CTH.Ontology.OntologiaCTH

    PFC Cristobal Ledesma Olias
    23/09/02

    Definición de la ontología: Objetos que representan
    frames y slots para que los agentes CTH y Agent0
    puedan intercambiar datos.

    *****/

package Agent0.agentes.CTH.Ontology;

import java.io.*;
import jade.onto.*;

import jade.onto.sl.sl.*;
import jade.onto.extensions.*;

//Definición de la ontología: Datos que han de recibir y enviar estos 2
agentes.
public class OntologiaCTH
{
    //nombre de la ontología.
    public static final String NAME="Ontologia_Trasplants_GruSMA";

    //*****Conceptos*****

    //Para datos en mensaje de CTH a Agent0.

    //Objectes definits en la ontologia.
    public static final String PACIENT="pacient";
    public static final String DADESCLINIQUES="dadescliniques";
    public static final String HOSPITAL="hospital";
    public static final String ADREZA="adreza";
    public static final String ORGAN="organ";
    public static final String OPERACIO="operacio";
    public static final String
    CARACTERISTIQUESBUSQUEDA="caracteristiquesbusqueda";

    public static final String ASSOCIACIO="associacio";
    public static final String RESULTATBUSQUEDA="resultatbusqueda";

    public static final String MORTPACIENT="mortpacient";

    //Acciones
    public static final String INTRODUIRPACIENT="introduirpacient";
    public static final String ANOTARMORTPACIENT="anotarmortpacient";
    public static final String
    TROBARPACIENTSCOMPATIBLES="trobarpatientscompatibles";
    public static final String ASSIGNACIOORGAN="assignacioorgan";
    public static final String ORGANREBUTXAT="organrebutxat";
    public static final String OPERACIOREALITZADA="operaciorealitzada";

    //Errores
    public static final String PACIENTINEXISTENT="patientinexistent";
    public static final String PACIENTMORT="patientmort";
    public static final String PACIENTTEORGANASSIGNAT="patientteorganassignat";
    public static final String PACIENTOPERAT="patientoperat";
    public static final String ORGANINEXISTENT="organinexistent";
    public static final String ORGANEXISTENT="organexistent";

```

```

public static final String
PACIENTSENSEORGANASSIGNAT="patientsenseorganassignat";
public static final String PACIENTSENSEOPERAR="patientsenseoperar";
public static final String ERRORACCESBD="erroraccesbd";
public static final String
PACIENTSENSEESPERARORGAN="patientsenseesperarorgan";
public static final String ORGANNODISPONIBLE="organnodisponible";
public static final String ORGANJAESTAVAREBUTXAT="organjaestavarebutxat";
public static final String HOSPITALSENSEAUTORIZAR="hospitalsenseautoritzar";

public static final String AGENTNOREGISTRAT="agentnoregistrat";
public static final String MASSA_OBJECTES="Massa-objectes";
public static final String MASSA_ATRIBUTS="Massa-atributs";
public static final String PESOS_NO_VALIDS="Pesos-no-valids";
public static final String CLASSES_INSUFICIENTS="Classes-insuficients";
public static final String ALGORITME_NO_IMPLEMENTAT="Algoritme-no-
implementat";

//Instancia de la ontologia para referenciar desde el agente
private static Ontology theInstanceCTH = new SL_DefaultOntologyImp();
public static OntologyHelper ontoCTH_helper = new OntologyHelper();

static{
    initInstance();
}

//Para consultar los conceptos de la ontologia
public static Ontology instanceCTH()
{
    return theInstanceCTH;
}

//Constructor
public OntologiaCTH(){
}

//Inicializa la ontologia
private static void initInstance(){
    try{
        //Objeto pacient: contiene todos los datos de un paciente
        theInstanceCTH.joinOntology(SL_Ontology.instance());
        theInstanceCTH.addRole(
            PACIENT,
            new SlotDescriptor[]{
                new
SlotDescriptor("CodiSS",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
                new
SlotDescriptor("Nom",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
                new
SlotDescriptor("DiaEntradaSistema",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,
Ontology.M),
                new
SlotDescriptor("MesEntradaSistema",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,
Ontology.M),
                new
SlotDescriptor("AnyEntradaSistema",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,
Ontology.M),
                new
SlotDescriptor("HoraEntradaSistema",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE
,Ontology.M),
            }
        );
    }
}

```

```

SlotDescriptor("MinutEntradaSistema",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYP
E,Ontology.M),
        new
SlotDescriptor("DiaMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.0
),
        new
SlotDescriptor("MesMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.0
),
        new
SlotDescriptor("AnyMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.0
),
        new
SlotDescriptor("HoraMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.
0),
        new
SlotDescriptor("MinutMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.0),
        new
SlotDescriptor("Estat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M
),
        new
        SlotDescriptor("Organo",Ontology.FRAME_SLOT,ORGAN,Ontology.M),
        new
SlotDescriptor("HospitalIngres",Ontology.FRAME_SLOT,HOSPITAL,Ontology.M),
        new
SlotDescriptor("Dades",Ontology.FRAME_SLOT,DADESCLINIQUES,Ontology.M),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new Pacient();
            }
            public Class getClassForRole(){
                return Pacient.class;
            }
        }
    };

    //Objeto Organ: Contiene todos los datos referentes a un Organ.
    theInstanceCTH.addRole(
        ORGAN,
        new SlotDescriptor[]{
            new
SlotDescriptor("IdOrgan",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology
.0),
            new
SlotDescriptor("Tipus",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M
),
            new
SlotDescriptor("Tamany",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.
M),
            new
SlotDescriptor("Estat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M
),
            },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new Organ();
            }
        }
    );

```

```

        }
        public Class getClassForRole(){
            return Organ.class;
        }
    }
};
//Objeto DadesCliniques: Contiene todos los datos clinicos de un
paciente.
theInstanceCTH.addRole(
    DADESCLINIQUES,
    new SlotDescriptor[]{
        new
SlotDescriptor("DiaNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("MesNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("AnyNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("Sexe",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M)
,
        new
SlotDescriptor("Alzada",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.
M),
        new
SlotDescriptor("Pes",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("TipusSang",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontolo
gy.M),
        new
SlotDescriptor("B0",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("B1",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("Dr0",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("Dr1",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new DadesCliniques();
        }
        public Class getClassForRole(){
            return DadesCliniques.class;
        }
    }
);

//Objeto Hospital: Datos Referentes a un hospital.
theInstanceCTH.addRole(
    HOSPITAL,
    new SlotDescriptor[]{
        new
SlotDescriptor("Nom",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        new
SlotDescriptor("Adreza",Ontology.FRAME_SLOT,ADREZA,Ontology.M),

```



```

    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Hospital();
        }
        public Class getClassForRole(){
            return Hospital.class;
        }
    }
);

//Objeto Adreza: Datos Referentes a la direccion de un hospital.
theInstanceCTH.addRole(
    ADREZA,
    new SlotDescriptor[]{
        new
SlotDescriptor("Carrer",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        new
SlotDescriptor("Numero",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        new
SlotDescriptor("Ciutat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        new
SlotDescriptor("CodiPostal",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("Província",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontolo
gy.M),
        new
SlotDescriptor("Pais",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M)
,
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Adreza();
        }
        public Class getClassForRole(){
            return Adreza.class;
        }
    }
);

//Objeto Operacio: Datos referentes a una operaciòn.
theInstanceCTH.addRole(
    OPERACIO,
    new SlotDescriptor[]{
        new
SlotDescriptor("DiaOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontolo
gy.M),
        new
SlotDescriptor("MesOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontolo
gy.M),
        new
SlotDescriptor("AnyOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontolo
gy.M),
        new
SlotDescriptor("HoraOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),

```

```

        new
SlotDescriptor("MinutOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("IdOrganTrasplantat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        new
SlotDescriptor("Hospital",Ontology.FRAME_SLOT,HOSPITAL,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Operacio();
        }
        public Class getClassForRole(){
            return Operacio.class;
        }
    }
);

//Objecto CaracteristiquesBusqueda: Objecto que contiene las
//caracteristicas de un organo, pera encontrar pacientes compatibles
//con dicho organo.

theInstanceCTH.addRole(
    CARACTERISTIQUESBUSQUEDA,
    new SlotDescriptor[]{
        new
SlotDescriptor("IdOrgan",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        new
SlotDescriptor("Distancia",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("Pes",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("Tamany",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("Antigens",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("Edat",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("TempsEspera",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("Funcio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("Agente",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new CaracteristiquesBusqueda();
        }
        public Class getClassForRole(){

```

```

        return CaracteristiquesBusqueda.class;
    }
};

//Objeto Resultado de encontrar pacientes compatibles con un organo
theInstanceCTH.addRole(
    RESULTATBUSQUEDA,
    new SlotDescriptor[]{
        new
SlotDescriptor("Assignacio",Ontology.SET_SLOT,ASSOCIACIO,Ontology.M),
        new
SlotDescriptor("NPacients",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.M),
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new ResultatBusqueda();
        }
        public Class getClassForRole(){
            return ResultatBusqueda.class;
        }
    }
);

//Objeto que indica un paciente y su grado de compatibilidad con
//un determinado organo.
theInstanceCTH.addRole(
    ASSOCIACIO,
    new SlotDescriptor[]{
        new
SlotDescriptor("Grau",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M)
,
        new
SlotDescriptor("CodiSS",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Associacio();
        }
        public Class getClassForRole(){
            return Associacio.class;
        }
    }
);

//Objeto MortPacient: Indica paciente y momento de su muerte.
theInstanceCTH.addRole(
    MORTPACIENT,
    new SlotDescriptor[]{
        //NUEVO SLOT: CodiSS.
        new
SlotDescriptor("CodiSS",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        //NUEVO SLOT: DiaMort
        new
SlotDescriptor("DiaMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M
),
        //NUEVO SLOT: MesMort
        new
SlotDescriptor("MesMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M
),

```

```

        //NUEVO SLOT: AnyMort
        new

SlotDescriptor("AnyMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology.M
),
        //NUEVO SLOT: HoraMort
        new

SlotDescriptor("HoraMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology.
M),
        //NUEVO SLOT: MinutMort
        new

SlotDescriptor("MinutMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology
.M),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new MortPacient();
            }
            public Class getClassForRole(){
                return MortPacient.class;
            }
        }
    );

//Objeto que indica un error porque paciente no existe.
theInstanceCTH.addRole(
    PACIENTINEXISTENT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientInexistent();
        }
        public Class getClassForRole(){
            return PacientInexistent.class;
        }
    }
);

//Objeto que indica un error porque organo ya existe.
theInstanceCTH.addRole(
    ORGANEXISTENT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganExistent();
        }
        public Class getClassForRole(){
            return OrganExistent.class;
        }
    }
);

//Objeto que indica un error porque paciente est muerto.
theInstanceCTH.addRole(
    PACIENTMORT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientMort();
        }
        public Class getClassForRole(){
            return PacientMort.class;
        }
    }
);

```

```

    }
);

//Objeto que indica error porque paciente tiene organo asignado.
theInstanceCTH.addRole(
    PACIENTTEORGANASSIGNAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientTeOrganAssignat();
        }
        public Class getClassForRole(){
            return PacientTeOrganAssignat.class;
        }
    }
);

//Objeto que indica error porque paciente no espera organo.
theInstanceCTH.addRole(
    PACIENTSENSEESPERARORGAN,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientSenseEsperarOrgan();
        }
        public Class getClassForRole(){
            return PacientSenseEsperarOrgan.class;
        }
    }
);

//Objeto que indica error porque paciente ya estaba operado.
theInstanceCTH.addRole(
    PACIENTOPERAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientOperat();
        }
        public Class getClassForRole(){
            return PacientOperat.class;
        }
    }
);

//Objeto que indica error porque no existe el organo.
theInstanceCTH.addRole(
    ORGANINEXISTENT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganInexistent();
        }
        public Class getClassForRole(){
            return OrganInexistent.class;
        }
    }
);

//Objeto que indica error porque organo esta asignado.
theInstanceCTH.addRole(
    ORGANNODISPONIBLE,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganNoDisponible();
        }
    }
);

```

```

        }
        public Class getClassForRole(){
            return OrganNoDisponible.class;
        }
    }
);

//Objeto que indica error porque paciente no tiene organo asignado.
theInstanceCTH.addRole(
    PACIENTSENSEORGANASSIGNAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientSenseOrganAssignat();
        }
        public Class getClassForRole(){
            return PacientSenseOrganAssignat.class;
        }
    }
);

//Objeto que indica error porque paciente est sin operar.
theInstanceCTH.addRole(
    PACIENTSENSEOPERAR,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientSenseOperar();
        }
        public Class getClassForRole(){
            return PacientSenseOperar.class;
        }
    }
);

//Objeto que indica error porque organo ya consta como rechazado.
theInstanceCTH.addRole(
    ORGANJAESTAVAREBUTXAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganJaEstavaRebutxat();
        }
        public Class getClassForRole(){
            return OrganJaEstavaRebutxat.class;
        }
    }
);

//Objeto que indica error porque hospital no esta autorizado a solicitar
//accion.
theInstanceCTH.addRole(
    HOSPITALSENSEAUTORITZAR,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new HospitalSenseAutoritzar();
        }
        public Class getClassForRole(){
            return HospitalSenseAutoritzar.class;
        }
    }
);

//CLUSDM:Objeto que indica un error por pesos no validos (no suman 100)

```

```

theInstanceCTH.addRole(
    PESOS_NO_VALIDS,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PesosNoValids();
        }
        public Class getClassForRole(){
            return PesosNoValids.class;
        }
    }
);

//CLUSDM: Objeto que indica un error por numero de pacientes excesivo
theInstanceCTH.addRole(
    MASSA_OBJECTES,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new MassaObjectes();
        }
        public Class getClassForRole(){
            return MassaObjectes.class;
        }
    }
);

//CLUSDM: Objeto que indica un error per numero de atributos excesivo
theInstanceCTH.addRole(
    MASSA_ATRIBUTS,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new MassaAtributs();
        }
        public Class getClassForRole(){
            return MassaAtributs.class;
        }
    }
);

//CLUSDM: Objeto que indica un error: no se han obtenido suficientes
clases
theInstanceCTH.addRole(
    CLASSES_INSUFICIENTS,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new ClassesInsuficients();
        }
        public Class getClassForRole(){
            return ClassesInsuficients.class;
        }
    }
);

//CLUSDM: Objeto que indica un error: no se ha implementado algoritmo
theInstanceCTH.addRole(
    ALGORITME_NO_IMPLEMENTAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){

            return new AlgoritmeNoImplementat();

```

```

    }
    public Class getClassForRole(){
        return AlgoritmeNoImplementat.class;
    }
}
);

//CLUSDM: Objeto que indica un error: no se ha implementado algoritmo
theInstanceCTH.addRole(
    AGENTNOREGISTRAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new AgentNoRegistrat();
        }
        public Class getClassForRole(){
            return AgentNoRegistrat.class;
        }
    }
);

//Objeto que indica error en acceso a BD.
theInstanceCTH.addRole(
    ERRORACCESBD,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new ErrorAccesBD();
        }
        public Class getClassForRole(){
            return ErrorAccesBD.class;
        }
    }
);

//Accio Introducir Paciente: Objeto que contiene informaci n
//que pasara AgentCTH a Agent  para realizar la accion
//introducir paciente
theInstanceCTH.addRole(
    INTRODUIRPACIENT,
    new SlotDescriptor[]{
        //NUEVO SLOT: Pacient.
        new
        SlotDescriptor(Ontology.FRAME_SLOT,PACIENT,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new IntroduirPacient();
        }
        public Class getClassForRole(){
            return IntroduirPacient.class;
        }
    }
);

//Objeto de la ontologia que contiene datos, que enviar AgentCTH a
//Agent , para que se realice la accion Anotar Muerte del Paciente.
theInstanceCTH.addRole(
    ANOTARMORTPACIENT,
    new SlotDescriptor[]{
        //NUEVO SLOT: MortPacient.
        new
        SlotDescriptor(Ontology.FRAME_SLOT,MORTPACIENT,Ontology.M),
        new
        SlotDescriptor(Ontology.FRAME_SLOT,HOSPITAL,Ontology.M),
    }
);

```



```

    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new AnotarMortPacient();
        }
        public Class getClassForRole(){
            return AnotarMortPacient.class;
        }
    }
);

//Objeto que enviar AgentCTH a Agent0, para que este realice la
//accion de encontrar pacientes compatibles con un organo.

theInstanceCTH.addRole(
    TROBARPACIENTSCOMPATIBLES,
    new SlotDescriptor[]{
        //NUEVO SLOT: CaracteristiquesBusqueda.
        new
SlotDescriptor(Ontology.FRAME_SLOT, CARACTERISTIQUESBUSQUEDA, Ontology.M),
        new
SlotDescriptor(Ontology.FRAME_SLOT, HOSPITAL, Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new TrobarPacientsCompatibles();
        }
        public Class getClassForRole(){
            return TrobarPacientsCompatibles.class;
        }
    }
);

//Objecto AssignacioOrgan: Indica paciente y organo que se le ha
//asignado.
//Cuando AgentCTH envíe este objeto a Agent0, Agent0 realizará la
//asignación del organo en Base de Datos.

theInstanceCTH.addRole(
    ASSIGNACIOORGAN,
    new SlotDescriptor[]{
        //NUEVO SLOT: CodiSS
        new
SlotDescriptor(Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.M),
        //NUEVO SLOT: IdOrgan
        new
SlotDescriptor(Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.M),
        new
SlotDescriptor(Ontology.FRAME_SLOT, HOSPITAL, Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new AssignacioOrgan();
        }
        public Class getClassForRole(){
            return AssignacioOrgan.class;
        }
    }
);

//Objeto OrganRebutxat: Indica paciente y organo que ha rechazado.
//Cuando AgentCTH envíe este objeto a Agent0, Agent0 anotará en
//en su base de datos que el organo a sido rechazado por paciente.
theInstanceCTH.addRole(
    ORGANREBUTXAT,
    new SlotDescriptor[]{
        //NUEVO SLOT: IdOrgan

```

```

        new
SlotDescriptor(Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        new
SlotDescriptor(Ontology.FRAME_SLOT,HOSPITAL,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganRebutxat();
        }
        public Class getClassForRole(){
            return OrganRebutxat.class;
        }
    }
);

//Objeto que enviara AgentCTH a Agent0, indicandole que se ha realizado
//un trasplante de un organo.
theInstanceCTH.addRole(
    OPERACIOREALITZADA,
    new SlotDescriptor[]{
        //NUEVO SLOT: Operacio
        new
SlotDescriptor(Ontology.FRAME_SLOT,OPERACIO,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OperacioRealitzada();
        }
        public Class getClassForRole(){
            return OperacioRealitzada.class;
        }
    }
);

}catch (OntologyException e){
    System.err.println("Error generant la Ontologia");
    e.printStackTrace();
}
}
}

```

### 8.2.3.2 Adreza.java

```
/*  
Agent0.Ontology
```

PFC Crist bal Ledesma Oljas  
25/09/02

```
Implementaci n del objeto Adreza definido en la ontologia  
*/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class Adreza extends Domain_OntologyObj  
{  
    //Variables  
    private String Carrer;  
    private String Numero;  
    private String Ciutat;  
    private String CodiPostal;  
    private String Provincia;  
    private String Pais;  
  
    //Constructor  
    public Adreza()  
    {  
        super("Adreza");  
        Carrer="No_inicializado";  
        Numero="No_inicializado";  
        Ciutat="No_inicializado";  
        CodiPostal="No_inicializado";  
        Provincia="No_inicializado";  
        Pais="No_inicializado";  
    }  
  
    public Adreza(String c,String n,String ciu,String cod,String pr,String pa)  
    {  
        super("Adreza");  
        Carrer=c;  
        Numero=n;  
        Ciutat=ciu;  
        CodiPostal=cod;  
        Provincia=pr;  
        Pais=pa;  
    }  
  
    //Metodos SET  
    public void setCarrer(String s)  
    {  
        this.Carrer=s;  
    }  
    public void setNumero(String s)  
    {  
        this.Numero=s;  
    }  
    public void setCiutat(String s)  
    {  
        this.Ciutat=s;  
    }  
    public void setCodiPostal(String s)  
    {  
        this.CodiPostal=s;  
    }  
    public void setProvincia(String s)  
    {  
        this.Provincia=s;  
    }  
    public void setPais(String s)
```

```
{
    this.Pais=s;
}
//Metodos GET
public String getCarrer()
{
    return this.Carrer;
}
public String getNumero()
{
    return this.Numero;
}
public String getCiutat()
{
    return this.Ciutat;
}
public String getCodiPostal()
{
    return this.CodiPostal;
}
public String getProvincia()
{
    return this.Provincia;
}
public String getPais()
{
    return this.Pais;
}
}
```

### 8.2.3.3 AgentNoRegistrat.java

```
/*  
CTHAg0Ontology.AgentNoRegistrat
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementaci3n del objeto AgentNoRegistrat definido en la ontologia  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class AgentNoRegistrat extends Domain_OntologyObj  
{  
    //Constructor  
    public AgentNoRegistrat()  
    {  
        super("AgentNoRegistrat");  
    }  
}
```

### 8.2.3.4 AlgoritmeNoImplementat.java

```
/*  
*****  
ontology.AlgoritmeNoImplementat
```

PFC  
David Sanchez Ruenes  
28-7-2001

Implementacio de l'error: AlgoritmeNoImplementat definit a l'ontologia

```
*****  
*****/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class AlgoritmeNoImplementat extends Domain_OntologyObj  
{  
  
    //Constructor  
    public AlgoritmeNoImplementat()  
    {  
        super ("Algoritme-no-implementat");  
    }  
  
}
```

### 8.2.3.5 AnotarMortPacient.java

```
/*  
CTHAg00Ontology.AnotarMortPacient
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto AnotarMortPaciente definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class AnotarMortPacient extends Domain_OntologyObj  
{  
    //Variables  
    private MortPacient MuertePaciente;  
    private Hospital Hosp;  
  
    //Constructor  
    public AnotarMortPacient()  
    {  
        super("AnotarMuertePaciente");  
        MuertePaciente=new MortPacient();  
        Hosp=new Hospital();  
    }  
    public AnotarMortPacient(MortPacient p, Hospital h)  
    {  
        super("AnotarMuertePaciente");  
        MuertePaciente=p;  
        Hosp=h;  
    }  
  
    //Metodos SET  
    public void set_0(MortPacient p)  
    {  
        this.MuertePaciente=p;  
    }  
  
    public void set_1(Hospital h)  
    {  
        this.Hosp=h;  
    }  
  
    //Metodos GET  
    public MortPacient get_0()  
    {  
        return this.MuertePaciente;  
    }  
    public Hospital get_1()  
    {  
        return this.Hosp;  
    }  
}
```

### 8.2.3.6 *AssignacioOrgan.java*

/\*\*\*\*\*

CTHAg00ontology.AssignacioOrgan

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto AssignacioOrgan definido en la ontología  
\*\*\*\*\*/

```
package Agent0.agentes.CTH.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

public class AssignacioOrgan extends Domain_OntologyObj
{
    //Variables
    private String CodiSS;
    private String IdOrgan;
    private Hospital hosp;
    //Constructor
    public AssignacioOrgan()
    {
        super("AssignacioOrgan");
        CodiSS="no_inicializado";
        IdOrgan="no_inicializado";
        hosp=new Hospital();
    }
    public AssignacioOrgan(String c, String i, Hospital h)
    {
        super("AssignacioOrgan");
        CodiSS=c;
        IdOrgan=i;
        hosp=h;
    }

    //Metodos SET
    public void set_0(String s)
    {
        this.CodiSS=s;
    }
    public void set_1(String s)
    {
        this.IdOrgan=s;
    }
    public void set_2(Hospital h)
    {
        this.hosp=h;
    }

    //Metodos GET
    public String get_0()
    {
        return this.CodiSS;
    }
    public String get_1()
    {
        return this.IdOrgan;
    }
    public Hospital get_2()
    {
        return this.hosp;
    }
}
```

### 8.2.3.7 Associacio.java.

```
/*  
CTHAg00Ontology.Associacio
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto Associacio definido en la ontología,  
que informa sobre el resultado de la acción  
EncontrarPacientesCompatibles con un órgano. Contiene lista  
de pacientes con su respectiva valoración de compatibilidad  
con órgano.

```
*/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class Associacio extends Domain_OntologyObj  
{  
    //Variables  
    private String Grau = new String("");  
    private String CodiSS = new String("");  
  
    //Constructor  
    public Associacio()  
    {  
        super("Associacio");  
        Grau="no_inicializado";  
        CodiSS="no_inicializado";  
    }  
  
    public Associacio(String g, String c){  
        super("Associacio");  
        Grau=g;  
        CodiSS=c;  
    }  
  
    //Metodos SET  
    public void setGrau (String g)  
    {  
        this.Grau=g;  
    }  
  
    public void setCodiSS (String c)  
    {  
        this.CodiSS=c;  
    }  
  
    //Metodos GET  
    public String getGrau()  
    {  
        return (this.Grau);  
    }  
  
    public String getCodiSS()  
    {  
        return (this.CodiSS);  
    }  
}
```



### 8.2.3.8 CaracteristiquesBusqueda.java

```
/*  
CTHAg00Ontology.CaracteristiquesBusqueda
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto CaracteristiquesBusqueda definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
public class CaracteristiquesBusqueda extends Domain_OntologyObj  
{  
    //Variables  
    private Double Distancia;  
    private Double Pes;  
    private Double Tamany;  
    private Double Antigens;  
    private Double Edat;  
    private Double TempsEspera;  
    private String IdOrgan;  
    private Long Funcio;  
    private String Agente;  
  
    //Constructor  
    public CaracteristiquesBusqueda()  
    {  
        super("CaracteristiquesBusqueda");  
        IdOrgan="no_inicializado";  
        Distancia=new Double(0.0);  
        Pes=new Double(0.0);  
        Tamany=new Double(0.0);  
        Antigens=new Double(0.0);  
        Edat=new Double(0.0);  
        TempsEspera=new Double(0.0);  
        Funcio=new Long(0);  
        Agente="no_inicializado";  
    }  
    public CaracteristiquesBusqueda (String i,Double d,  
    Double p,Double t,Double a,Double e,  
    Double te,Long f,String ag)  
    {  
        super("Operacio");  
        IdOrgan=i;  
        Distancia=d;  
        Pes=p;  
        Tamany=t;  
        Antigens=a;  
        Edat=e;  
        TempsEspera=te;  
        Funcio=f;  
        Agente=ag;  
    }  
    //Metodos SET  
    public void setDistancia(Double d)  
    {  
        this.Distancia=d;  
    }  
    public void setPes(Double m)  
    {  
        this.Pes=m;  
    }  
    public void setTamany(Double a)  
    {  
        this.Tamany=a;  
    }  
    public void setAntigens(Double h)
```

```

    {
        this.Antigens=h;
    }
    public void setEdat(Double m)
    {
        this.Edat=m;
    }
    public void setTempsEspera(Double i)
    {
        this.TempsEspera=i;
    }
    public void setIdOrgan(String c)
    {
        this.IdOrgan=c;
    }
    public void setFuncio(Long i)
    {
        this.Funcio=i;
    }
    public void setAgente(String c)
    {
        this.Agente=c;
    }

    //Metodos GET
    public Double getDistancia()
    {
        return this.Distancia;
    }
    public Double getPes()
    {
        return this.Pes;
    }
    public Double getTamany()
    {
        return this.Tamany;
    }
    public Double getAntigens()
    {
        return this.Antigens;
    }
    public Double getEdat()
    {
        return this.Edat;
    }
    public Double getTempsEspera()
    {
        return this.TempsEspera;
    }
    public String getIdOrgan()
    {
        return this.IdOrgan;
    }
    public Long getFuncio()
    {
        return this.Funcio;
    }
    public String getAgente()
    {
        return this.Agente;
    }
}

```

### 8.2.3.9 *ClassesInsuficients.java*

```

/*****
*****
ontology.ClassesInsuficients

PFC
David Sanchez Ruenes
28-7-2001

Classe del projecte David Sanchez utilitzat tambe per Agent CTH.

Implementacio de l'objecte ClassesInsuficients (error) definit a l'ontologia
*****
*****/

package Agent0.agentes.CTH.Ontology;

import jade.onto.sl.sl.*;

//Implementacio de l'objecte ClassesInsuficients (error) definit a l'ontologia
public class ClassesInsuficients extends Domain_OntologyObj
{
    //Constructor
    public ClassesInsuficients()
    {
        super ("Classes-insuficients");
    }
}

```

### 8.2.3.10 *DadesCliniques.java*

```

/*****
CTHAg00Ontology.DadesCliniques

PFC Crist bal Ledesma Oljas
25/09/02

Implementaci n del objeto DadesCliniques definido en la ontologia
*****/

package Agent0.agentes.CTH.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

public class DadesCliniques extends Domain_OntologyObj
{
    //Variables
    private Long DiaNaixement;
    private Long MesNaixement;
    private Long AnyNaixement;
    private String Sexe;
    private Double Alzada;
    private Double Pes;
    private String TipusSang;
    private Long B0;
    private Long B1;
    private Long Dr0;
    private Long Dr1;

    //Constructor
    public DadesCliniques()
    {
        super("DadesCliniques");
        DiaNaixement=new Long(0);
        MesNaixement=new Long(0);
        AnyNaixement=new Long(0);
        Sexe="no_inicializado";
        Alzada=new Double(0.0);
        Pes=new Double(0.0);
        TipusSang="no_inicializado";
        B0=new Long(0);
        B1=new Long(0);
        Dr0=new Long(0);
        Dr1=new Long(0);
    }

    public DadesCliniques(Long dn,Long mn,Long an, String s,Double al,Double pe,
    String ts,Long bc,Long bu,Long drc, Long dru)
    {
        super("DadesCliniques");
        DiaNaixement=dn;
        MesNaixement=mn;
        AnyNaixement=an;
        Sexe=s;
        Alzada=al;
        Pes=pe;
        TipusSang=ts;
        B0=bc;
        B1=bu;
        Dr0=drc;
        Dr1=dru;
    }

    //Metodos SET
    public void setDiaNaixement(Long d)
    {
        this.DiaNaixement=d;
    }
    public void setMesNaixement(Long m)

```

```

{
    this.MesNaixement=m;
}
public void setAnyNaixement(Long a)
{
    this.AnyNaixement=a;
}
public void setSexe (String s)
{
    this.Sexe=s;
}
public void setPes (Double p)
{
    this.Pes=p;
}
public void setAlzada (Double a)
{
    this.Alzada=a;
}
public void setTipusSang (String s)
{
    this.TipusSang=s;
}
public void setB0 (Long l)
{
    this.B0=l;
}
public void setB1 (Long l)
{
    this.B1=l;
}
public void setDr0 (Long l)
{
    this.Dr0=l;
}
public void setDr1 (Long l)
{
    this.Dr1=l;
}

//Metodos GET
public Long getDiaNaixement()
{
    return this.DiaNaixement;
}
public Long getMesNaixement()
{
    return this.MesNaixement;
}
public Long getAnyNaixement()
{
    return this.AnyNaixement;
}
public String getSexe ()
{
    return this.Sexe;
}
public Double getPes ()
{
    return this.Pes;
}
public Double getAlzada ()
{
    return this.Alzada;
}
public String getTipusSang ()
{
    return this.TipusSang;
}
}

```

```
public Long getB0 ()
{
    return this.B0;
}
public Long getB1 ()
{
    return this.B1;
}
public Long getDr0 ()
{
    return this.Dr0;
}
public Long getDr1 ()
{
    return this.Dr1;
}
}
```

### 8.2.3.11 *Domain\_OntologyObj.java*

```

/*****
*****
ontology.Domain_OntologyObj

PFC Agent0
Cristóbal Ledesma Olías
7/10/2002
(ESTE ARCHIVO HA SIDO PROPORCIONADO POR TUTORES: TONI y AØDA)

Implementa interface OntologyEntity i dona suport a variables
*****/

package Agent0.agentes.CTH.Ontology;

import jade.onto.sl.sl.*;
import jade.onto.extensions.*;
import java.util.List;

//Implementa interface OntologyEntity i da suporte a variables
public class Domain_OntologyObj implements OntologyEntity
{
    //variables
    public String role_name=OntologyEntity.UNSET_MARKER;

    //constructor
    public Domain_OntologyObj (String rname)
    {
        role_name=rname;
    }

    //metodos GET
    public String getRole()
    {
        return role_name;
    }

    //Permite insertar i consultar variables de slots
    private VariableManager vh=new VariableManager();

    public VariableManager VarManager()
    {
        return vh;
    }
}

```

### 8.2.3.12 *ErrorAccesBD.java*

```
/*  
CTHAg0Ontology.ErrorAccesBD
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto ErrorAccesBD definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class ErrorAccesBD extends Domain_OntologyObj  
{  
    //Constructor  
    public ErrorAccesBD()  
    {  
        super("ErrorAccesBD");  
    }  
}
```



### 8.2.3.13 *Hospital.java*

```
/******  
CTHAg00Ontology.Hospital
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto Hospital definido en la ontología  
\*\*\*\*\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class Hospital extends Domain_OntologyObj  
{  
    //Variables  
    private String Nom = new String();  
    private Adreza Adr;  
  
    //Constructor  
    public Hospital()  
    {  
        super("Hospital");  
        Nom="no_inicializado";  
        Adr=new Adreza();  
    }  
  
    public Hospital(String n, Adreza a)  
    {  
        super("Hospital");  
        Nom=n;  
        Adr=a;  
    }  
  
    //Metodos SET  
    public void setNom (String n)  
    {  
        this.Nom=n;  
    }  
  
    public void setAdreza (Adreza a)  
    {  
        this.Adr=a;  
    }  
  
    //Metodos GET  
    public String getNom ()  
    {  
        return this.Nom;  
    }  
  
    public Adreza getAdreza ()  
    {  
        return this.Adr;  
    }  
}
```

#### 8.2.3.14 *HospitalSenseAutoritzar.java*

```
/*  
CTHAg00ontology.HospitalSenseAutoritzar
```

```
PFC Crist bal Ledesma Oljas  
25/09/02
```

```
Implementaci n del objeto HospitalSenseAutoritzar definido en la ontologia  
*/
```

```
package Agent0.agentes.CTH.Ontology;
```

```
import jade.onto.sl.sl.*;
```

```
public class HospitalSenseAutoritzar extends Domain_OntologyObj  
{  
    //Constructor  
    public HospitalSenseAutoritzar()  
    {  
        super("HospitalSenseAutoritzar");  
    }  
}
```

### 8.2.3.15 *IntroduirPacient.java*

```
/*  
CTHAg00ontology.IntroduirPacient
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto IntroduirPacient definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class IntroduirPacient extends Domain_OntologyObj  
{  
    //Variables  
    private Pacient Paciente;  
  
    //Constructor  
    public IntroduirPacient()  
    {  
        super("IntroduirPacient");  
        Paciente=new Pacient();  
    }  
    public IntroduirPacient(Pacient p)  
    {  
        super("IntroduirPacient");  
        Paciente=p;  
    }  
  
    //Metodos SET  
    public void set_0(Pacient p)  
    {  
        this.Paciente=p;  
    }  
  
    //Metodos GET  
    public Pacient get_0()  
    {  
        return this.Paciente;  
    }  
}
```

### 8.2.3.16 *MassaAtributs.java*

```
/*  
*****  
ontology.MassaAtributs
```

PFC  
David Sanchez Ruenes  
28-7-2001

Classe del projecte David Sanchez utilitzat tambe per Agent CTH.

Implementacio de l'objecte: MassaAtributs definit a l'ontologia

```
*****  
*****/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class MassaAtributs extends Domain_OntologyObj  
{  
  
    //Constructor  
    public MassaAtributs()  
    {  
        super ("Massa-atributs");  
    }  
  
}
```

### 8.2.3.17 *MassaObjectes.java*

```
/*  
*****  
ontology.MassaObjectes
```

PFC  
David Sanchez Ruenes  
28-7-2001

Classe del projecte David Sanchez utilitzat tambe per Agent CTH.

Implementacio de l'error: MassaObjectes definit a l'ontologia

```
*****  
*****/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class MassaObjectes extends Domain_OntologyObj  
{  
  
    //Constructor  
    public MassaObjectes()  
    {  
        super ("Massa-objectes");  
    }  
  
}
```

### 8.2.3.18 *MortPacient.java*

```
/******  
CTHAg00Ontology.MortPacient
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto MortPacient definido en la ontología  
\*\*\*\*\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class MortPacient extends Domain_OntologyObj  
{  
    //Variables  
    private String CodiSS;  
    private Long DiaMort;  
    private Long MesMort;  
    private Long AnyMort;  
    private Long HoraMort;  
    private Long MinutMort;  
  
    //Constructor  
    public MortPacient()  
    {  
        super("MortPacient");  
        CodiSS="no_inicializado";  
        DiaMort=new Long(0);  
        MesMort=new Long(0);  
        AnyMort=new Long(0);  
        HoraMort=new Long(0);  
        MinutMort=new Long(0);  
    }  
  
    public MortPacient(String s,Long dm,Long mesm,Long am,Long hm,Long mm)  
    {  
        super("MortPacient");  
        CodiSS=s;  
        DiaMort=dm;  
        MesMort=mesm;  
        AnyMort=am;  
        HoraMort=hm;  
        MinutMort=mm;  
    }  
  
    //Metodos SET  
    public void setCodiSS(String s)  
    {  
        this.CodiSS=s;  
    }  
    public void setDiaMort(Long l)  
    {  
        this.DiaMort=l;  
    }  
    public void setMesMort(Long l)  
    {  
        this.MesMort=l;  
    }  
    public void setAnyMort(Long l)  
    {  
        this.AnyMort=l;  
    }  
    public void setHoraMort(Long l)  
    {  
        this.HoraMort=l;  
    }  
    public void setMinutMort(Long l)
```

```

    {
        this.MinutMort=l;
    }
    //Metodos GET
    public String getCodiSS()
    {
        return this.CodiSS;
    }
    public Long getDiaMort()
    {
        return this.DiaMort;
    }
    public Long getMesMort()
    {
        return this.MesMort;
    }
    public Long getAnyMort()
    {
        return this.AnyMort;
    }
    public Long getHoraMort()
    {
        return this.HoraMort;
    }
    public Long getMinutMort()
    {
        return this.MinutMort;
    }
}

```

### 8.2.3.19 *Operacio.java*

```
/*  
CTHAg00ontology.Operacio
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto Operacio definido en la ontología

```
*/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
public class Operacio extends Domain_OntologyObj  
{  
    //Variables  
    private Long DiaOperacio;  
    private Long MesOperacio;  
    private Long AnyOperacio;  
    private Long HoraOperacio;  
    private Long MinutOperacio;  
    private String IdOrganTrasplantat;  
    private Hospital HospitalOperacio;  
  
    //Constructor  
    public Operacio()  
    {  
        super("Operacio");  
        DiaOperacio=new Long(0);  
        MesOperacio=new Long(0);  
        AnyOperacio=new Long(0);  
        HoraOperacio=new Long(0);  
        MinutOperacio=new Long(0);  
        IdOrganTrasplantat="no_inicializado";  
        HospitalOperacio=new Hospital();  
    }  
    public Operacio (Long dop,Long mo,Long ao,Long ho,Long mio,String i,  
        Hospital hos)  
    {  
        super("Operacio");  
        DiaOperacio=dop;  
        MesOperacio=mo;  
        AnyOperacio=ao;  
        HoraOperacio=ho;  
        MinutOperacio=mio;  
        IdOrganTrasplantat=i;  
        HospitalOperacio=hos;  
    }  
    //Metodos SET  
    public void setDiaOperacio(Long d)  
    {  
        this.DiaOperacio=d;  
    }  
    public void setMesOperacio(Long m)  
    {  
        this.MesOperacio=m;  
    }  
    public void setAnyOperacio(Long a)  
    {  
        this.AnyOperacio=a;  
    }  
    public void setHoraOperacio(Long h)  
    {  
        this.HoraOperacio=h;  
    }  
    public void setMinutOperacio(Long m)  
    {  
        this.MinutOperacio=m;  
    }  
}
```

```

public void setIdOrganTrasplantat(String i)
{
    this.IdOrganTrasplantat=i;
}
public void setHospital(Hospital h)
{
    this.HospitalOperacio=h;
}

//Metodos GET
public Long getDiaOperacio()
{
    return this.DiaOperacio;
}
public Long getMesOperacio()
{
    return this.MesOperacio;
}
public Long getAnyOperacio()
{
    return this.AnyOperacio;
}
public Long getHoraOperacio()
{
    return this.HoraOperacio;
}
public Long getMinutOperacio()
{
    return this.MinutOperacio;
}
public String getIdOrganTrasplantat()
{
    return this.IdOrganTrasplantat;
}
public Hospital getHospital()
{
    return this.HospitalOperacio;
}
}

```



### 8.2.3.20 *OperacioRealitzada.java*

```
/*  
CTHAg00ontology.OperacioRealitzada
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto OperacioRealitzada definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class OperacioRealitzada extends Domain_OntologyObj  
{  
    //Variables  
    private Operacio Operacion;  
  
    //Constructor  
    public OperacioRealitzada()  
    {  
        super("OperacioRealitzada");  
        Operacion=new Operacio();  
    }  
    public OperacioRealitzada(Operacio o)  
    {  
        super("OperacioRealitzada");  
        Operacion=o;  
    }  
  
    //Metodos SET  
    public void set_0(Operacio o)  
    {  
        this.Operacion=o;  
    }  
    //Metodos GET  
    public Operacio get_0()  
    {  
        return this.Operacion;  
    }  
}
```

### 8.2.3.21 *Organ.java*

```
/*  
CTHAg00ontology.Organ
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto Organ definido en la ontología

```
*/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class Organ extends Domain_OntologyObj  
{  
    //Variables  
    private String Tipus;  
    private String IdOrgan;  
    private Double Tamany;  
    private String Estat;  
  
    //Creo que este slot no hace falta, pero no lo quito por ahora.  
    //private String Estat;  
  
    //Constructor  
    public Organ()  
    {  
        super("Organ");  
        Tipus="no_inicializado";  
        IdOrgan="no_inicializado";  
        Tamany=new Double(0.0);  
        Estat="no_inicializado";  
    }  
  
    public Organ(String id,String tip,Double tam,String est)  
    {  
        super("Organ");  
        Tipus=tip;  
        IdOrgan=id;  
        Tamany=tam;  
        Estat=est;  
    }  
  
    //Metodos SET  
    public void setTipus(String tip)  
    {  
        this.Tipus=tip;  
    }  
    public void setIdOrgan(String id)  
    {  
        this.IdOrgan=id;  
    }  
    public void setTamany(Double tam)  
    {  
        this.Tamany=tam;  
    }  
  
    public void setEstat(String est)  
    {  
        this.Estat=est;  
    }  
  
    //Metodos GET  
    public String getTipus()  
    {  
        return this.Tipus;  
    }  
}
```

```
public String getIdOrgan()  
{  
    return this.IdOrgan;  
}  
public Double getTamany()  
{  
    return this.Tamany;  
}  
  
public String getEstat()  
{  
    return this.Estat;  
}  
}
```

### 8.2.3.22      *OrganExistent.java*

```
/*  
CTHAg00Ontology.OrganExistent
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto OrganExistent definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class OrganExistent extends Domain_OntologyObj  
{  
    //Constructor  
    public OrganExistent()  
    {  
        super("OrganExistent");  
    }  
}
```

### 8.2.3.23      *OrganInexistent.java*

```
/*  
CTHAg00Ontology.OrganInexistent
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto OrganInexistent definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class OrganInexistent extends Domain_OntologyObj  
{  
    //Constructor  
    public OrganInexistent()  
    {  
        super("OrganInexistent");  
    }  
}
```

### 8.2.3.24 *OrganJaEstavaRebutxat.java*

```
/*  
CTHAg0Ontology.OrganJaEstavaRebutxat
```

PFC Crist bal Ledesma Oljas  
25/09/02

Implementaci n del objeto OrganJaEstavaRebutxat definido en la ontologia  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class OrganJaEstavaRebutxat extends Domain_OntologyObj  
{  
    //Constructor  
    public OrganJaEstavaRebutxat()  
    {  
        super("OrganJaEstavaRebutxat");  
    }  
}
```

### 8.2.3.25 *OrganNoDisponible.java*

```
/*  
CTHAg0Ontology.OrganNoDisponible
```

PFC Crist bal Ledesma Oljas  
25/09/02

Implementaci n del objeto OrganNoDisponible definido en la ontologia  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class OrganNoDisponible extends Domain_OntologyObj  
{  
    //Constructor  
    public OrganNoDisponible()  
    {  
        super("OrganNoDisponible");  
    }  
}
```

### 8.2.3.26 *OrganRebutxat.java*

```
/*  
CTHAg0Ontology.OrganRebutxat
```

PFC Cristóbal Ledesma Oljas  
25/09/02

```
Implementaci3n del objeto OrganRebutxat definido en la ontologia  
*/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class OrganRebutxat extends Domain_OntologyObj  
{  
    //Variables  
    private String IdOrgan;  
    private Hospital hosp;  
    //Constructor  
    public OrganRebutxat()  
    {  
        super("OrganRebutxat");  
        IdOrgan="no_inicializado";  
        hosp = new Hospital();  
    }  
    public OrganRebutxat(String i, Hospital h)  
    {  
        super("OrganRebutxat");  
        IdOrgan=i;  
        hosp=h;  
    }  
  
    //Metodos SET  
    public void set_0(String s)  
    {  
        this.IdOrgan=s;  
    }  
    public void set_1(Hospital h)  
    {  
        this.hosp=h;  
    }  
  
    //Metodos GET  
    public String get_0()  
    {  
        return this.IdOrgan;  
    }  
    public Hospital get_1()  
    {  
        return this.hosp;  
    }  
}
```

### 8.2.3.27 *Pacient.java*

```
/*  
CTHAg00ontology.Pacient
```

PFC Cristóbal Ledesma Olías  
25/09/02

Implementación del objeto Pacient definido en la ontología

```
*/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class Pacient extends Domain_OntologyObj  
{  
  
    //Variables  
    private String CodiSS=new String("");  
    private String Nom=new String("");  
    private Long DiaEntradaSistema;  
    private Long MesEntradaSistema;  
    private Long AnyEntradaSistema;  
    private Long HoraEntradaSistema;  
    private Long MinutEntradaSistema;  
    private Long DiaMort;  
    private Long MesMort;  
    private Long AnyMort;  
    private Long HoraMort;  
    private Long MinutMort;  
    private String Estat;  
  
    private Organ Organo;  
    private Hospital HospitalIngres;  
    private DadesCliniques Dades;  
  
    //Constructor  
    public Pacient()  
    {  
        super("Pacient");  
        CodiSS="no_inicializado";  
        Nom="no_inicializado";  
        DiaEntradaSistema=new Long(0);  
        MesEntradaSistema=new Long(0);  
        AnyEntradaSistema=new Long(0);  
        HoraEntradaSistema=new Long(0);  
        MinutEntradaSistema=new Long(0);  
        DiaMort=null;  
        MesMort=null;  
        AnyMort=null;  
        HoraMort=null;  
        MinutMort=null;  
        Estat="no_inicializado";  
        Organo=new Organ();  
        HospitalIngres= new Hospital();  
        Dades= new DadesCliniques();  
    }  
  
    public Pacient(String Codi,String n,Long dsist,Long messist,  
Long asist,Long hsist,Long msist,Long dmort,Long mesmort,  
Long amort,Long hmort,Long mmort,String est,Organ org,Hospital hos,  
DadesCliniques dc){  
        super("Pacient");  
        CodiSS=Codi;  
        Nom=n;  
        DiaEntradaSistema=dsist;  
        MesEntradaSistema=messist;  
        AnyEntradaSistema=asist;  
        HoraEntradaSistema=hsist;
```

```

        MinutEntradaSistema=msist;
        DiaMort=dmort;
        MesMort=mesmort;
        AnyMort=amort;
        HoraMort=hmort;
        MinutMort=mmort;
        Estat=est;
        Organo=org;
        HospitalIngres=hos;
        Dades=dc;
    }

    public Pacient(String Codi,String n,Long dsist,Long messist,Long asist,
    Long hsist,Long msist,String est,Organ org,Hospital hos,DadesCliniques
    dc){
        super("Pacient");
        CodiSS=Codi;
        Nom=n;
        DiaEntradaSistema=dsist;
        MesEntradaSistema=messist;
        AnyEntradaSistema=asist;
        HoraEntradaSistema=hsist;
        MinutEntradaSistema=msist;

        DiaMort=null;
        MesMort=null;
        AnyMort=null;
        HoraMort=null;
        MinutMort=null;

        Estat=est;
        Organo=org;
        HospitalIngres=hos;
        Dades=dc;
    }

    //Metodos SET
    public void setOrgano (Organ org)
    {
        this.Organo=org;
    }

    public void setHospitalIngres (Hospital hosp)
    {
        this.HospitalIngres=hosp;
    }

    /** Es un metode set en realitat, ja que no es una llista.
    Pero l'hem d'implementar aixi.*/
    public void setDades (DadesCliniques d)
    {
        this.Dades=d;
    }

    //Metodos GET
    /**L'hem d'implementar aixi.*/
    public Organ getOrgano()
    {
        return this.Organo;
    }

    public Hospital getHospitalIngres ()
    {
        return this.HospitalIngres;
    }

    public DadesCliniques getDades()
    {
        return this.Dades;
    }

```



```

}

//Metodos SET de datos primitivos.
public void setCodiSS(String cod)
{
    this.CodiSS=cod;
}
public void setNom(String n)
{
    this.Nom=n;
}
public void setDiaEntradaSistema(Long dia)
{
    this.DiaEntradaSistema=dia;
}
public void setMesEntradaSistema(Long mes)
{
    this.MesEntradaSistema=mes;
}
public void setAnyEntradaSistema(Long any)
{
    this.AnyEntradaSistema=any;
}
public void setHoraEntradaSistema(Long hora)
{
    this.HoraEntradaSistema=hora;
}
public void setMinutEntradaSistema(Long min)
{
    this.MinutEntradaSistema=min;
}
public void setDiaMort(Long dia)
{
    this.DiaMort=dia;
}
public void setMesMort(Long mes)
{
    this.MesMort=mes;
}
public void setAnyMort(Long any)
{
    this.AnyMort=any;
}

public void setHoraMort(Long hora)
{
    this.HoraMort=hora;
}
public void setMinutMort(Long min)
{
    this.MinutMort=min;
}
public void setEstat(String e)
{
    this.Estat=e;
}

//Metodos get de los datos primitivos.
public String getCodiSS()
{
    return this.CodiSS;
}
public String getNom()
{
    return this.Nom;
}
public Long getDiaEntradaSistema()
{

```

```

        return this.DiaEntradaSistema;
    }
    public Long getMesEntradaSistema()
    {
        return this.MesEntradaSistema;
    }
    public Long getAnyEntradaSistema()
    {
        return this.AnyEntradaSistema;
    }
    public Long getHoraEntradaSistema()
    {
        return this.HoraEntradaSistema;
    }
    public Long getMinutEntradaSistema()
    {
        return this.MinutEntradaSistema;
    }
    public Long getDiaMort()
    {
        return this.DiaMort;
    }
    public Long getMesMort()
    {
        return this.MesMort;
    }
    public Long getAnyMort()
    {
        return this.AnyMort;
    }
    public Long getHoraMort()
    {
        return this.HoraMort;
    }
    public Long getMinutMort()
    {
        return this.MinutMort;
    }
    public String getEstat()
    {
        return this.Estat;
    }
}

```

### 8.2.3.28 *PacientInexistent.java*

```
/*  
CTHAg00Ontology.PacientInexistent
```

PFC Cristóbal Ledesma Olías  
25/09/02

Implementación del objeto PacientInexistent definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class PacientInexistent extends Domain_OntologyObj  
{  
    //Constructor  
    public PacientInexistent()  
    {  
        super("PacientInexistent");  
    }  
}
```

### 8.2.3.29 *PacientMort.java*

```
/*  
CTHAg00Ontology.PacientMort
```

PFC Cristóbal Ledesma Olías  
25/09/02

Implementación del objeto PacientMort definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class PacientMort extends Domain_OntologyObj  
{  
    //Constructor  
    public PacientMort()  
    {  
        super("PacientMort");  
    }  
}
```

### 8.2.3.30 *PacientOperat.java*

```
/*  
CTHAg00Ontology.PacientOperat
```

PFC Cristóbal Ledesma Olías  
25/09/02

Implementación del objeto PacientOperat definido en la ontología  
\*/

```
package Agent0.agentes.CTH.Ontology;  
import jade.onto.sl.sl.*;  
public class PacientOperat extends Domain_OntologyObj  
{  
    //Constructor  
    public PacientOperat()  
    {  
        super("PacientOperat");  
    }  
}
```

### 8.2.3.31 *PacientSenseEsperarOrgan.java*

```
/******  
CTHAg0Ontology.PacientSenseEsperarOrgan  
PFC Crist bal Ledesma Oljas  
25/09/02
```

```
Implementaci n del objeto PacientSenseEsperarOrgan definido en la ontologia  
*****/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class PacientSenseEsperarOrgan extends Domain_OntologyObj  
{  
    //Constructor  
    public PacientSenseEsperarOrgan()  
    {  
        super("PacientSenseEsperarOrgan");  
    }  
}
```

### 8.2.3.32 *PacientSenseOperar.java*

```
/******  
CTHAg0Ontology.PacientSenseOperar  
  
PFC Crist bal Ledesma Oljas  
25/09/02
```

```
Implementaci n del objeto PacientSenseOperar  
definido en la ontologia  
*****/
```

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class PacientSenseOperar extends Domain_OntologyObj  
{  
    //Constructor  
    public PacientSenseOperar()  
    {  
        super("PacientSenseOperar");  
    }  
}
```

### 8.2.3.33 *PacientSenseOrganAssignat.java*

```
/*  
Agent0.Ontology.PacientSenseOrganAssignat
```

PFC Crist bal Ledesma Oljas  
25/09/02

Implementaci n del objeto PacientSenseOrganAssignat definido en la ontologia  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class PacientSenseOrganAssignat extends Domain_OntologyObj  
{  
    //Constructor  
    public PacientSenseOrganAssignat()  
    {  
        super("PacientSenseOrganAssignat");  
    }  
}
```

### 8.2.3.34 *PacientTeOrganAssignat.java*

```
/*  
CTHAg0Ontology.PacientTeOrganAssignat
```

PFC Crist bal Ledesma Oljas  
25/09/02

Implementaci n del objeto PacientTeOrganAssignat definido en la ontologia  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
  
public class PacientTeOrganAssignat extends Domain_OntologyObj  
{  
    //Constructor  
    public PacientTeOrganAssignat()  
    {  
        super("PacientTeOrganAssignat");  
    }  
}
```

### 8.2.3.35 *PesosNoValids.java*

```

/*****
*****
ontology.PesosNoValids

PFC
David Sanchez Ruenes
28-7-2001

Classe del projecte David Sanchez utilitzat tambe per Agent CTH.

Implementacio de l'error: PesosNoValids definit a l'ontologia
*****/

package Agent0.agentes.CTH.Ontology;

import jade.onto.sl.sl.*;

public class PesosNoValids extends Domain_OntologyObj
{
    //Constructor
    public PesosNoValids()
    {
        super ("Pesos-no-valids");
    }
}

```

### 8.2.3.36 *ResultatBusqueda.java*

```
/*  
CTHAg00Ontology.ResultatBusqueda
```

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto ResultatBusqueda definido en la ontología,  
que informa sobre el resultado de la acción  
EncontrarPacientesCompatibles con un órgano.  
\*/

```
package Agent0.agentes.CTH.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
public class ResultatBusqueda extends Domain_OntologyObj  
{  
    //Variables  
    private List Assignacio = new ArrayList();  
    private Long NPacients;  
  
    //Constructor  
    public ResultatBusqueda()  
    {  
        super("ResultatBusqueda");  
        NPacients=new Long(0);  
    }  
  
    public ResultatBusqueda(Long n){  
        super("ResultatBusqueda");  
        NPacients=n;  
    }  
  
    //Metodos SET  
    public void setNPacients (Long n)  
    {  
        this.NPacients=n;  
    }  
  
    public void clearAssignacio ()  
    {  
        Assignacio=new ArrayList();  
    }  
  
    public void addAssignacio (Object O)  
    {  
        (this.Assignacio).add(O);  
    }  
  
    //Metodos GET  
    public Iterator getAllAssignacio()  
    {  
        return (this.Assignacio).iterator();  
    }  
  
    public Long getNPacients()  
    {  
        return this.NPacients;  
    }  
}
```

### 8.2.3.37 *TrobarPacientsCompatibles.java*

/\*\*\*\*\*  
CTHAg00Ontology.TrobarPacientsCompatibles

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto TrobarPacientsCompatibles definido en la ontologia  
\*\*\*\*\*/

```
package Agent0.agentes.CTH.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

public class TrobarPacientsCompatibles extends Domain_OntologyObj
{
    //Variables
    private CaracteristiquesBusqueda DatosBusqueda;
    private Hospital hosp;

    //Constructor
    public TrobarPacientsCompatibles()
    {
        super("TrobarPacientsCompatibles");
        DatosBusqueda=new CaracteristiquesBusqueda();
        hosp=new Hospital();
    }
    public TrobarPacientsCompatibles(CaracteristiquesBusqueda cb, Hospital h)
    {
        super("TrobarPacientsCompatibles");
        DatosBusqueda=cb;
        hosp=h;
    }

    //Metodos SET
    public void set_0(CaracteristiquesBusqueda cb)
    {
        this.DatosBusqueda=cb;
    }
    public void set_1(Hospital h)
    {
        this.hosp=h;
    }

    //Metodos GET
    public CaracteristiquesBusqueda get_0()
    {
        return this.DatosBusqueda;
    }
    public Hospital get_1()
    {
        return this.hosp;
    }
}
```



## 8.2.4 Impresión de código contenido en ./Agent0/agentes/A0.

### 8.2.4.1 Ag0.java

```

/*****
Agent0.agentes.A0.

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Agente que es el encargado de interactuar con la base de datos de
este proyecto, con el AgentCTH, con el AgentONT y con el agente
Especialista.(Este ultimo perteneciente al proyecto de David Sanchez
y adaptado por Jaime Bocio).

*****/

package Agent0.agentes.A0;

//Importacion de clases necesarias para implementar un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;
import jade.domain.FIPAAgentManagement.*;

import jade.proto.*;
import jade.onto.sl.sl.*;

//Librerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Importacion de la ontologia.
import Agent0.agentes.A0.Ontology.*;

/*****
Clase Ag0 que implementa el agente encargado de gestionar la BD de
este proyecto.
*****/
public class Ag0 extends Agent{

    //Objeto util para los comportamientos asociados a este agente.
    public static Agent agente0;

    /*****
    Constructor de la Clase Ag0.
    *****/
    public Ag0() {
        super();
        agente0=Ag0.this;
    }

    //Metodos necesarios en este agente.

    /*****
    Metodo que se ejecuta la primera vez, cuando se crea el agente.
    *****/
    protected void setup(){

        //Intentamos cargar el driver JDBC-ODBC.
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        }catch (ClassNotFoundException e){
            System.err.println(getLocalName()+"No s'ha pogut trobar "+
            "la classe de driver JDBC/ODBC.");
            //S'esborra l'agent.
            System.out.println(getLocalName()+ ": Esborrant Agent.");

```

```

    doDelete();
}

//Registrarse en el DF.
RegistreEnEldf();

//Crear y añadir comportamientos para realizar acciones solicitadas
//por AgentCTH.
MessageTemplate mt =
MessageTemplate.and(MessageTemplate.MatchProtocol("FIPA-Request"),
MessageTemplate.MatchOntology(OntologiaAg0.NAME));

//Creamos un Objeto contestador determinado para cada
//accion que el paciente puede solicitar a Agent0.
FipaRequestResponderBehaviour_sl contestador= new
FipaRequestResponderBehaviour_sl(this,mt);

FipaRequestResponderBehaviour_sl contestador2 = contestador;

FipaRequestResponderBehaviour_sl contestador3 = contestador;

FipaRequestResponderBehaviour_sl contestador4 = contestador;

FipaRequestResponderBehaviour_sl contestador5 = contestador;

FipaRequestResponderBehaviour_sl contestador6 = contestador;

//COMPORTAMIENTO PARA ANOTAR MUERTE PACIENTE.
//Lo asociamos con la ontologia
contestador.registerFactory(OntologiaAg0.ANOTARMORTPACIENT,
new RespuestaAnotacionMuertePaciente (this, new
ACLMessage(ACLMessage.NOT_UNDERSTOOD)));
//Añadimos el comportamiento.
addBehaviour(contestador);

//COMPORTAMIENTO PARA INTRODUCCION DE ORGANO.
//Lo asociamos con la ontologia
contestador2.registerFactory(OntologiaAg0.INTRODUIRPACIENT,
new RespuestaIntroduccionPaciente (this,
new ACLMessage(ACLMessage.NOT_UNDERSTOOD)));
//Añadimos el comportamiento.
addBehaviour(contestador2);

//COMPORTAMIENTO PARA ASIGNACION DE ORGANO.
//Lo asociamos con la ontologia.
contestador3.registerFactory(OntologiaAg0.ASIGNACIOORGAN,
new RespuestaAsignacionOrgano (this,
new ACLMessage(ACLMessage.NOT_UNDERSTOOD)));
//Añadimos el comportamiento.
addBehaviour(contestador3);

//COMPORTAMIENTO PARA OPERACION REALIZADA.
//Lo asociamos con la ontologia
contestador4.registerFactory(OntologiaAg0.OPERACIOREALITZADA,
new RespuestaOperacionRealizada (this, new
ACLMessage(ACLMessage.NOT_UNDERSTOOD)));
//Añadimos el comportamiento.
addBehaviour(contestador4);

//COMPORTAMIENTO PARA ORGANO RECHAZADO.
//Lo asociamos con la ontologia
contestador5.registerFactory(OntologiaAg0.ORGANREBUTXAT,
new RespuestaOrganoRechazado (this, new
ACLMessage(ACLMessage.NOT_UNDERSTOOD)));
//Añadimos el comportamiento.
addBehaviour(contestador5);

//COMPORTAMIENTO PARA ENCONTRAR PACIENTES COMPATIBLES.
//Lo asociamos con la ontologia
contestador6.registerFactory(OntologiaAg0.TROBARPACIENTSCOMPATIBLES,

```

```

        new RespuestaEncontrarPacientesCompatibles (this, new
        ACLMessage(ACLMessage.NOT_UNDERSTOOD)));
        //Añadimos el comportamiento.
        addBehaviour(contestador6);
    }

/*****
Metodo que se ejecuta cuando el agente deja de funcionar.
*****/
protected void takeDown(){
    DesregistreEnElDf();
}

/*****
Metodo que sirve para registrar agente en el DF.
*****/
public void RegistreEnElDf(){

    ServiceDescription sd1 = new ServiceDescription();

    //Indicamos los parametros del agente.
    sd1.setType("Agent0");
    sd1.setName(getLocalName());
    sd1.setOwnership("PFCAgent0");

    //Obtenemos una instancia del DF.
    DFAgentDescription dfd=new DFAgentDescription();

    //Nos añadimos como un nuevo servicio.
    dfd.addServices(sd1);

    //Indicamos nuestro nombre.
    AID aid=new AID();
    aid.setName(getLocalName());
    dfd.setName(aid);

    //Registramos el lenguaje.
    registerLanguage(SLCodec.NAME, new SLCodec());

    //Registramos la ontologia.
    registerOntology(OntologiaAg0.NAME,
    OntologiaAg0.instanceAg0());

    System.out.println("Hospital (" + getLocalName() +
    " de " + sd1.getOwnership() + ") on-line");

    try{
        //Registramos agente en el DF.
        DFServiceCommunicator.register(this,dfd);
    }catch (FIPAException e) {
        System.err.println(getLocalName()+
        ": Error en el registro al DF. Error: "+e.getMessage());
        doDelete();
    }
}

/*****
Metodo que desregistra un agente del DF.
*****/
public void DesregistreEnElDf(){

    //Obtenemos una instancia del DF
    DFAgentDescription dfd=new DFAgentDescription();
    AID aid=new AID();
    aid.setName(getLocalName());
    dfd.setName(aid);
    try{
        //Nos desregistramos.
        DFServiceCommunicator.deregister(this,dfd);
    }
}

```

```
        System.out.println("Agente " + getLocalName() + " eliminado del DF");
    } catch (FIPAException e){
        System.out.println(getLocalName()+
            ": Error en el desregistro del DF. Error: "+e.getMessage());
    }
}
```

#### 8.2.4.2 ESPECIALISTABehaviour.java

```

/*****
PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Comportamiento que Agent0 instancia para iniciar un request con Agente
Especialista y comunicarse con AgentCTH.

*****/

package Agent0.agentes.A0;

//Jade Core
import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.FIPAAException;

//Comportamiento
import jade.proto.*;

import jade.domain.*;
import jade.domain.FIPAAgentManagement.*;

//Language de comunicacion SL
import jade.lang.sl.sl.*;
import jade.lang.acl.*;

//Ontologias
import jade.onto.*;
import jade.onto.sl.sl.*;

import java.io.*;
import java.util.*;

//Ontologia definida en el paquete correspondiente a Ag0
import Agent0.agentes.A0.Ontology.*;

/*****
Clase que implementa un comportamiento Request: Peticion de Ag0 para
que Especialista ordene una lista de pacientes, segun la compatibilidad
de estos con un determinado organo y responde a AgentCTH.
*****/
public class ESPECIALISTABehaviour extends FipaRequestInitiatorBehaviour
{
    //Para relacionar orden de pacientes devuelto por agente especialista
    //con su CodiSS.
    public Vector ordre_pacients;

    public ACLMessage MensajeOrigen;
    private Agent a;
    public AID AID_agent0;

    UtilidadesAgent0 u = new UtilidadesAgent0();

/*****
Constructor del comportamiento que sigue protocolo Request.
*****/
    public ESPECIALISTABehaviour (Agent agent, ACLMessage requestMsg)
    {
        super(agent,requestMsg);
        a=agent;
        System.out.println(a.getLocalName() +
            ":Iniciando Request-Protocol para Encontrar pacientes compatibles con"+
            " un determinado organo.");
    }
}

```

```

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Agree.
*****/
protected void handleAgree(ACLMessage msg)
{
    System.out.println("\n-----
-");
    System.out.println("Requester Agent: Ag0 aceptado, esperando
resultado...");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());

    try{
        //Se crea mensaje que se pasara a agente CTH.
        ACLMessage reply = MensajeOrigen.createReply();

        List l = myAgent.extractContent(MensajeOrigen);

        SL_Action ac = (SL_Action) l.get(0);

        l.clear();
        l.add(ac);
        myAgent.fillContent(reply, l);
        reply.setPerformative(ACLMessage.AGREE);
        myAgent.send(reply);

    }catch (FIPAException fe){
        System.err.println("Error leyendo el mensaje");
    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Inform.
*****/
protected void handleInform(ACLMessage msg)
{
    System.out.println("\n-----
-");
    System.out.println("Requester Agent: Ag0 completado");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());
    System.out.println("\nResultado:");
    try{
        //Se crea mensaje que se pasara a agente CTH.
        //Creamos respuesta.
        ACLMessage reply = MensajeOrigen.createReply();

        //Obtenemos el contenido del mensaje.
        List l = myAgent.extractContent(msg);
        //Es de tipo RESULT
        SL_Result res = (SL_Result) l.get(0);

        //Crearemos Mensaje de respuesta para AgentCTH (o cualquier agente
        //que solicite este servicio).
        //Se pasan datos del objeto Resultat recibido de Agente Especialista,
        //a objeto ResultatBusqueda que se enviara a agente CTH.
        Iterator it = res.getAll_1();
        Resultat r = (Resultat) it.next();

        Long n = r.getnobjectes();
        Iterator llista_graus = r.getAllassignacio();
        Grau g;

        //Objeto resultado que se pasara a AgentCTH.
        ResultatBusqueda rb = new ResultatBusqueda();

        rb.setNPacients(n);

```

```

while (llista_graus.hasNext())
{
    g=(Grau) llista_graus.next();
    Long id = g.getIdObjecte();
    String va = g.getvalorGrau();

    String CodSS = (String) ordre_pacients.elementAt(id.intValue()-1);

    Associacio ass = new Associacio();
    ass.setCodiSS("/_"+u.StringSinEspaciosEnBlanco(CodSS));
    ass.setGrau(va);
    rb.addAssignacio(ass);
}

//Creamos objeto Result para devolver a CTH.
//Obtenim el contingut del missatge
List lm = myAgent.extractContent(MensajeOrigen);
//Es de tipus SL_Action
SL_Action ac = (SL_Action) lm.get(0);

//Enviamos objeto resultado de busqueda de pacientes compatibles
//a AgentCTH.
SL_Result res_CTH = new SL_Result();
res_CTH.set_0(ac);
res_CTH.add_1(rb);

l.clear();
l.add(res_CTH);
myAgent.fillContent(reply, l);
reply.setPerformative(ACLMessage.INFORM);
myAgent.send(reply);

}catch (FIPAException fe){
    System.err.println("Error leyendo el mensaje");
}catch (Exception e){
    System.err.println("Excepcion general"+e.toString());
}
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje
NotUnderstood.
*****/
protected void handleNotUnderstood(ACLMessage msg)
{
    System.out.println("\n-----
-");
    System.out.println("Requester Agent: Ag0 no entendido");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());

    try{
        //Se crea mensaje que se pasara a agente CTH.
        //Creamos respuesta.
        ACLMessage reply = MensajeOrigen.createReply();

        //Obtenim el contingut del missatge
        List l = myAgent.extractContent(MensajeOrigen);
        //Es de tipus SL_Action
        SL_Action ac = (SL_Action) l.get(0);

        l.clear();
        l.add(ac);
        myAgent.fillContent(reply, l);
        reply.setPerformative(ACLMessage.NOT_UNDERSTOOD);
        myAgent.send(reply);
    }catch(Exception e){
        System.err.println("Error desconocido leyendo el mensaje del fallo.");
    }
}

```

```

    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Failure.
*****/
protected void handleFailure(ACLMessage msg)
{
    System.out.println("\n-----");
    System.out.println("Requester Agent: Ag0 fallado");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());

    //Recuperamos la razon del fallo
    try{
        //Se crea mensaje que se pasara a agente CTH.
        //Creamos respuesta.
        ACLMessage reply = MensajeOrigen.createReply();

        //Extraemos el contenido del mensaje
        List l=myAgent.extractContent(msg);

        List lm = myAgent.extractContent(MensajeOrigen);
        //Recuperem l'objecte enviat
        SL_Action a = (SL_Action) lm.get(0);

        //Obtenemos el objeto enviado
        Object rao = l.get(1);

        //Buscamos en la ontologia la clase que se corresponde con objeto.
        Ontology o = myAgent.lookupOntology(msg.getOntology());
        String nomRao = o.getRoleName(rao.getClass());
        //La indicamos
        System.out.println("La razon del fallo es: " + nomRao);

        l.clear();
        l.add(a);
        l.add(rao);
        myAgent.fillContent(reply, l);
        reply.setPerformative(ACLMessage.FAILURE);
        myAgent.send(reply);
    }catch(FIPAException fe){
        System.err.println("FIPAException leyendo el fallo: " + fe);
    }catch(OntologyException oe){
        System.err.println("OntologyException leyendo el fallo: " + oe);
    }catch (Exception e){
        System.err.println("Error desconocido leyendo el mensaje del fallo");
    }
}

/*****
Metodo que se ejecuta cuando comportamiento recibe un mensaje Refuse.
*****/
protected void handleRefuse(ACLMessage msg)
{
    System.out.println("\n-----");
    System.out.println("Requester Agent: Ag0 rechazado");
    System.out.println("Requester Agent: mensaje recibido: ");
    System.out.println(msg.toString());

    //Recuperamos la razon del rechazo
    try{
        //Se crea mensaje que se pasara a agente CTH.
        //Creamos respuesta.
        ACLMessage reply = MensajeOrigen.createReply();

```



```

//Obtenemos el contenido del mensaje
List l = myAgent.extractContent(msg);

List lm = myAgent.extractContent(MensajeOrigen);
//Recuperem l'objecte enviat
SL_Action a = (SL_Action) lm.get(0);

//Recuperamos el objeto recibido
Object rao = l.get(1);
//Buscamos en la ontologia la clase que se corresponde con el objeto
//recibido.
Ontology o = myAgent.lookupOntology(msg.getOntology());
String nomRao = o.getRoleName(rao.getClass());
//Escribimos la razon del rechazo.
System.out.println("La razon del rechazo es: " + nomRao);

l.clear();
l.add(a);
l.add(rao);
myAgent.fillContent(reply, l);
reply.setPerformative(ACLMessage.REFUSE);
myAgent.send(reply);

} catch (FIPAException fe){
    System.err.println("FIPAException leyendo el mensaje de rechazo: " +
fe);
} catch (OntologyException oe){
    System.err.println("OntologyException leyendo el mensaje de rechazo: " +
oe);
} catch (Exception e){
    System.err.println("Error desconocido leyendo el mensaje de rechazo.");
}
}
}

```

### 8.2.4.3 RespuestaAsignacionOrgano.java

/\*\*\*\*\*\*

PFC Agent0.  
Cristóbal Ledesma Oljas  
8/10/02

Clase que implementa el comportamiento que responde en el protocolo request por parte de Agent0, para que AgentCTH tenga constancia, de como ha ido la asignacion de un organo a un paciente. Ademas de realizar dicha accion.

\*\*\*\*\*/

```
package Agent0.agentes.A0;
```

```
//Importacion de clases necesarias para implementar un agente.
```

```
import jade.core.*;  
import jade.core.behaviours.*;  
import jade.domain.*;  
import jade.domain.FIPAAgentManagement.*;
```

```
import jade.proto.*;  
import jade.onto.sl.sl.*;  
import java.sql.*;  
import java.util.*;
```

```
//Librerias necesarias para construir un mensaje.
```

```
import jade.lang.acl.*;  
import jade.lang.sl.sl.*;
```

```
//Importacion de la ontologia.
```

```
import Agent0.agentes.A0.Ontology.*;
```

/\*\*\*\*\*\*

Clase que implementa un comportamiento Request:  
Petición de AgentCTH para que Agent0 anote en su base de datos la  
asignacion de un organo a un paciente.

\*\*\*\*\*/

```
public class RespuestaAsignacionOrgano extends  
FipaRequestResponderBehaviour_sl.ActionHandler  
implements FipaRequestResponderBehaviour_sl.Factory  
{
```

```
    //Datos para conectarse con la base de datos.
```

```
    static String login ="db1";  
    static String passwd ="A93652ZY";  
    static String url="jdbc:odbc:db1";
```

```
    private final static int FALLOSQL=-1;  
    private final static int CORRECTE=0;  
    private final static int PACIENTMORT=1;  
    private final static int PACIENTINEXISTENT=2;  
    private final static int ORGANINEXISTENT=3;  
    private final static int PACIENTAMBORGANASSIGNAT=4;  
    private final static int PACIENTOPERAT=5;  
    private final static int PACIENTSENSEESPERARORGAN=6;  
    private final static int ORGANREBUTXAT=7;  
    private final static int ORGANASSIGNAT=8;  
    private final static int HOSPITALNOAUTORITZAT=9;
```

```
    AID AID_agent0 = new AID();  
    Agent a=null;  
    UtilidadesAgent0 u =new UtilidadesAgent0();
```

/\*\*\*\*\*\*

```
Constructor del comportamiento que se encarga de anotar la asignacion
```

```

de un organo a un paciente en la BD.
*****/
public RespuestaAsignacionOrgano (Agent agent, ACLMessage respuesta)
{
    super(agent, respuesta);
    a=agent;
    AID_agent0=a.getAID();

    System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para RESPONDER A anotar asignacion"+
        " muerte de paciente con "
        + AID_agent0);
}

/*****
Implementacion de la interficie para manejar las acciones recibidas.
*****/
public FipaRequestResponderBehaviour_sl.ActionHandler create(ACLMessage msg)
{
    System.out.println(AID_agent0+": Mensaje inicial :"+ msg.toString());
    return new RespuestaAsignacionOrgano (Ag0.agente0,msg);
}

/*****
Manejador de mensajes Request: Accion a realizar por este
comportamiento.
*****/
public void action ()
{
    try{
        //Obtenemos la peticion del mensaje recibido.
        ACLMessage msg=getRequest();

        System.out.println(a.getLocalName() +
            ": Mensaje de REQUEST recibido: \n"
            + msg.toString() + "\n");

        //Evaluacion del mensaje y composicion de la posibles respuestas.
        //Accion a realizar por Agent0.
        //A partir de aqui, comenzamos a realizar la accion pedida.
        System.out.println("Procesant IntroducirPacient...");

        //Extraemos contenido de mensaje inicial de Request.
        List l_ag = a.extractContent(msg);
        SL_Action SLa = (SL_Action) l_ag.get(0);

        //Sabemos que es de tipo AssignacioOrgan el objeto recibido.
        AssignacioOrgan ao = (AssignacioOrgan) SLa.get_1();

        //Lo hacemos aqui para escribirlo una sola vez.
        //Creamos un ACLMessage de respuesta.
        ACLMessage reply = msg.createReply();

        //Ag0 comprueba si es posible realizar accion pedida por AgentCTH.
        int decision = EsPosibleExtraerYTratarDatos(ao);
        switch (decision)
        {
            case (ORGANREBUTXAT):
                //ORGANREBUTXAT no quiere decir que no se pueda asignar el nuevo
                //organo al paciente, sino que tendra un tratamiento diferente
                //a que el paciente este esperando organo y nunca se le haya
                //asignado ninguno. POR ESTE MOTIVO SE EJECUTARA CORRECTE.

            case CORRECTE:
                //Enviamos el mensaje: AGREE.
                //Obtenemos la accion.
                l_ag.clear();
                l_ag.add(SLa);
                l_ag.add(new SL_True());
        }
    }
}

```

```

myAgent.fillContent(reply, l_ag);

sendReply(ACLMessage.AGREE, reply.getContent());

if (ExtraerYTratarDatos(ao) == 0)
{
    //Generamos un mensaje con accion realizada correctamente
    //Lo llenamos con contenido de mensaje original

    SL_Done d = new SL_Done();
    d.set_0(SLa);
    l_ag.clear();
    l_ag.add(d);

    reply.setPerformative(ACLMessage.INFORM);
    myAgent.fillContent(reply, l_ag);

    myAgent.send(reply);
}
else
{
    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.FAILURE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new ErrorAccesBD());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
}
break;
case PACIENTINEXISTENT:
    //Generamos un mensaje resultado indicando que paciente,
    //con identificador introducido, no existe en BD.
    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientInexistent());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
break;
case PACIENTMORT:
    //Generamos un mensaje resultado indicando que el paciente,
    //con identificador introducido, esta registrado como muerto.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientMort());

```

```

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;
case ORGANINEXISTENT:
//Generamos un mensaje resultado indicando que organo,
//con identificador introducido, no existe en BD.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new OrganInexistent());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case PACIENTAMBORGANASSIGNAT:
//Generamos un mensaje resultado indicando que el paciente,
//con identificador introducido, ya tiene organo asignado en BD.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new PacientTeOrganAssignat());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case PACIENTOPERAT:
//Generamos un mensaje resultado indicando que paciente,
//con identificador introducido, ya se le ha trasplantado
//organo de ese tipo.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new PacientOperat());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case PACIENTSENSEESPERARORGAN:
//Generamos un mensaje refuse indicando que paciente,
//con identificador introducido, no espera ese tipo de organo.

```

```

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new PacientSenseEsperarOrgan());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case ORGANASSIGNAT:
//Generamos un mensaje refuse indicando que organo,
//con identificador introducido, ya esta asignado a otro paciente.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new OrganNoDisponible());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case HOSPITALNOAUTORITZAT:
//Generamos un mensaje refuse indicando que el hospital
//no esta autorizado a solicitar accion.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new HospitalSenseAutoritzar());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);

break;

case FALLOSQL:
//Generamos un mensaje REFUSE indicando que se ha producido
//un error al acceder a BD.

reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new ErrorAccesBD());

```

```

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply, l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

        break;
    }
} catch (FIPAException fe){
    System.out.println(a.getName() +
        ": Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}
}

/*****
Manejador de mensajes Request: Metodo done que indica si comportamiento
ha de seguir activo o no.
*****/
public boolean done()
{
    System.out.println(a.getLocalName() +
        ":Comportamiento finalizado correctamente");

    //Ninguna accion a realizar al finalizar el comportamiento.
    return true;
}

/*****
Manejador de mensajes Request: Metodo que reinicia comportamiento.
*****/
//Reinicializacion del comportamiento.
public void reset()
{
    System.out.println(a.getLocalName() +
        ": Reinicializado el comportamiento");
}

/*****
Metodo que consulta BD y dice si es posible realizar accion pedida
por usuario.

Retorna:
-1 -> Si se produce cualquier error de tipo SQLException.
0 -> Si se puede actualizar datos correctamente.
1 -> Si paciente esta registrado en BD y esta muerto.
2 -> Si paciente no esta registrado en BD.
3 -> Si organo no esta registrado en BD.
4 -> Si paciente ya tiene un organo del mismo tipo asignado.
5 -> Si paciente ya se le ha trasplantado un organo del mismo tipo.
6 -> Si paciente no espera recibir un organo de este tipo.
7 -> Si paciente espera recibir organo, pero ha rechazado otro.
8 -> Si organo esta asignado a otro paciente que esta vivo.
*****/
private int EsPosibleExtraerYTratarDatos (AssignacioOrgan ao)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;
    try{
        int i=CORRECTE; //i contiene el valor a retornar por esta funcion.
        //Intentamos Conectar con la base de datos.
        con = DriverManager.getConnection (url, login, passwd);
        stmt=con.createStatement();

        //Se consulta estado del paciente en BD.
        //Indica si paciente esta en base de datos, y esta vivo.
        i=PacientViu(ao,stmt);
    }
}

```

```

//Se consulta si el organo esta registrado en BD.
if (i==CORRECTE) i=ExisteixOrgan(ao,stmt);

//Hace falta hacer comprobacion de que el organo no esta siendo
//utilizado por ninguna otra persona viva.
if (i==CORRECTE) i=OrganNoUtilitzat(ao,stmt);

//Se comprueba si organo esta asignado a paciente que esta
//muerto, y que no esta en estado operado o rechazado.
if (i==CORRECTE) i=PacientReceptorMortAmbOrganAssignat(ao,stmt);

//Se consulta si el paciente esta operado o tiene asignado un
//organo del mismo tipo, o simplemente no espera recibir este
//tipo de organo.
//O que el estado del paciente no sea 'En Espera De àrgano', pero
//haya rechazado otro del mismo tipo, y por tanto espere recibirlo.
if (i==CORRECTE) i=TeOrganAssignatOOperat(ao,stmt);

//Se consulta si hospital esta autorizado para solicitar accion.
if (i==CORRECTE) i = HospitalAutorizado(ao,stmt);

return i;

}catch (SQLException e){
    System.out.println(a.getLocalName()+
        ": codi d'error "+e.getErrorCode()+
        ".SQLException -> "+e.getMessage());

    return -1;
}finally{ //Este codigo siempre se ejecutara.
    try{ //Cerramos Statement y conexion con BD.
        if (stmt != null) stmt.close();
        if (con != null) con.close();
    }catch (SQLException e){
        System.out.println(a.getLocalName()+
            ": codi d'error "+e.getErrorCode()+
            ".SQLException -> "+e.getMessage().toString());
        return -1;
    }
}
}

/*****
Metodo que busca en BD si hay registrado un paciente con CodiSS igual
al que se pasa por parametro en el objeto AssignacioOrgan y consta como
un paciente vivo.

Retorna: 0 -> si existe paciente en BD como Vivo.
        1 -> si existe paciente en BD como Muerto.
        2 -> si no esta registrado paciente en BD.
*****/
private int PacientViu (AssignacioOrgan p,Statement stmt) throws
SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT P.CodiSS, P.Estat FROM Pacients P"+
            " WHERE P.CodiSS = '"+u.StringConEspaciosEnBlanco(p.get_0())+"'");

        if (result.next())
        {
            if (result.getString(2).equals("Vivo"))
            {
                System.out.println(a.getLocalName()+
                    "Existe paciente vivo en BD.");
                return CORRECTE;
            }
        }
        else
        {
            System.out.println(a.getLocalName()+

```



```

        "Existe paciente muerto en BD.");
        return PACIENTMORT;
    }
}
//No existe paciente en BD con este CodiSS.
System.out.println(a.getLocalName()+"No existe paciente en BD.");
return PACIENTINEXISTENT;
}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
}
}

/*****
Metodo que busca en BD si hay registrado un organo con IdOrgan igual
al que se pasa por parametro en el objeto AssignacioOrgan.Y si esta
registrado que no sea una descripcion de un organo que espera recibir
otro paciente.

Retorna: 0 -> si existe organo en BD.
        3 -> si no existe organo en BD.
*****/
private int ExisteixOrgan (AssignacioOrgan p,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        //Se comprueba que existe organo en BD y que este realmente
        //se ha donado y no es una descripcion de un organo, que
        //se espera recibir.

        result = stmt.executeQuery ("SELECT O.IdOrgan FROM OrgansDonats O"+
        " WHERE O.IdOrgan = '"+u.StringConEspaciosEnBlanco(p.get_1())+"'");

        if (result.next())
        {
            System.out.println(a.getLocalName()+"Existe organo en BD.");
            return CORRECTE;
        }
        //No existe paciente en BD con este IdOrgan.
        System.out.println(a.getLocalName()+"No existe organo en BD.");
        return ORGANINEXISTENT;

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que busca en BD si el paciente necesita que se le asigne
un organo del mismo tipo al que se pasa por parametro
en el objeto AssignacioOrgan.

Retorna: 0 -> Paciente no tiene organo operado o asignado del mismo tipo.
        4 -> Paciente tiene organo asignado del mismo tipo.
        5 -> Paciente esta operado de organo del mismo tipo.
        6 -> Paciente no espera este tipo de organ.
        7 -> Paciente ha rechazado un organo del mismo tipo y no
            se le ha vuelto a asignar otro.

Retorna 7, porque es necesario diferenciar entre estado de
la interrelacion Necesita 'En Espera De Organ' y 'Rechazado,
ya que el tratamiento en la base de datos no sera el mismo,
a la hora de hacer la asignacion.
*****/
private int TeOrganAssignatOOperat (AssignacioOrgan p,Statement stmt) throws
SQLException
{
    ResultSet result = null;
    ResultSet result2 = null;
    int rechazo=0;

```

```

try{
    //Obtenemos todos los estados de la tabla Necesita
    //de los organos que son del mismo tipo que el que
    //se referencia por el objeto AssignacioOrgan
    //que estan relacionados con el paciente en cuestion.

    result = stmt.executeQuery ("SELECT A.Estat FROM Assignacio A"+
    " WHERE A.CodiSS = '"+u.StringConEspaciosEnBlanco(p.get_0())+"'"+
    " and A.IdOrgan IN (SELECT O.IdOrgan FROM OrgansDonats O"+
    " WHERE O.Tipus IN (SELECT O2.Tipus FROM OrgansDonats O2"+
    " WHERE O2.IdOrgan = '"
    +u.StringConEspaciosEnBlanco(p.get_1())+"'))");

    //Un paciente puede estar relacionado con mas de un organo
    //del mismo tipo. p.e.:Cuando ha habido un rechazo y se ha de
    //reasignar otro organo al paciente.
    while (result.next())
    {
        //Creamos variable Estat para no tener que acceder varias
        //veces a BD.
        String Estat = result.getString(1);

        if (Estat.equals("Organo Asignado"))
        {
            System.out.println(a.getLocalName()+
            "El paciente ya tiene un organo asignado.");
            return PACIENTAMBORGANASSIGNAT;
        }

        if (Estat.equals("Operado"))
        {
            System.out.println(a.getLocalName()+
            "Al paciente ya se le ha trasplantado organo de mismo tipo.");
            return PACIENTOPERAT;
        }

        if (Estat.equals("Rechazado"))
        {
            //Aquí se utiliza un booleano para poder seguir con las
            //comprobaciones dentro del while. Ya que un paciente
            //puede haber rechazado un organo, pero tener asignado otro.
            rechazo=1;
        }
    }

    if (rechazo==1)
    {
        System.out.println(a.getLocalName()+
        "Al paciente ya se le ha trasplantado organo de mismo tipo,"+
        " pero lo ha rechazado.");
        return ORGANREBUTXAT;
    }

    //Para comprobar si paciente ha solicitado ese tipo de organo.
    result2 = stmt.executeQuery ("SELECT O.CodiSSEesperat "+
    "FROM OrgansEsperats O"+
    " WHERE O.CodiSSEesperat = '"+u.StringConEspaciosEnBlanco(p.get_0())+"'"+
    " and O.Tipus IN (SELECT O2.Tipus FROM OrgansDonats O2"+
    " WHERE O2.IdOrgan = '"
    +u.StringConEspaciosEnBlanco(p.get_1())+"')");

    if (result2.next()==false){
        //Paciente no espera recibir este tipo de organo.
        System.out.println(a.getLocalName()+"Paciente no espera recibir este"+
        " tipo de organo.");
        return PACIENTSENSEESPERARORGAN;
    }

    //En realidad esperamos que nunca se ejecute este ultimo return,
    //pero el compilador da error en caso de que no haya un return en una
    //funcion.
    return CORRECTE;
}

```

```

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
        if (result2!=null) result2.close();
    }
}

/*****
Metodo que busca en BD que el organo, pasado en el objeto AssignacioOrgan,
no este asignado a otra persona que esta muerta, y pueda ser reaprovechado.

Retorna: 0 -> Organo disponible teniendo en cuenta solo este motivo.
        8 -> Organo no disponible.
*****/
private int PacientReceptorMortAmbOrganAssignat
(AssignacioOrgan ao,Statement stmt) throws SQLException
{
    ResultSet result = null;
    try{
        //Un organo que se habia asignado previamente a un paciente,
        //puede ser utilizado por otro al fallecer el primero.

        result = stmt.executeQuery ("SELECT A.Estat FROM Assignacio A"+
            " WHERE A.CodiSS IN (SELECT P.CodiSS FROM Pacients P"+
            " WHERE P.Estat = 'Muerto')"+
            " and A.Estat NOT IN ('Organos Asignados')"+
            " and A.IdOrgan = '"+u.StringConEspaciosEnBlanco(ao.get_1())+"'");

        if (result.next())
        {
            //Organos no puede ser utilizado por otro paciente porque
            //fue utilizado por paciente que fallecio.
            System.out.println(a.getLocalName()+"Organos no puede ser aprovechado"+
                "por otro paciente.");
            return ORGANASIGNAT;
        }
        else
        {
            //Organos puede ser aprovechado por otro paciente.
            System.out.println(a.getLocalName()+"Organos puede ser aprovechado"+
                "por otro paciente.");
            return CORRECTE;
        }
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result != null) result.close();
    }
}

/*****
Metodo que busca en BD, si el organo pasado en el objeto AssignacioOrgan
pertenece a otra persona que esta viva.

Retorna: 0 -> Organos disponible. (Teniendo en cuenta solo esto).
        8 -> Organos no disponible.
*****/
private int OrganNoUtilitzat (AssignacioOrgan ao,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        //El Organos en cuestion puede estar siendo utilizado
        //por otro paciente que esta vivo.
        result = stmt.executeQuery ("SELECT P.CodiSS FROM Pacients P"+
            " WHERE P.Estat = 'Vivo'+
            " and P.CodiSS IN (SELECT A.CodiSS FROM Assignacio A"+
            " WHERE A.IdOrgan = '"+u.StringConEspaciosEnBlanco(ao.get_1())+"'");

        if (result.next())
        {
            System.out.println(a.getLocalName()+"Organos esta siendo aprovechado"+

```

```

        "por otro paciente.");

        return ORGANASSIGNAT;
    }
    else
    {
        //Organo no esta siendo utilizado por otro paciente que este vivo.
        System.out.println(a.getLocalName()+"Organo esta disponible");
        return CORRECTE;
    }
}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
}
}

/*****
Metodo que actualiza BD con accion pedida por el usuario.
Caso particular en que paciente haya rechazado un organo con
anterioridad, y se le vaya a asignar otro.

Retorna:
    -1 -> Si se produce cualquier error de tipo SQLException.
    0 -> Si todo ha ido bien.
*****/
private int ExtraerYTratarDatos (AssignacioOrgan ao)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;
    try{
        //Intentamos Conectar con la base de datos.
        con = DriverManager.getConnection (url, login, passwd);
        stmt=con.createStatement();

        //Insertamos fila en tupla Necesita con nueva relacion.

        int i=stmt.executeUpdate ("INSERT INTO Assignacio" +
            " VALUES ('"+u.StringConEspaciosEnBlanco(ao.get_1())+"', '"+
            u.StringConEspaciosEnBlanco(ao.get_0())+"', '"+
            "Organo Asignado'");

        System.out.println(a.getLocalName()+
            "Se ha actualizado relacion Assignacio. i->" +i);

        return 0;

    }catch (SQLException e){
        System.out.println(a.getLocalName()+
            ": codi d'error "+e.getErrorCode()+
            ".SQLException -> "+e.getMessage());
        return -1;
    }finally{ //Este codigo siempre se ejecutara.
        try{ //Cerramos Statement y conexion con BD.
            if (stmt != null) stmt.close();
            if (con != null) con.close();
        }catch (SQLException e){
            System.out.println(a.getLocalName()+": codi d'error "+
                e.getErrorCode()+
                ".SQLException -> "+e.getMessage().toString());
            return -1;
        }
    }
}

/*****
Metodo que busca en BD si hay registrado un hospital con NomHospital y
Ciutat igual al que se pasa por parametro en el objeto Hospital, y si
el paciente en cuestion esta ingresado alli.

```

```

Retorna: 0 -> si existe hospital => Hospital autorizado.
        9 -> no existe hospital => Hospital no autorizado..
*****/
private int HospitalAutorizado (AssignacioOrgan ao,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{

        Hospital h = ao.get_2();
        Adreza adr = h.getAdreza();
        String ciudad = u.StringConEspaciosEnBlanco(adr.getCiutat());

        result = stmt.executeQuery ("SELECT H.NomHospital FROM HospitalIngres
H"+
        " WHERE H.NomHospital = '"+u.StringConEspaciosEnBlanco(h.getNom())+"'"+
        " and H.Ciutat = '"+ciudad+"'"+
        " and H.CodiSS = '"+u.StringConEspaciosEnBlanco(ao.get_0())+"'");

        if (result.next())
        {
            //Existe paciente ingresado en hospital; por lo tanto, esta autorizado
            //a solicitar accion.
            System.out.println(a.getLocalName()+"Hospital autorizado.");
            return CORRECTE;
        }
        //Paciente no ingresado en ese hospital y no esta autorizado
        //a solicitar accion.
        System.out.println(a.getLocalName()+"Hospital no autorizado.");
        return HOSPITALNOAUTORITZAT;

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}
}

```

#### 8.2.4.4 RespuestaEncontrarPacientesCompatibles.java

```

/*****

PFC Agent0.
Cristóbal Ledesma Olías
8/10/02

Clase que implementa el comportamiento que responde en el protocolo
request por parte de Agent0, para informar a AgentCTH de posibles
pacientes compatibles con un organo y se inicia protocolo request con
agente Especialista.

*****/

package Agent0.agentes.A0;

//Importacion de clases necesarias para implementar un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;
import jade.domain.FIPAAgentManagement.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.sql.*;
import java.util.*;

//Librerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Importacion de la ontologia.
import Agent0.agentes.A0.Ontology.*;

/*****
Clase que implementa un comportamiento Request:
Peticion de AgentCTH para que Agent0 inicie la busqueda de pacientes
compatibles con un determinado organo.
*****/
public class RespuestaEncontrarPacientesCompatibles extends
FipaRequestResponderBehaviour_sl.ActionHandler
implements FipaRequestResponderBehaviour_sl.Factory
{
    UtilidadesAgent0 u = new UtilidadesAgent0();
    AID AID_agent0 = new AID();
    Agent a=null;

    //Relaciona id asignado a los pacientes por agente Especialista con su
    //CodiSS.
    Vector ordre_pacients = new Vector();

    //Datos para conectarse con la base de datos.
    static String login ="db1";
    static String passwd ="A93652ZY";
    static String url="jdbc:odbc:db1";

    private final static int FALLOSQL=-1;
    private final static int CORRECTE=0;
    private final static int PACIENTINEXISTENT=1;
    private final static int ORGANINEXISTENT=2;
    private final static int HOSPITALNOAUTORITZAT=3;

/*****
Constructor del comportamiento que se encarga de buscar pacientes
compatibles con un determinado organo.
*****/

```

```

    public RespuestaEncontrarPacientesCompatibles (Agent agent, ACLMessage
respuesta)
    {
        super(agent, respuesta);
        a=agent;
        AID_agent0=a.getAID();

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para RESPONDER A Encontrar Pacientes
Compatibles"+
        " con "+ a.getAID());
    }

/*****
Implementacion de la interficie para manejar las acciones recibidas.
*****/
    public FipaRequestResponderBehaviour_sl.ActionHandler create
(ACLMessage msg){

        //Instanciacion a este comportamiento.
        return new RespuestaEncontrarPacientesCompatibles (Ag0.agente0,msg);
    }

/*****
Manejador de mensajes Request: Accion a realizar por este
comportamiento.
*****/
    public void action ()
    {
        try{

            //Obtenemos la peticion del mensaje recibido.
            ACLMessage msg=getRequest();

            System.out.println(a.getLocalName() +
            ": Mensaje de REQUEST recibido: \n"
            + msg.toString() + "\n");

            //Evaluacion del mensaje y composicion de las posibles respuestas.
            //Accion a realizar por Agent0.
            //A partir de aqui, comenzamos a realizar la accion pedida.
            System.out.println(a.getLocalName()+" : Procesant
EncontrarPacientes...");

            //Extraemos contenido de mensaje inicial de Request.
            List l_ag = a.extractContent(msg);
            SL_Action SLa = (SL_Action) l_ag.get(0);

            //Sabemos que es de tipo TrobarPacientsCompatibles el objeto recibido.
            TrobarPacientsCompatibles tpc = (TrobarPacientsCompatibles) SLa.get_1();

            //Lo hacemos aqui para escribirlo una sola vez.
            //Creamos un ACLMessage de respuesta.
            ACLMessage reply = msg.createReply();

            //Objeto que se ha de rellenar con datos pertinentes que se
            //enviara en la accion AccioConsultaEspecialista.
            ConsultaEspecialista ce = new ConsultaEspecialista();

            //Agent0 consulta BD para comprobar si se puede realizar accion
            //solicitada por AgentCTH.
            switch (EsPosibleExtraerYTratarDatos(tpc,ce))
            {
                //Se puede realizar accion.
                case CORRECTE:

                    //A partir de aqui comenzamos a realizar la peticion pedida.
                    System.out.println("Ag0 envia mensaje Request a Especialista...");

                    //Consulta al DF agente de tipo Especialista que hay en el sistema.

```

```

java.util.List agentes = Busca_en_DF("URV","Especialista");
if (!agentes.isEmpty())
{ //Esta registrado agente de cual se requiere servicio.
  ACLMessage reqMsg=createESPECIALISTAResponse(msg,ce);

  //Instanciamos un comportamiento tipo Request
  ESPECIALISTABehaviour requestBehaviour=
  new ESPECIALISTABehaviour(a,reqMsg);

  //Le pasamos mensaje que se ha recibido para poder contestar a
  agente

  //que ha pedido el servicio a Ag0.
  requestBehaviour.MensajeOrigen=msg;

  //Pasamos vector con pacientes enviados en mensaje, se hace para
  //saber el orden en que se han enviado.
  requestBehaviour.orden_pacientes=orden_pacientes;

  //para pasar AID a subcomportamiento.
  requestBehaviour.AID_agente = AID_agente;

  //Añadimos este comportamiento
  a.addBehaviour(requestBehaviour);
  System.out.println("\nAg0: Request con agente Especialista
iniciado...\n");
}
else
{ //Enviamos mensaje REFUSE.
  //Indicamos que es de refuse, porque no se puede realizar accion.
  reply.setPerformative(ACLMessage.REFUSE);

  //Creamos contenido del mensaje.

  l_ag.clear();
  l_ag.add(SLa);
  l_ag.add(new AgentNoRegistrat());

  //Introducimos contenido en mensaje.
  myAgent.fillContent(reply,l_ag);

  //Enviamos mensaje.
  myAgent.send(reply);
}

break;

case PACIENTINEXISTENT:
  //Generamos un mensaje resultado indicando que no hay ningun
  //paciente que espere recibir organo del mismo tipo, que el que
  //tiene el pasado con IdOrgan.

  //Indicamos que es de refuse, porque no se puede realizar accion.
  reply.setPerformative(ACLMessage.REFUSE);

  //Creamos contenido del mensaje.

  l_ag.clear();
  l_ag.add(SLa);
  l_ag.add(new PatientInexistent());

  //Introducimos contenido en mensaje.
  myAgent.fillContent(reply,l_ag);

  //Enviamos mensaje.
  myAgent.send(reply);

break;

case ORGANINEXISTENT:
  //Generamos un mensaje resultado indicando que organo,

```



```

        //con identificador introducido,no existe en BD.

        //Indicamos que es de refuse, porque no se puede realizar accion.
        reply.setPerformative(ACLMessage.REFUSE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new OrganInexistent());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply,l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

    break;

    case HOSPITALNOAUTORITZAT:
        //Generamos un mensaje refuse indicando que el hospital no esta
        //autorizado a solicitar accion.

        //Indicamos que es de refuse, porque no se puede realizar accion.
        reply.setPerformative(ACLMessage.REFUSE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new HospitalSenseAutoritzar());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply,l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

    break;

    case FALLOSQL:
        //Generamos un mensaje REFUSE indicando que se ha producido
        //un error al acceder a BD.

        reply.setPerformative(ACLMessage.FAILURE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new ErrorAccesBD());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply,l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

    break;
}

}catch (FIPAException fe){
    System.out.println(a.getName() +
        ": Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}
}

/*****

```

```

Manejador de mensajes Request: Metodo done que indica si comportamiento
ha de seguir activo o no.
*****/
public boolean done()
{
    System.out.println(a.getLocalName() +
        ":Comportamiento finalizado correctamente");

    //Ninguna accion a realizar al finalizar el comportamiento.
    return true;
}

/*****
Manejador de mensajes Request: Metodo que reinicia comportamiento.
*****/
public void reset()
{
    System.out.println(a.getLocalName() +
        ": Reinicializado el comportamiento");
}

/*****
Metodo que consulta BD para ver si puede realizarse accion pedida
por AgentCTH.

Retorna:
-1 -> Si se produce cualquier error de tipo SQLException.
0 -> Si se puede realizar accion correctamente.
1 -> No hay candidatos que esperan recibir el organo.
2 -> No existe organo donado con IdOrgan recibido.
*****/
private int EsPosibleExtraerYTratarDatos (TrobarPacientsCompatibles tpc,
ConsultaEspecialista ce)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;
    try{
        //Intentamos conectar con la base de datos.
        con = DriverManager.getConnection (url, login, passwd);
        stmt=con.createStatement();

        int i=CORRECTE; //i contiene el valor a retornar por esta funcion.

        CaracteristiquesBusqueda cb = tpc.get_0();

        i=EsUnOrganoDonado(cb,ce,stmt);

        if (i==CORRECTE) i=HayPacientesCandidatos(cb,ce,stmt);

        //Se consulta si hospital esta autorizado para solicitar accion.
        //Se le pasa por parametro el hospital que ha solicitado accion.
        if (i == CORRECTE) i=HospitalAutorizado(tpc.get_1(),stmt);

        return i;
    }catch (SQLException e){
        System.out.println(a.getLocalName()+
            ": codi d'error "+e.getErrorCode()+
            ".SQLException -> "+e.getMessage());
        return -1;
    }finally{ //Este codigo siempre se ejecutara.
        try{ //Cerramos Statement y conexon con BD.
            if (stmt != null) stmt.close();
            if (con != null) con.close();
        }catch (SQLException e){
            System.out.println(a.getLocalName()+
                ": codi d'error "+e.getErrorCode()+
                ".SQLException -> "+e.getMessage().toString());
        }
    }
}

```

```

    }
}

/*****
Metodo que busca en BD si hay registrado un organo con IdOrgan igual
al que se pasa por parametro en el objeto CaracteristiquesBusqueda
y consta como organo donado.

Retorna: 0 -> si existe organo en BD como donado.
        2 -> si no esta registrado organo en BD como donado.
*****/
private int EsUnOrganDonado (CaracteristiquesBusqueda c,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT D.CodiSSDonant FROM OrgansDonats D"+
        " WHERE D.IdOrgan = '"+u.StringConEspaciosEnBlanco(c.getIdOrgan())+"'");

        if (result.next())
        {
            IntroducirDonante(result.getString(1),
            u.StringConEspaciosEnBlanco(c.getIdOrgan()),ce,stmt);

            return CORRECTE;
        }
        //No existe organo en BD con este IdOrgan.
        System.out.println(a.getLocalName()+"No existe organo en BD.");
        return ORGANINEXISTENT;
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que busca en BD datos relativos a un donante para introducirlo
en objeto de la ontologia para enviar a agente Especialista.
*****/
private void IntroducirDonante (String CodiSS, String IdOrgan,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    ResultSet result2 = null;
    ResultSet result3 = null;
    ResultSet result4 = null;

    try{
        DadesPacients dp = new DadesPacients();
        Pacients p = new Pacients();

        result = stmt.executeQuery ("SELECT O.Tipus, O.Tamany FROM OrgansDonats
0"+
        " WHERE O.IdOrgan = '"+IdOrgan+"'");

        //Estamos seguros de que result!=null, por que lo hemos comprobado
        //previamente.
        result.next();

        //Traduccion de agente Ag0 (Castellano) a Especialista (Catalan).
        String org = result.getString(1);

        if (org.equals("Corazon")){dp.setTipusOrgan("cor");}
        if (org.equals("Higado")){dp.setTipusOrgan("fetge");}
        if (org.equals("Rinyon")){dp.setTipusOrgan("ronyo");}
        if (org.equals("Pulmon")){dp.setTipusOrgan("pulmo");}

        //Introducimos tamaño del organo en el objeto Pacients.
        p.setTamanyOrgan(u.DeFloatADouble(new Float(result.getFloat(2))));

```

```

//Introducimos datos relativos al hospital de ingreso.
result3 = stmt.executeQuery ("SELECT HI.NomHospital, HI.Ciutat"+
                             " FROM HospitalIngres HI"+
                             " WHERE HI.CodiSS = '"+CodiSS+"'");

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result3.next();

result2 = stmt.executeQuery ("SELECT H.NomHospital, H.Ciutat,"+
                             " H.Provincia FROM Hospitals H"+
                             " WHERE H.NomHospital = '"+result3.getString(1)+"'+
                             " AND H.Ciutat = '"+result3.getString(2)+"'");

//Estamos seguros de que result2!=null, por que lo hemos comprobado
//previamente.
result2.next();

p.setNomHospital(u.StringSinEspaciosEnBlanco(result2.getString(1)));
p.setLocalitat(u.StringSinEspaciosEnBlanco(result2.getString(2)));
p.setProvincia(u.StringSinEspaciosEnBlanco(result2.getString(3)));

//Introducimos datos relativos al paciente.
result4 = stmt.executeQuery ("SELECT P.DiaNaixement, P.MesNaixement,"+
                             " P.AnyNaixement, P.TipusSang, P.Pes, P.Antigens"+
                             " FROM Pacients P"+
                             " WHERE P.CodiSS = '"+CodiSS+"'");

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result4.next();

p.setDiaNaixement(new Long(result4.getLong(1)));
p.setMesNaixement(new Long(result4.getLong(2)));
p.setAnyNaixement(new Long(result4.getLong(3)));
p.setTipusSang(result4.getString(4));
p.setPesPersona(u.DeFloatADouble(new Float(result4.getFloat(5))));

//Introducimos antigenos del paciente en objeto de ontologia
//pertinente (Pacients).
char ant[] = result4.getString(6).toCharArray();
if (ant[0]=='0') p.setB0(new Long(0)); else p.setB0(new Long(1));
if (ant[1]=='0') p.setB1(new Long(0)); else p.setB1(new Long(1));
if (ant[2]=='0') p.setDr0(new Long(0)); else p.setDr0(new Long(1));
if (ant[3]=='0') p.setDr1(new Long(0)); else p.setDr1(new Long(1));

//Dia espera en paciente donante no tiene sentido.
p.setDiaEspera(null);
p.setMesEspera(null);
p.setAnyEspera(null);

//Introducimos datos de Paciente Donante en Consulta Especialista.
dp.setnPacients(new Long(0)); //El 0 pertenece a paciente donante.
dp.addLlistaPacients(p);
ce.setPacientes(dp);

}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
    if (result2!=null) result2.close();
    if (result3!=null) result3.close();
    if (result4!=null) result4.close();
}
}

/*****
Metodo que busca en BD si hay registrado pacientes que esperan recibir
un organo del mismo tipo que el que tiene IdOrgan igual
al que se pasa por parametro en el objeto CaracteristiquesBusqueda.

Retorna: 0 -> si existe organo en BD como donado.

```

```

1 -> si no esta registrado organo en BD como donado.
*****/
private int HayPacientesCandidatos (CaracteristiquesBusqueda cb,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery
        ("SELECT O.IdOrganEsperat, O.CodiSSEesperat FROM OrgansEsperats O"+
         " WHERE O.Tipus IN (SELECT O2.Tipus FROM OrgansDonats O2"+
         " WHERE O2.IdOrgan = '"+
         u.StringConEspaciosEnBlanco(cb.getIdOrgan()+"')");

        //Aprovechamos consulta para enviar mensaje a Especialista.
        if (result.wasNull() == false)
        {
            ce.setFuncio_dist(cb.getFuncio());
            ce.setAgente(cb.getAgente());

            Pesos p_distancia = new Pesos();
            Pesos p_pes = new Pesos();
            Pesos p_tamany = new Pesos();
            Pesos p_antigens = new Pesos();
            Pesos p_tempsespera = new Pesos();
            Pesos p_edat = new Pesos();

            p_distancia.setPes(cb.getDistancia());
            p_distancia.setAtribut("Distancia");
            ce.addLlistaPesos(p_distancia);

            p_pes.setPes(cb.getPes());
            p_pes.setAtribut("Pes");
            ce.addLlistaPesos(p_pes);

            p_tamany.setPes(cb.getTamany());
            p_tamany.setAtribut("Tamany");
            ce.addLlistaPesos(p_tamany);

            p_antigens.setPes(cb.getAntigens());
            p_antigens.setAtribut("Antigens");
            ce.addLlistaPesos(p_antigens);

            p_tempsespera.setPes(cb.getTempsEspera());
            p_tempsespera.setAtribut("Temps");
            ce.addLlistaPesos(p_tempsespera);

            p_edat.setPes(cb.getEdat());
            p_edat.setAtribut("Edat");
            ce.addLlistaPesos(p_edat);

            //Para guardar orden de pacientes en mensaje en un vector.
            //Se hace para poder interpretar mensaje que se reciba de
            Especialista.
            int i=0;

            //Primero guardamos valores en estos vectores y luego se los
            //pasamos al metodo IntroducirCandidato, para no tener problemas
            //a la hora de acceder a la BD.
            Vector SS = new Vector();
            Vector Organs = new Vector();

            while (result.next())
            {
                Organs.insertElementAt(new Long
                (result.getLong("IdOrganEsperat")),i);

                String Cod=result.getString("CodiSSEesperat");

```

```

        SS.insertElementAt(Cod,i);

        //Para saber orden de pacientes a la hora de introducirlos en el
        //objeto de la ontologia que enviaremos a Especialista, y así saber
        //la correspondencia id que hace Especialista con CodiSS del
paciente.
        //Especialista asigna id por orden de llegada del paciente.
        ordre_pacients.insertElementAt(Cod,i);
        i+=1;
    }

    for (int j=0;j<i;j++)
    {
        IntroducirCandidato((String) SS.elementAt(j),
        (Long) Organs.elementAt(j),ce,stmt);
    }

    return CORRECTE;
}
//No existe paciente en BD que espere recibir este tipo de organo.
System.out.println(a.getLocalName()+"No existe paciente en BD"+
" que espere recibir este tipo de organo.");
return PACIENTINEXISTENT;
}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
}
}

/*****
Metodo que introduce datos relativos a un candidato
en objeto de la ontologia para enviar a agente Especialista.
*****/
private void IntroducirCandidato (String CodiSS, Long IdOrgan,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    ResultSet result2 = null;
    ResultSet result3 = null;
    ResultSet result4 = null;

    try{
        DadesPacients dp = ce.getPacientes();
        Pacients p = new Pacients();

        result = stmt.executeQuery ("SELECT O.Tamany, O.DiaEspera, "+
        "O.MesEspera, O.AnyEspera FROM OrgansEsperats O"+
        " WHERE O.IdOrganEsperat = "+IdOrgan);

        //Estamos seguros de que result!=null, por que lo hemos comprobado
        //previamente.
        result.next();

        //Introducimos tamaño del organo en el objeto Pacients.
        p.setTamanyOrgan(u.DeFloatADouble(new Float(result.getFloat(1))));

        p.setDiaEspera(new Long(result.getLong(2)));
        p.setMesEspera(new Long(result.getLong(3)));
        p.setAnyEspera(new Long(result.getLong(4)));

        //Introducimos datos relativos al hospital de ingreso.
        result3 = stmt.executeQuery ("SELECT HI.NomHospital, HI.Ciutat"+
        " FROM HospitalIngres HI"+
        " WHERE HI.CodiSS = '"+CodiSS+"'");

        //Estamos seguros de que result!=null, por que lo hemos comprobado
        //previamente.
        result3.next();

        result2 = stmt.executeQuery ("SELECT H.NomHospital, H.Ciutat,"+
        " H.Provincia FROM Hospitals H"+

```

```

        " WHERE H.NomHospital = '"+result3.getString(1)+"'" +
        " AND H.Ciutat = '"+result3.getString(2)+"'";

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result2.next();

p.setNomHospital(u.StringSinEspaciosEnBlanco(result2.getString(1)));
p.setLocalitat(u.StringSinEspaciosEnBlanco(result2.getString(2)));
p.setProvincia(u.StringSinEspaciosEnBlanco(result2.getString(3)));

//Introducimos datos relativos al paciente.
result4 = stmt.executeQuery ("SELECT P.DiaNaixement, P.MesNaixement, "+
        " P.AnyNaixement, P.TipusSang, P.Pes, P.Antigens"+
        " FROM Pacients P"+
        " WHERE P.CodiSS = '"+CodiSS+"'");

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result4.next();

p.setDiaNaixement(new Long(result4.getLong(1)));
p.setMesNaixement(new Long(result4.getLong(2)));
p.setAnyNaixement(new Long(result4.getLong(3)));
p.setTipusSang(result4.getString(4));
p.setPesPersona(u.DeFloatADouble(new Float(result4.getFloat(5))));

//Introducimos antigenos del paciente en objeto de ontologia
//pertinente (Pacients).
char ant[] = result4.getString(6).toCharArray();
if (ant[0]=='0') p.setB0(new Long(0)); else p.setB0(new Long(1));
if (ant[1]=='0') p.setB1(new Long(0)); else p.setB1(new Long(1));
if (ant[2]=='0') p.setDr0(new Long(0)); else p.setDr0(new Long(1));
if (ant[3]=='0') p.setDr1(new Long(0)); else p.setDr1(new Long(1));

//Introducimos datos de Paciente Donante en Consulta Especialista.
dp.setnPacients(new Long(dp.getnPacients().intValue()+1));
dp.addLlistaPacients(p);
ce.setPacientes(dp);

}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
    if (result2!=null) result2.close();
    if (result3!=null) result3.close();
    if (result4!=null) result4.close();
}
}

/*****
Metodo que crea un mensaje Request para solicitar a Especialista
que ordene la lista de pacientes candidatos a recibir un organo,
segun su compatibilidad.
*****/
private ACLMessage createESPECIALISTAREquest
(ACLMessage msg,ConsultaEspecialista ce) throws jade.domain.FIPAException
{

    //Objeto definido en la ontologia (accion que se pasa a Especialista)
    AccioConsultaEspecialista ace =new AccioConsultaEspecialista(ce);

    //Missatge de request
    ACLMessage reqMsg = new ACLMessage(ACLMessage.REQUEST);

    //Consulta al DF agente de tipo Agent0 que hay en el sistema.
    java.util.List agentes = Busca_en_DF("URV","Especialista");
    //Consigue AID de agente buscado.(Estamos seguros de que esta registrado).

```

```

AID aid_especialista = new AID ((String) agentes.get(0));

//Llenamos campos del mensaje
reqMsg.addReceiver(aid_especialista);
//Receptor.
reqMsg.setSender(AID_agent0);
//Language
reqMsg.setLanguage(SLCodec.NAME);
//Ontologia
reqMsg.setOntology(OntologiaAg0.NAME);
//Protocolo
reqMsg.setProtocol("fipa-request");

//Contenido Accion
SL_Action ac=new SL_Action();

//Primer argumento de la accion: agente receptor
ac.set_0 ((AID) aid_especialista);
//Segundo parametro: accion
ac.set_1 (ace);

//Escribimos la accion en el mensaje
List l = new ArrayList(1);
l.add(ac);
//Codificamos el mensaje creado a un mensaje Request de tipo SL string

a.fillContent(reqMsg, l);

//retornamos el mensaje
return reqMsg;
}

/*****
Metodo que busca en el DF todos los nombres de los agentes que cumplen
un cierto servicio.
*****/
public java.util.List Busca_en_DF (String propietario, String tipo)
{
    java.util.List resultado = new ArrayList();
    DFAgentDescription dfad = new DFAgentDescription();
    ServiceDescription sd = new ServiceDescription();
    java.util.List result;

    sd.setOwnership(propietario);
    sd.setType(tipo);
    dfad.addServices(sd); //Aadimos el servicio al DF, para la busqueda.

    try { //Procedemos a buscar en el df lo que queremos (dfad)
        result = jade.domain.DFServiceCommunicator.search(a,dfad);
        if (result.isEmpty()) return resultado;
        for (int i=0;i<result.size();i++) {
            dfad = (DFAgentDescription) result.get(i);

            //Obtenemos el nombre del agente.
            resultado.add(dfad.getName().getName());
        }
    }
    catch (Exception exc) {
        System.out.println (exc.toString());
    }
    return resultado;
}

/*****
Metodo que busca en BD si hay registrado un hospital con NomHospital y
Ciutat igual al que se pasa por parametro en el objeto Hospital, y si
el paciente en cuestion esta ingresado alli.

Retorna: 0 -> si existe hospital => Hospital autorizado.

```



```

        3 -> no existe hospital => Hospital no autorizado..
        *****/
private int HospitalAutorizado (Hospital h,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        Adreza adr = h.getAdreza();
        String ciudad = u.StringConEspaciosEnBlanco(adr.getCiutat());
        result = stmt.executeQuery ("SELECT H.NomHospital FROM Hospitals H"+
        " WHERE H.NomHospital = '"+u.StringConEspaciosEnBlanco(h.getNom())+"'"+
        " and H.Ciutat = '"+ciudad+"'");

        if (result.next())
        {
            //Existe hospital en base de datos y por lo tanto esta autorizado
            //a solicitar accion.
            System.out.println(a.getLocalName()+"Hospital autorizado.");
            return CORRECTE;
        }
        //No existe hospital en BD y no esta autorizado a solicitar accion.
        System.out.println(a.getLocalName()+"Hospital no autorizado.");
        return HOSPITALNOAUTORITZAT;

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}
}

```

#### 8.2.4.5 RespuestaIntroduccionPaciente.java

```

/*****

PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Clase que implementa el comportamiento que responde en el protocolo
request por parte de Agent0, para informar a AgentCTH de posibles
pacientes compatibles con un organo y se inicia protocolo request con
agente Especialista.

*****/

package Agent0.agentes.A0;

//Importacion de clases necesarias para implementar un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;
import jade.domain.FIPAAgentManagement.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.sql.*;
import java.util.*;

//Librerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Importacion de la ontologia.
import Agent0.agentes.A0.Ontology.*;

/*****
Clase que implementa un comportamiento Request:
Petición de AgentCTH para que Agent0 inicie la búsqueda de pacientes
compatibles con un determinado organo.
*****/
public class RespuestaEncontrarPacientesCompatibles extends
FipaRequestResponderBehaviour_sl.ActionHandler
implements FipaRequestResponderBehaviour_sl.Factory
{
    UtilidadesAgent0 u = new UtilidadesAgent0();
    AID AID_agent0 = new AID();
    Agent a=null;

    //Relaciona id asignado a los pacientes por agente Especialista con su
    //CodiSS.
    Vector ordre_pacients = new Vector();

    //Datos para conectarse con la base de datos.
    static String login ="db1";
    static String passwd ="A93652ZY";
    static String url="jdbc:odbc:db1";

    private final static int FALLOSQL=-1;
    private final static int CORRECTE=0;
    private final static int PACIENTINEXISTENT=1;
    private final static int ORGANINEXISTENT=2;
    private final static int HOSPITALNOAUTORIZAT=3;

/*****
Constructor del comportamiento que se encarga de buscar pacientes
compatibles con un determinado organo.
*****/

```

```

    public RespuestaEncontrarPacientesCompatibles (Agent agent, ACLMessage
respuesta)
    {
        super(agent, respuesta);
        a=agent;
        AID_agent0=a.getAID();

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para RESPONDER A Encontrar Pacientes
Compatibles"+
        " con "+ a.getAID());
    }

/*****
Implementacion de la interficie para manejar las acciones recibidas.
*****/
    public FipaRequestResponderBehaviour_sl.ActionHandler create
(ACLMessage msg){

        //Instanciacion a este comportamiento.
        return new RespuestaEncontrarPacientesCompatibles (Ag0.agente0,msg);
    }

/*****
Manejador de mensajes Request: Accion a realizar por este
comportamiento.
*****/
    public void action ()
    {
        try{

            //Obtenemos la peticion del mensaje recibido.
            ACLMessage msg=getRequest();

            System.out.println(a.getLocalName() +
            ": Mensaje de REQUEST recibido: \n"
            + msg.toString() + "\n");

            //Evaluacion del mensaje y composicion de las posibles respuestas.
            //Accion a realizar por Agent0.
            //A partir de aqui, comenzamos a realizar la accion pedida.
            System.out.println(a.getLocalName()+" : Procesant
EncontrarPacientes...");

            //Extraemos contenido de mensaje inicial de Request.
            List l_ag = a.extractContent(msg);
            SL_Action SLa = (SL_Action) l_ag.get(0);

            //Sabemos que es de tipo TrobarPacientsCompatibles el objeto recibido.
            TrobarPacientsCompatibles tpc = (TrobarPacientsCompatibles) SLa.get_1();

            //Lo hacemos aqui para escribirlo una sola vez.
            //Creamos un ACLMessage de respuesta.
            ACLMessage reply = msg.createReply();

            //Objeto que se ha de rellenar con datos pertinentes que se
            //enviara en la accion AccioConsultaEspecialista.
            ConsultaEspecialista ce = new ConsultaEspecialista();

            //Agent0 consulta BD para comprobar si se puede realizar accion
            //solicitada por AgentCTH.
            switch (EsPosibleExtraerYTratarDatos(tpc,ce))
            {
                //Se puede realizar accion.
                case CORRECTE:

                    //A partir de aqui comenzamos a realizar la peticion pedida.
                    System.out.println("Ag0 envia mensaje Request a Especialista...");

                    //Consulta al DF agente de tipo Especialista que hay en el sistema.

```

```

java.util.List agentes = Busca_en_DF("URV","Especialista");
if (!agentes.isEmpty())
{ //Esta registrado agente de cual se requiere servicio.
  ACLMessage reqMsg=createESPECIALISTAResponse(msg,ce);

  //Instanciamos un comportamiento tipo Request
  ESPECIALISTABehaviour requestBehaviour=
  new ESPECIALISTABehaviour(a,reqMsg);

  //Le pasamos mensaje que se ha recibido para poder contestar a
  agente

  //que ha pedido el servicio a Ag0.
  requestBehaviour.MensajeOrigen=msg;

  //Pasamos vector con pacientes enviados en mensaje, se hace para
  //saber el orden en que se han enviado.
  requestBehaviour.orden_pacientes=orden_pacientes;

  //para pasar AID a subcomportamiento.
  requestBehaviour.AID_agente = AID_agente;

  //Añadimos este comportamiento
  a.addBehaviour(requestBehaviour);
  System.out.println("\nAg0: Request con agente Especialista
iniciado...\n");
}
else
{ //Enviamos mensaje REFUSE.
  //Indicamos que es de refuse, porque no se puede realizar accion.
  reply.setPerformative(ACLMessage.REFUSE);

  //Creamos contenido del mensaje.

  l_ag.clear();
  l_ag.add(SLa);
  l_ag.add(new AgentNoRegistrat());

  //Introducimos contenido en mensaje.
  myAgent.fillContent(reply,l_ag);

  //Enviamos mensaje.
  myAgent.send(reply);
}

break;

case PACIENTINEXISTENT:
  //Generamos un mensaje resultado indicando que no hay ningun
  //paciente que espere recibir organo del mismo tipo, que el que
  //tiene el pasado con IdOrgan.

  //Indicamos que es de refuse, porque no se puede realizar accion.
  reply.setPerformative(ACLMessage.REFUSE);

  //Creamos contenido del mensaje.

  l_ag.clear();
  l_ag.add(SLa);
  l_ag.add(new PatientInexistent());

  //Introducimos contenido en mensaje.
  myAgent.fillContent(reply,l_ag);

  //Enviamos mensaje.
  myAgent.send(reply);

break;

case ORGANINEXISTENT:
  //Generamos un mensaje resultado indicando que organo,

```

```

        //con identificador introducido,no existe en BD.

        //Indicamos que es de refuse, porque no se puede realizar accion.
        reply.setPerformative(ACLMessage.REFUSE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new OrganInexistent());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply,l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

    break;

    case HOSPITALNOAUTORITZAT:
        //Generamos un mensaje refuse indicando que el hospital no esta
        //autorizado a solicitar accion.

        //Indicamos que es de refuse, porque no se puede realizar accion.
        reply.setPerformative(ACLMessage.REFUSE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new HospitalSenseAutoritzar());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply,l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

    break;

    case FALLOSQL:
        //Generamos un mensaje REFUSE indicando que se ha producido
        //un error al acceder a BD.

        reply.setPerformative(ACLMessage.FAILURE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new ErrorAccesBD());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply,l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

    break;
}

}catch (FIPAException fe){
    System.out.println(a.getName() +
        ": Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}
}

/*****

```

```

Manejador de mensajes Request: Metodo done que indica si comportamiento
ha de seguir activo o no.
*****/
public boolean done()
{
    System.out.println(a.getLocalName() +
        ":Comportamiento finalizado correctamente");

    //Ninguna accion a realizar al finalizar el comportamiento.
    return true;
}

/*****
Manejador de mensajes Request: Metodo que reinicia comportamiento.
*****/
public void reset()
{
    System.out.println(a.getLocalName() +
        ": Reinicializado el comportamiento");
}

/*****
Metodo que consulta BD para ver si puede realizarse accion pedida
por AgentCTH.

Retorna:
-1 -> Si se produce cualquier error de tipo SQLException.
0 -> Si se puede realizar accion correctamente.
1 -> No hay candidatos que esperan recibir el organo.
2 -> No existe organo donado con IdOrgan recibido.
*****/
private int EsPosibleExtraerYTratarDatos (TrobarPacientsCompatibles tpc,
ConsultaEspecialista ce)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;
    try{
        //Intentamos conectar con la base de datos.
        con = DriverManager.getConnection (url, login, passwd);
        stmt=con.createStatement();

        int i=CORRECTE; //i contiene el valor a retornar por esta funcion.

        CaracteristiquesBusqueda cb = tpc.get_0();

        i=EsUnOrganoDonado(cb,ce,stmt);

        if (i==CORRECTE) i=HayPacientesCandidatos(cb,ce,stmt);

        //Se consulta si hospital esta autorizado para solicitar accion.
        //Se le pasa por parametro el hospital que ha solicitado accion.
        if (i == CORRECTE) i=HospitalAutorizado(tpc.get_1(),stmt);

        return i;
    }catch (SQLException e){
        System.out.println(a.getLocalName()+
            ": codi d'error "+e.getErrorCode()+
            ".SQLException -> "+e.getMessage());
        return -1;
    }finally{ //Este codigo siempre se ejecutara.
        try{ //Cerramos Statement y conexon con BD.
            if (stmt != null) stmt.close();
            if (con != null) con.close();
        }catch (SQLException e){
            System.out.println(a.getLocalName()+
                ": codi d'error "+e.getErrorCode()+
                ".SQLException -> "+e.getMessage().toString());
        }
    }
}

```

```

    }
}

/*****
Metodo que busca en BD si hay registrado un organo con IdOrgan igual
al que se pasa por parametro en el objeto CaracteristiquesBusqueda
y consta como organo donado.

Retorna: 0 -> si existe organo en BD como donado.
        2 -> si no esta registrado organo en BD como donado.
*****/
private int EsUnOrganDonado (CaracteristiquesBusqueda c,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT D.CodiSSDonant FROM OrgansDonats D"+
        " WHERE D.IdOrgan = '"+u.StringConEspaciosEnBlanco(c.getIdOrgan())+"'");

        if (result.next())
        {
            IntroducirDonante(result.getString(1),
            u.StringConEspaciosEnBlanco(c.getIdOrgan()),ce,stmt);

            return CORRECTE;
        }
        //No existe organo en BD con este IdOrgan.
        System.out.println(a.getLocalName()+"No existe organo en BD.");
        return ORGANINEXISTENT;
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que busca en BD datos relativos a un donante para introducirlo
en objeto de la ontologia para enviar a agente Especialista.
*****/
private void IntroducirDonante (String CodiSS, String IdOrgan,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    ResultSet result2 = null;
    ResultSet result3 = null;
    ResultSet result4 = null;

    try{
        DadesPacients dp = new DadesPacients();
        Pacients p = new Pacients();

        result = stmt.executeQuery ("SELECT O.Tipus, O.Tamany FROM OrgansDonats
0"+
        " WHERE O.IdOrgan = '"+IdOrgan+"'");

        //Estamos seguros de que result!=null, por que lo hemos comprobado
        //previamente.
        result.next();

        //Traduccion de agente Ag0 (Castellano) a Especialista (Catalan).
        String org = result.getString(1);

        if (org.equals("Corazon")){dp.setTipusOrgan("cor");}
        if (org.equals("Higado")){dp.setTipusOrgan("fetge");}
        if (org.equals("Rinyon")){dp.setTipusOrgan("ronyo");}
        if (org.equals("Pulmon")){dp.setTipusOrgan("pulmo");}

        //Introducimos tamaño del organo en el objeto Pacients.
        p.setTamanyOrgan(u.DeFloatADouble(new Float(result.getFloat(2))));

```

```

//Introducimos datos relativos al hospital de ingreso.
result3 = stmt.executeQuery ("SELECT HI.NomHospital, HI.Ciutat"+
                             " FROM HospitalIngres HI"+
                             " WHERE HI.CodiSS = '"+CodiSS+"'");

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result3.next();

result2 = stmt.executeQuery ("SELECT H.NomHospital, H.Ciutat,"+
                             " H.Provincia FROM Hospitals H"+
                             " WHERE H.NomHospital = '"+result3.getString(1)+"'+
                             " AND H.Ciutat = '"+result3.getString(2)+"'");

//Estamos seguros de que result2!=null, por que lo hemos comprobado
//previamente.
result2.next();

p.setNomHospital(u.StringSinEspaciosEnBlanco(result2.getString(1)));
p.setLocalitat(u.StringSinEspaciosEnBlanco(result2.getString(2)));
p.setProvincia(u.StringSinEspaciosEnBlanco(result2.getString(3)));

//Introducimos datos relativos al paciente.
result4 = stmt.executeQuery ("SELECT P.DiaNaixement, P.MesNaixement,"+
                             " P.AnyNaixement, P.TipusSang, P.Pes, P.Antigens"+
                             " FROM Pacients P"+
                             " WHERE P.CodiSS = '"+CodiSS+"'");

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result4.next();

p.setDiaNaixement(new Long(result4.getLong(1)));
p.setMesNaixement(new Long(result4.getLong(2)));
p.setAnyNaixement(new Long(result4.getLong(3)));
p.setTipusSang(result4.getString(4));
p.setPesPersona(u.DeFloatADouble(new Float(result4.getFloat(5))));

//Introducimos antigenos del paciente en objeto de ontologia
//pertinente (Pacients).
char ant[] = result4.getString(6).toCharArray();
if (ant[0]=='0') p.setB0(new Long(0)); else p.setB0(new Long(1));
if (ant[1]=='0') p.setB1(new Long(0)); else p.setB1(new Long(1));
if (ant[2]=='0') p.setDr0(new Long(0)); else p.setDr0(new Long(1));
if (ant[3]=='0') p.setDr1(new Long(0)); else p.setDr1(new Long(1));

//Dia espera en paciente donante no tiene sentido.
p.setDiaEspera(null);
p.setMesEspera(null);
p.setAnyEspera(null);

//Introducimos datos de Paciente Donante en Consulta Especialista.
dp.setnPacients(new Long(0)); //El 0 pertenece a paciente donante.
dp.addLlistaPacients(p);
ce.setPacientes(dp);

}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
    if (result2!=null) result2.close();
    if (result3!=null) result3.close();
    if (result4!=null) result4.close();
}
}

/*****
Metodo que busca en BD si hay registrado pacientes que esperan recibir
un organo del mismo tipo que el que tiene IdOrgan igual
al que se pasa por parametro en el objeto CaracteristiquesBusqueda.

Retorna: 0 -> si existe organo en BD como donado.

```



```

1 -> si no esta registrado organo en BD como donado.
*****/
private int HayPacientesCandidatos (CaracteristiquesBusqueda cb,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery
        ("SELECT O.IdOrganEsperat, O.CodiSSEesperat FROM OrgansEsperats O"+
        " WHERE O.Tipus IN (SELECT O2.Tipus FROM OrgansDonats O2"+
        " WHERE O2.IdOrgan = '"+
        u.StringConEspaciosEnBlanco(cb.getIdOrgan()+"')");

        //Aprovechamos consulta para enviar mensaje a Especialista.
        if (result.wasNull() == false)
        {
            ce.setFuncio_dist(cb.getFuncio());
            ce.setAgente(cb.getAgente());

            Pesos p_distancia = new Pesos();
            Pesos p_pes = new Pesos();
            Pesos p_tamany = new Pesos();
            Pesos p_antigens = new Pesos();
            Pesos p_tempsespera = new Pesos();
            Pesos p_edat = new Pesos();

            p_distancia.setPes(cb.getDistancia());
            p_distancia.setAtribut("Distancia");
            ce.addLlistaPesos(p_distancia);

            p_pes.setPes(cb.getPes());
            p_pes.setAtribut("Pes");
            ce.addLlistaPesos(p_pes);

            p_tamany.setPes(cb.getTamany());
            p_tamany.setAtribut("Tamany");
            ce.addLlistaPesos(p_tamany);

            p_antigens.setPes(cb.getAntigens());
            p_antigens.setAtribut("Antigens");
            ce.addLlistaPesos(p_antigens);

            p_tempsespera.setPes(cb.getTempsEspera());
            p_tempsespera.setAtribut("Temps");
            ce.addLlistaPesos(p_tempsespera);

            p_edat.setPes(cb.getEdat());
            p_edat.setAtribut("Edat");
            ce.addLlistaPesos(p_edat);

            //Para guardar orden de pacientes en mensaje en un vector.
            //Se hace para poder interpretar mensaje que se reciba de
            Especialista.
            int i=0;

            //Primero guardamos valores en estos vectores y luego se los
            //pasamos al metodo IntroducirCandidato, para no tener problemas
            //a la hora de acceder a la BD.
            Vector SS = new Vector();
            Vector Organs = new Vector();

            while (result.next())
            {
                Organs.insertElementAt(new Long
                (result.getLong("IdOrganEsperat")),i);

                String Cod=result.getString("CodiSSEesperat");

```

```

        SS.insertElementAt(Cod,i);

        //Para saber orden de pacientes a la hora de introducirlos en el
        //objeto de la ontologia que enviaremos a Especialista, y así saber
        //la correspondencia id que hace Especialista con CodiSS del
paciente.
        //Especialista asigna id por orden de llegada del paciente.
        ordre_pacients.insertElementAt(Cod,i);
        i+=1;
    }

    for (int j=0;j<i;j++)
    {
        IntroducirCandidato((String) SS.elementAt(j),
        (Long) Organs.elementAt(j),ce,stmt);
    }

    return CORRECTE;
}
//No existe paciente en BD que espere recibir este tipo de organo.
System.out.println(a.getLocalName()+"No existe paciente en BD"+
" que espere recibir este tipo de organo.");
return PACIENTINEXISTENT;
}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
}
}

/*****
Metodo que introduce datos relativos a un candidato
en objeto de la ontologia para enviar a agente Especialista.
*****/
private void IntroducirCandidato (String CodiSS, Long IdOrgan,
ConsultaEspecialista ce,Statement stmt) throws SQLException
{
    ResultSet result = null;
    ResultSet result2 = null;
    ResultSet result3 = null;
    ResultSet result4 = null;

    try{
        DadesPacients dp = ce.getPacientes();
        Pacients p = new Pacients();

        result = stmt.executeQuery ("SELECT O.Tamany, O.DiaEspera, "+
        "O.MesEspera, O.AnyEspera FROM OrgansEsperats O"+
        " WHERE O.IdOrganEsperat = "+IdOrgan);

        //Estamos seguros de que result!=null, por que lo hemos comprobado
        //previamente.
        result.next();

        //Introducimos tamaño del organo en el objeto Pacients.
        p.setTamanyOrgan(u.DeFloatADouble(new Float(result.getFloat(1))));

        p.setDiaEspera(new Long(result.getLong(2)));
        p.setMesEspera(new Long(result.getLong(3)));
        p.setAnyEspera(new Long(result.getLong(4)));

        //Introducimos datos relativos al hospital de ingreso.
        result3 = stmt.executeQuery ("SELECT HI.NomHospital, HI.Ciutat"+
        " FROM HospitalIngres HI"+
        " WHERE HI.CodiSS = '"+CodiSS+"'");

        //Estamos seguros de que result!=null, por que lo hemos comprobado
        //previamente.
        result3.next();

        result2 = stmt.executeQuery ("SELECT H.NomHospital, H.Ciutat,"+
        " H.Provincia FROM Hospitals H"+

```

```

        " WHERE H.NomHospital = '"+result3.getString(1)+"'" +
        " AND H.Ciutat = '"+result3.getString(2)+"'";

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result2.next();

p.setNomHospital(u.StringSinEspaciosEnBlanco(result2.getString(1)));
p.setLocalitat(u.StringSinEspaciosEnBlanco(result2.getString(2)));
p.setProvincia(u.StringSinEspaciosEnBlanco(result2.getString(3)));

//Introducimos datos relativos al paciente.
result4 = stmt.executeQuery ("SELECT P.DiaNaixement, P.MesNaixement, "+"
    " P.AnyNaixement, P.TipusSang, P.Pes, P.Antigens"+"
    " FROM Pacients P"+"
    " WHERE P.CodiSS = '"+CodiSS+"'");

//Estamos seguros de que result!=null, por que lo hemos comprobado
//previamente.
result4.next();

p.setDiaNaixement(new Long(result4.getLong(1)));
p.setMesNaixement(new Long(result4.getLong(2)));
p.setAnyNaixement(new Long(result4.getLong(3)));
p.setTipusSang(result4.getString(4));
p.setPesPersona(u.DeFloatADouble(new Float(result4.getFloat(5))));

//Introducimos antigenos del paciente en objeto de ontologia
//pertinente (Pacients).
char ant[] = result4.getString(6).toCharArray();
if (ant[0]=='0') p.setB0(new Long(0)); else p.setB0(new Long(1));
if (ant[1]=='0') p.setB1(new Long(0)); else p.setB1(new Long(1));
if (ant[2]=='0') p.setDr0(new Long(0)); else p.setDr0(new Long(1));
if (ant[3]=='0') p.setDr1(new Long(0)); else p.setDr1(new Long(1));

//Introducimos datos de Paciente Donante en Consulta Especialista.
dp.setnPacients(new Long(dp.getnPacients().intValue()+1));
dp.addLlistaPacients(p);
ce.setPacientes(dp);

}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
    if (result2!=null) result2.close();
    if (result3!=null) result3.close();
    if (result4!=null) result4.close();
}
}

/*****
Metodo que crea un mensaje Request para solicitar a Especialista
que ordene la lista de pacientes candidatos a recibir un organo,
segun su compatibilidad.
*****/
private ACLMessage createESPECIALISTAREquest
(ACLMessage msg,ConsultaEspecialista ce) throws jade.domain.FIPAException
{

    //Objeto definido en la ontologia (accion que se pasa a Especialista)
    AccioConsultaEspecialista ace =new AccioConsultaEspecialista(ce);

    //Missatge de request
    ACLMessage reqMsg = new ACLMessage(ACLMessage.REQUEST);

    //Consulta al DF agente de tipo Agent0 que hay en el sistema.
    java.util.List agentes = Busca_en_DF("URV","Especialista");
    //Consigue AID de agente buscado.(Estamos seguros de que esta registrado).

```

```

AID aid_especialista = new AID ((String) agentes.get(0));

//Llenamos campos del mensaje
reqMsg.addReceiver(aid_especialista);
//Receptor.
reqMsg.setSender(AID_agent0);
//Language
reqMsg.setLanguage(SLCodec.NAME);
//Ontologia
reqMsg.setOntology(OntologiaAg0.NAME);
//Protocolo
reqMsg.setProtocol("fipa-request");

//Contenido Accion
SL_Action ac=new SL_Action();

//Primer argumento de la accion: agente receptor
ac.set_0 ((AID) aid_especialista);
//Segundo parametro: accion
ac.set_1 (ace);

//Escribimos la accion en el mensaje
List l = new ArrayList(1);
l.add(ac);
//Codificamos el mensaje creado a un mensaje Request de tipo SL string

a.fillContent(reqMsg, l);

//retornamos el mensaje
return reqMsg;
}

/*****
Metodo que busca en el DF todos los nombres de los agentes que cumplen
un cierto servicio.
*****/
public java.util.List Busca_en_DF (String propietario, String tipo)
{
    java.util.List resultado = new ArrayList();
    DFAgentDescription dfad = new DFAgentDescription();
    ServiceDescription sd = new ServiceDescription();
    java.util.List result;

    sd.setOwnership(propietario);
    sd.setType(tipo);
    dfad.addServices(sd); //Aadimos el servicio al DF, para la busqueda.

    try { //Procedemos a buscar en el df lo que queremos (dfad)
        result = jade.domain.DFServiceCommunicator.search(a,dfad);
        if (result.isEmpty()) return resultado;
        for (int i=0;i<result.size();i++) {
            dfad = (DFAgentDescription) result.get(i);

            //Obtenemos el nombre del agente.
            resultado.add(dfad.getName().getName());
        }
    }
    catch (Exception exc) {
        System.out.println (exc.toString());
    }
    return resultado;
}

/*****
Metodo que busca en BD si hay registrado un hospital con NomHospital y
Ciutat igual al que se pasa por parametro en el objeto Hospital, y si
el paciente en cuestion esta ingresado alli.

Retorna: 0 -> si existe hospital => Hospital autorizado.

```

```

        3 -> no existe hospital => Hospital no autorizado..
        *****/
private int HospitalAutorizado (Hospital h,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        Adreza adr = h.getAdreza();
        String ciudad = u.StringConEspaciosEnBlanco(adr.getCiutat());
        result = stmt.executeQuery ("SELECT H.NomHospital FROM Hospitals H"+
        " WHERE H.NomHospital = '"+u.StringConEspaciosEnBlanco(h.getNom())+"'"+
        " and H.Ciutat = '"+ciudad+"'");

        if (result.next())
        {
            //Existe hospital en base de datos y por lo tanto esta autorizado
            //a solicitar accion.
            System.out.println(a.getLocalName()+"Hospital autorizado.");
            return CORRECTE;
        }
        //No existe hospital en BD y no esta autorizado a solicitar accion.
        System.out.println(a.getLocalName()+"Hospital no autorizado.");
        return HOSPITALNOAUTORITZAT;
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}
}

```

#### 8.2.4.6 RespuestaOperacionRealizada.java

```
/******
```

```
PFC Agent0.  
Cristóbal Ledesma Oljas  
8/10/02
```

Clase que implementa el comportamiento que responde en el protocolo request por parte de Agent0, para que AgentCTH tenga constancia, de como ha ido la introduccion de datos relativos a una operacion. Ademas de realizar dicha accion.

```
*****/
```

```
package Agent0.agentes.A0;
```

```
//Importacion de clases necesarias para implementar un agente.
```

```
import jade.core.*;  
import jade.core.behaviours.*;  
import jade.domain.*;  
import jade.domain.FIPAAgentManagement.*;
```

```
import jade.proto.*;  
import jade.onto.sl.sl.*;  
import java.sql.*;  
import java.util.*;
```

```
//Librerias necesarias para construir un mensaje.
```

```
import jade.lang.acl.*;  
import jade.lang.sl.sl.*;
```

```
//Importacion de la ontologia.
```

```
import Agent0.agentes.A0.Ontology.*;
```

```
/******
```

Clase que implementa un comportamiento Request: Peticion de AgentCTH para que Agent0 anote en su base de datos los datos relativos a una operacion.

```
*****/
```

```
public class RespuestaOperacionRealizada extends  
FipaRequestResponderBehaviour_sl.ActionHandler  
implements FipaRequestResponderBehaviour_sl.Factory  
{
```

```
    AID AID_agent0 = new AID();  
    Agent a=null;  
    UtilidadesAgent0 u =new UtilidadesAgent0();
```

```
    private final static int FALLOSQL=-1;  
    private final static int CORRECTE=0;  
    private final static int ORGANINEXISTENT=1;  
    private final static int PACIENTMORT=2;  
    private final static int PACIENTINEXISTENT=3;  
    private final static int ORGANSSENSEASSIGNAR=4;  
    private final static int ORGANNNOESTADISPONIBLE=5;  
    private final static int HOSPITALNOAUTORIZAT=6;
```

```
    //Datos para conectarse con la base de datos.
```

```
    static String login ="db1";  
    static String passwd ="A93652ZY";  
    static String url="jdbc:odbc:db1";
```

```
/******
```

Constructor de la clase que implementa este comportamiento.

```
*****/
```

```
    public RespuestaOperacionRealizada (Agent agent, ACLMessage respuesta)  
    {  
        super(agent,respuesta);  
        a=agent;  
        AID_agent0=a.getAID();
```

```

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para RESPONDER A operacion realizada con "
        + AID_agent0);
    }

/*****
Clase que implementa un comportamiento Request: Peticion de AgentCTH para
que Agent0 anote en su base de datos relativos a una operacion.
*****/
    public FipaRequestResponderBehaviour_sl.ActionHandler create(ACLMessage msg)
    {
        System.out.println(AID_agent0+ ": Mensaje inicial :"+ msg.toString());
        return new RespuestaOperacionRealizada (Ag0.agente0,msg);
    }

/*****
Manejador de mensajes Request asociados a este Comportamiento.
*****/
    public void action ()
    {
        try{
            //Obtenemos la peticion del mensaje recibido.
            ACLMessage msg=getRequest();

            System.out.println(a.getLocalName() +
            ": Mensaje de REQUEST recibido: \n"
            + msg.toString() + "\n");

            //Accion a realizar por Agent0.
            //A partir de aqui, comenzamos a realizar la acci0n pedida.
            System.out.println("Procesant OperacionRealizada...");
            //Extraemos contenido de mensaje inicial de Request.
            List l_ag = a.extractContent(msg);
            SL_Action SLa = (SL_Action) l_ag.get(0);

            //Sabemos que es de tipo IntroducirPacient el objeto recibido.
            OperacioRealitzada or = (OperacioRealitzada) SLa.get_1();

            //Lo hacemos aqui para escribirlo una sola vez.
            //Creamos un ACLMessage de respuesta.
            ACLMessage reply = msg.createReply();

            //Evaluacion del mensaje y composicion de la posibles respuestas.
            switch (EsPosibleExtraerYTratarDatos(or))
            {
                case CORRECTE:
                    //Enviamos el mensaje: AGREE.
                    //Obtenemos la accion.
                    l_ag.clear();
                    l_ag.add(SLa);
                    l_ag.add(new SL_True());
                    myAgent.fillContent(reply, l_ag);

                    sendReply(ACLMessage.AGREE,reply.getContent());

                    if ((ExtraerYTratarDatos(or))==0)
                    {
                        //Generamos un mensaje con accion realizada correctamente
                        //Lo llenamos con contenido de mensaje original
                        SL_Done d = new SL_Done();
                        d.set_0(SLa);
                        l_ag.clear();
                        l_ag.add(d);

                        reply.setPerformative(ACLMessage.INFORM);
                        myAgent.fillContent(reply, l_ag);

                        myAgent.send(reply);
                    }
            }
        }
    }

```

```

else
{
    //Indicamos que es de failure, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.FAILURE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new ErrorAccesBD());
    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
}
break;
case ORGANINEXISTENT:
    //Generamos un mensaje resultado indicando que organo,
    //con identificador introducido, no existe en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new OrganInexistent());
    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
break;
case PACIENTMORT:
    //Generamos un mensaje resultado indicando que paciente,
    //con identificador introducido, consta como muerto en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientMort());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
break;
case PACIENTINEXISTENT:
    //Generamos un mensaje resultado indicando que paciente,
    //con identificador introducido, no existe en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientInexistent());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

```



```

        //Enviamos mensaje.
        myAgent.send(reply);
        break;

case ORGANSSENSEASSIGNAR:
    //Generamos un mensaje refuse indicando que tipo de organo,
    //no lo tiene asignado ningun paciente.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientSenseOrganAssignat());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
    break;

case ORGANNOSTADISPONIBLE:
    //Generamos un mensaje refuse indicando que tipo de organo,
    //no lo tiene asignado ningun paciente.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new OrganNoDisponible());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
    break;

case HOSPITALNOAUTORITZAT:
    //Generamos un mensaje refuse indicando que el hospital no esta
    //autorizado a solicitar accion.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new HospitalSenseAutoritzar());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);

    break;

case FALLOSQL:
    //Generamos un mensaje REFUSE indicando que se ha producido
    //un error al acceder a BD.
    reply.setPerformative(ACLMessage.REFUSE);

```

```

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new ErrorAccesBD());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply, l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

        break;
    }
} catch (FIPAException fe){
    System.out.println(a.getName() + ": Error en el manegador de
missatges:");
    System.out.println(fe.getMessage().toString());
}
}

/*****
Manejador de mensajes Request: Metodo done que indica si comportamiento
ha de seguir activo o no.
*****/
public boolean done()
{
    System.out.println(a.getLocalName() +
        ":Comportamiento finalizado correctamente.");
    //Ninguna accion a realizar al finalizar el comportamiento.
    return true;
}

/*****
Manejador de mensajes Request: Metodo que reinicia comportamiento.
*****/
public void reset()
{
    System.out.println(a.getLocalName() + ": Reinicializado el
comportamiento");
}

/*****
Metodo que mira si es posible actualizar base de datos a partir de los
datos introducidos por usuario.

Retorna:

0 -> Si se puede introducir datos relativos a un organo.
1 -> Si el organo introducido ya estaba registrado en la base de datos.
2 -> Si el paciente que necesita el organo esta muerto.
-1 -> Si hay algun error de tipo SQLException.
*****/
private int EsPosibleExtraerYTratarDatos (OperacioRealitzada or)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;
    try{
        //Intentamos Conectar con la base de datos.
        con = DriverManager.getConnection (url, login, passwd);
        stmt=con.createStatement();
        Operacio op = or.get_0();

        //Se consulta si existe organo en BD.
        int i = ExisteixOrgan(op, stmt);

        //Se consulta si organo esta asignado a un paciente.
        if (i == CORRECTE) i=OrganUtilitzat(op, stmt);
    }
}

```

```

        //Se consulta si organo esta asignado a un paciente.
        if (i == CORRECTE) i=OrganAssignat(op,stmt);

        //Paciente registrado con estado muerto.
        if (i == CORRECTE) i=RegistratPacientMort(op,stmt);

        //Se consulta si hospital esta autorizado para solicitar accion.
        if (i==CORRECTE) i = HospitalAutorizado(op,stmt);

        return i;

    }catch (SQLException e){
        System.out.println(a.getLocalName()+" : codi d'error "
            +e.getErrorCode()+
            ".SQLException -> "+e.getMessage());

        return -1;
    }finally{ //Este codigo siempre se ejecutara.
        try{ //Cerramos Statement y conexion con BD.
            if (stmt != null) stmt.close();
            if (con != null) con.close();
        }catch (SQLException e){
            System.out.println(a.getLocalName()+" : codi d'error "
                +e.getErrorCode()+
                ".SQLException -> "+e.getMessage().toString());
            return -1;
        }
    }
}

/*****
Metodo que actualiza base de datos a partir de los datos introducidos
por usuario.

Retorna:
    0 -> Si se ha podido introducir datos relativos a un organo.
    -1 -> Si hay algun error de tipo SQLException.
*****/
private int ExtraerYTratarDatos (OperacioRealitzada or)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;

    Operacio op =or.get_0();
    try{
        //Intentamos Conectar con la base de datos.
        con = DriverManager.getConnection (url,login,passwd);
        stmt=con.createStatement();

        //Modificamos estado 'Organo Asignado' a 'Operado'
        //en la relacion Assignacio.
        ModificarFilaAssignacio(op,stmt);

        //Insertar fila en la relacion Operacio.
        InsertarFilaOperacio(op,stmt);

        return 0;
    }catch (SQLException e){
        System.out.println(a.getLocalName()+" : codi d'error "
            +e.getErrorCode()+
            ".SQLException -> "+e.getMessage());
        return -1;
    }finally{ //Este codigo siempre se ejecutara.
        try{ //Cerramos Statement y conexion con BD.
            if (stmt != null) stmt.close();
            if (con != null) con.close();
        }catch (SQLException e){
            System.out.println(a.getLocalName()+" : codi d'error "+

```

```

        e.getErrorCode()+
        ".SQLException -> "+e.getMessage().toString());
        return -1;
    }
}

/*****
Metodo que busca en BD si hay registrado un organo igual al que se
pasa por parametro en el objeto Patient.

Retorna: 0 -> No hay coincidencia.
         1 -> Hay coincidencia.
*****/
private int ExisteixOrgan (Operacio op, Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT O.IdOrgan FROM OrgansDonats O"+
        " WHERE O.IdOrgan = '"+
        u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+"'");

        if (result.next())
        {
            System.out.println(a.getLocalName()+"Existe Organo en BD.");
            return CORRECTE;
        }

        //No existeix organ a BD amb aquest IdOrgan.
        System.out.println(a.getLocalName()+"No existe Organo en BD.");
        return ORGANINEXISTENT;
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que modifica un registro de la tabla Necesita de BD. Cambia
estado 'Organ Asignado' -> 'Operado, del paciente vivo que tenia
asignado el organo.
*****/
private void ModificarFilaAssignacio (Operacio op, Statement stmt) throws
SQLException
{
    int i=stmt.executeUpdate ("UPDATE Assignacio SET"+
    " Estat = 'Operado'"+
    " WHERE IdOrgan = '"+
    u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+"'"+
    " and CodiSS IN (SELECT P.CodiSS FROM Patients P"+
    " WHERE P.Estat='Vivo')");

    System.out.println(a.getLocalName()+
    "Se ha modificado registro en Necesita. i->"+i);
}

/*****
Metodo que busca en BD si un paciente esta registrado con el estado
Muerto.

Retorna: 0 -> Si paciente esta registrado con estado vivo.
         1 -> Si paciente esta registrado con estado muerto.
         2 -> Si paciente no esta registrado en BD.
*****/
private int RegistratPatientMort(Operacio op,Statement stmt) throws
SQLException
{
    ResultSet result = null;

```

```

//Sera booleano que indicara si todos los pacientes a los que
//ha sido asignado el organo han fallecido.
int mort=0;
try{
    result = stmt.executeQuery ("SELECT P.Estat FROM Pacients P"+
    " WHERE P.CodiSS IN (SELECT A.CodiSS FROM Assignacio A"+
    " WHERE A.IdOrgan = '"+
    u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+"'+
    " and A.Estat = 'Organo Asignado')");

    //Miramos si hay algun paciente vivo con el organo asignado.
    while (result.next())
    {
        String estat = result.getString(1);

        if (estat.equals("Muerto"))
        {
            mort=1;
        }
        else
        {
            return CORRECTE;
        }
    }

    if (mort == 1)
    { //Todos los pacientes a los que se le ha asignado el organo han
      //fallecido.
      return PACIENTMORT;
    }
    else
    {
        System.out.println("El pacient no existeix a BD."+
        " Error greu de consistencia de BD.");
        return PACIENTINEXISTENT;
    }
}finally{ //Se necesita cerrar el tipo de objeto ResultSet
    if (result!=null) result.close();
}

}

/*****
Metodo que busca en BD si el organo esta asignado a un paciente.

Retorna:
    0 -> Si organ esta asignado.
    4 -> Si organ no esta asignado a nadie.
*****/
private int OrganAssignat (Operacio op,Statement stmt) throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT A.CodiSS FROM Assignacio A"+
        " WHERE A.IdOrgan = '"+
        u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+"'+
        " and A.Estat = 'Organo Asignado')");

        if (result.next())
        {
            System.out.println("Organo esta asignado a paciente");
            return CORRECTE;
        }
        else
        {
            System.out.println("Organo no esta asignado a ningun paciente");
            return ORGANSSENSEASSIGNAR;
        }
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet

```

```

        if (result!=null) result.close();
    }
}

/*****
Metodo que inserta una fila en la tabla Operacio.
*****/
private void InsertarFilaOperacio (Operacio op, Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{

        Hospital h = op.getHospital();
        Adreza ad = h.getAdreza();
        result = stmt.executeQuery ("SELECT A.CodiSS FROM Assignacio A"+
        " WHERE A.IdOrgan = '"+
        u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+"'"+
        " and A.Estat = 'Operado'");

        //Estamos seguros de que existe registro en result.
        result.next();

        int i=stmt.executeUpdate ("INSERT INTO Operacio" +
        " VALUES ('"+u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+
        "','"+u.StringConEspaciosEnBlanco(result.getString(1))+ "','"+
        u.StringConEspaciosEnBlanco(h.getNom())+"','"+
        u.StringConEspaciosEnBlanco(ad.getCiutat())+"','"+
        op.getDiaOperacio()+"','"+op.getMesOperacio()+"','"+
        +op.getAnyOperacio()+"','"+
        op.getHoraOperacio()+"','"+op.getMinutOperacio()+"')");

        System.out.println(a.getLocalName()+
        "Se ha insertado registro en Operacio. i->" +i);

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que busca en BD si hay registrado un organo que se puede
asignar a otro paciente, o bien ya esta siendo utilizado.
Concretamente se mira si el organo ya se ha trasplantado a otro
paciente.

Retorna: 0 -> si organo no esta siendo utilizado por nadie.
        5 -> si organo esta siendo utilizado por otro paciente.
*****/
private int OrganUtilitzat (Operacio op,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{

        //Se mira si el organo ha sido trasplantado o rechazado.
        result =stmt.executeQuery ("SELECT A.CodiSS FROM Assignacio A"+
        " WHERE A.IdOrgan = '"+
        u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+"'"+
        " and (A.Estat = 'Operado' or A.Estat='Rechazado')");

        if (result.next())
        {
            //El organo no puede ser asignado a otro paciente, porque
            //ya se ha utilizado.
            System.out.println(a.getLocalName()+
            "Organo ya ha sido utilizado y no se puede reasignar.");
            return ORGANOESTADISPONIBLE;
        }
        //El organo no ha sido utilizado por ningun paciente.
    }
}

```

```

        System.out.println(a.getLocalName()+"Organo no ha sido utilizado.");
        return CORRECTE;

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que busca en BD si hay registrado un hospital con NomHospital y
Ciutat igual al que se pasa por parametro en el objeto Hospital, y si
el paciente en cuestion esta ingresado alli.

Retorna: 0 -> si existe hospital => Hospital autorizado.
        6 -> no existe hospital => Hospital no autorizado..
*****/
private int HospitalAutorizado (Operacio op,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    ResultSet result2 = null;
    try{

        Hospital h = op.getHospital();
        Adreza adr = h.getAdreza();
        String ciudad = u.StringConEspaciosEnBlanco(adr.getCiutat());

        //Se mira el paciente que tiene asignado el ¢rgano en ese momento
        //y esta vivo. Ya que pueden tenerlo asignado varios pacientes, pero
        //solo uno de ellos estara vivo.
        result =stmt.executeQuery ("SELECT A.CodiSS FROM Assignacio A"+
        " WHERE A.IdOrgan = '"+
        u.StringConEspaciosEnBlanco(op.getIdOrganTrasplantat())+"'"+
        " and A.Estat = 'Organo Asignado'"+
        " and A.CodiSS IN (SELECT P.CodiSS FROM Pacients P"+
        " WHERE P.Estat = 'Vivo')");
        //Estamos seguros de que result!=null, porque lo hemos comprobado
        //previamente.
        result.next();

        String cod = u.StringConEspaciosEnBlanco(result.getString(1));

        result2 = stmt.executeQuery ("SELECT H.NomHospital FROM HospitalIngres
H"+
        " WHERE H.NomHospital = '"+u.StringConEspaciosEnBlanco(h.getNom())+"'"+
        " and H.Ciutat = '"+ciudad+"'"+
        " and H.CodiSS = '"+cod+"'");

        if (result2.next())
        {
            //Existe paciente ingresado en hospital; por lo tanto, esta autorizado
            //a solicitar accion.
            System.out.println(a.getLocalName()+"Hospital autorizado.");
            return CORRECTE;
        }
        //Paciente no ingresado en ese hospital y no esta autorizado
        //a solicitar accion.
        System.out.println(a.getLocalName()+"Hospital no autorizado.");
        return HOSPITALNOAUTORITZAT;

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
        if (result2!=null) result2.close();
    }
}
}

```





#### 8.2.4.7 RespuestaOrganoRechazado.java

/\*\*\*\*\*

PFC Agent0.  
Cristóbal Ledesma Oljas  
8/10/02

Clase que implementa el comportamiento que responde en el protocolo request por parte de Agent0, para que AgentCTH tenga constancia, de como ha ido la introduccion de datos relativos al rechazo de un organo por parte de un paciente.  
Ademas de realizar dicha accion.

\*\*\*\*\*/

package Agent0.agentes.A0;

//Importacion de clases necesarias para implementar un agente.

import jade.core.\*;  
import jade.core.behaviours.\*;  
import jade.domain.\*;  
import jade.domain.FIPAAgentManagement.\*;

import jade.proto.\*;  
import jade.onto.sl.sl.\*;  
import java.sql.\*;  
import java.util.\*;

//Librerias necesarias para construir un mensaje.

import jade.lang.acl.\*;  
import jade.lang.sl.sl.\*;

//Importacion de la ontologia.

import Agent0.agentes.A0.Ontology.\*;

/\*\*\*\*\*

Clase que implementa un comportamiento Request: Peticion de AgentCTH para que Agent0 anote en su base de datos el rechazo de un organo por parte de un paciente.

\*\*\*\*\*/

public class RespuestaOrganoRechazado extends  
FipaRequestResponderBehaviour\_sl.ActionHandler  
implements FipaRequestResponderBehaviour\_sl.Factory  
{

AID AID\_agent0 = new AID();  
Agent a=null;  
UtilidadesAgent0 u = new UtilidadesAgent0();

private final static int FALLOSQL=-1;  
private final static int CORRECTE=0;  
private final static int ORGANINEXISTENT=1;  
private final static int PACIENTMORT=2;  
private final static int PACIENTINEXISTENT=3;  
private final static int ORGANSENSETRASPLANTAR=4;  
private final static int ORGANJASHAREBUTXAT=5;  
private final static int HOSPITALNOAUTORITZAT=6;

//Datos para conectarse con la base de datos.

static String login ="db1";  
static String passwd ="A93652ZY";  
static String url ="jdbc:odbc:db1";

/\*\*\*\*\*

Constructor de la clase que implementa este comportamiento.

\*\*\*\*\*/

public RespuestaOrganoRechazado (Agent agent, ACLMessage respuesta)  
{

```

        super(agent, respuesta);
        a=agent;
        AID_agent0=a.getAID();

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para RESPONDER A organo rechazado con "
        + AID_agent0);
    }

/*****
Implementacion de la interficie para manejar las acciones recibidas.
*****/
public FipaRequestResponderBehaviour_sl.ActionHandler
create(ACLMessage msg){
    System.out.println(AID_agent0+ ": Mensaje inicial :"+ msg.toString());
    return new RespuestaOrganoRechazado (Ag0.agente0,msg);
}

/*****
Manejador de mensajes Request: Accion a realizar por este
comportamiento.
*****/
public void action ()
{
    try{
        //Obtenemos la peticion del mensaje recibido.
        ACLMessage msg=getRequest();

        System.out.println(a.getLocalName() +
        ": Mensaje de REQUEST recibido: \n"
        + msg.toString() + "\n");

        //Accion a realizar por Agent0.
        //A partir de aqui, comenzamos a realizar la accion pedida.
        System.out.println("Procesant OperacionRealizada...");

        //Extraemos contenido de mensaje inicial de Request.
        List l_ag = a.extractContent(msg);
        SL_Action SLa = (SL_Action) l_ag.get(0);

        //Sabemos que es de tipo IntroducirPacient el objeto recibido.
        OrganRebutxat or = (OrganRebutxat) SLa.get_1();

        //Lo hacemos aqui para escribirlo una sola vez.
        //Creamos un ACLMessage de respuesta.
        ACLMessage reply = msg.createReply();

        switch (EsPosibleExtraerYTratarDatos(or))
        {
            case CORRECTE:
                //Enviamos el mensaje: AGREE.
                //Obtenemos la accion.
                l_ag.clear();
                l_ag.add(SLa);
                l_ag.add(new SL_True());
                myAgent.fillContent(reply, l_ag);

                sendReply(ACLMessage.AGREE, reply.getContent());

                if ((ExtraerYTratarDatos(or))==0)
                {
                    //Generamos un mensaje con accion realizada correctamente
                    //Lo llenamos con contenido de mensaje original
                    SL_Done d = new SL_Done();
                    d.set_0(SLa);
                    l_ag.clear();
                    l_ag.add(d);

                    reply.setPerformative(ACLMessage.INFORM);
                    myAgent.fillContent(reply, l_ag);
                }
            }
        }

```

```

        myAgent.send(reply);
    }
    else
    {
        //Indicamos que es de failure, porque no se puede realizar accion.
        reply.setPerformative(ACLMessage.FAILURE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new ErrorAccesBD());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply, l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);
    }
    break;

case ORGANINEXISTENT:
    //Generamos un mensaje resultado indicando que organo,
    //con identificador introducido, no existe en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new OrganInexistent());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
    break;

case PACIENTMORT:
    //Generamos un mensaje resultado indicando que paciente,
    //con identificador introducido, esta registrado en estado
    //fallecido en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientMort());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);
    break;

case PACIENTINEXISTENT:
    //Generamos un mensaje resultado indicando que paciente,
    //con identificador introducido, no existe en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

```

```

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new PacientInexistent());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case ORGANSENSETRASPLANTAR:
//Generamos un mensaje refuse indicando que tipo de organo,
//que se intenta introducir no se le ha trasplantado a nadie.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new PacientSenseOperar());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case ORGANJASHAREBUTXAT:
//Generamos un mensaje refuse indicando que tipo de organo,
//que se intenta introducir no se le ha trasplantado a nadie.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new OrganJaEstavaRebutxat());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

//Enviamos mensaje.
myAgent.send(reply);
break;

case HOSPITALNOAUTORITZAT:
//Generamos un mensaje refuse indicando que el hospital
//no esta autorizado a solicitar accion.

//Indicamos que es de refuse, porque no se puede realizar accion.
reply.setPerformative(ACLMessage.REFUSE);

//Creamos contenido del mensaje.

l_ag.clear();
l_ag.add(SLa);
l_ag.add(new HospitalSenseAutoritzar());

//Introducimos contenido en mensaje.
myAgent.fillContent(reply, l_ag);

```

```

        //Enviamos mensaje.
        myAgent.send(reply);

    break;

    case FALLOSQL:
        //Generamos un mensaje REFUSE indicando que se ha producido
        //un error al acceder a BD.

        reply.setPerformative(ACLMessage.REFUSE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new ErrorAccesBD());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply, l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);

    break;
}
}catch (FIPAException fe){
    System.out.println(a.getName() +
        ": Error en el manejador de missatges:");
    System.out.println(fe.getMessage().toString());
}
}

/*****
Manejador de mensajes Request: Metodo done que indica si comportamiento
ha de seguir activo o no.
*****/
public boolean done()
{
    System.out.println(a.getLocalName() +
        ":Comportamiento finalizado correctamente.");
    //Ninguna accion a realizar al finalizar el comportamiento.
    return true;
}

/*****
Manejador de mensajes Request: Metodo que reinicia comportamiento.
*****/
public void reset()
{
    System.out.println(a.getLocalName() +
        ": Reinicializado el comportamiento");
}

/*****
Metodo que mira si se puede realizar accion a partir de los
datos introducidos por el usuario.

Retorna:

    0 -> Si se puede introducir datos relativos a un organo.
    1 -> Si el organo introducido ya estaba registrado en la base de datos.
    2 -> Si el paciente que necesita el organo esta muerto.
    -1 -> Si hay algun error de tipo SQLException.
*****/
private int EsPosibleExtraerYTratarDatos (OrganRebutxat or)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;

```

```

try{
    //Intentamos Conectar con la base de datos.
    con = DriverManager.getConnection (url, login, passwd);
    stmt=con.createStatement();

    //Se consulta si existe organo en BD.
    int i = ExisteixOrgan(or,stmt);

    //Se consulta si organo ya constaba como rechazado en BD.
    if (i == CORRECTE) i=OrganJaRebutxat(or,stmt);

    //Se consulta si organo esta asignado a un paciente.
    if (i == CORRECTE) i=OrganTrasplantat(or,stmt);

    //Paciente registrado como fallecido.
    if (i == CORRECTE) i=RegistratPacientMort(or,stmt);

    //Se consulta si hospital esta autorizado para solicitar accion.
    if (i==CORRECTE) i = HospitalAutorizado(or,stmt);

    return i;

}catch (SQLException e){
    System.out.println(a.getLocalName()+" : codi d'error "+e.getErrorCode()+
        ".SQLException -> "+e.getMessage());

    return -1;
}finally{ //Este codigo siempre se ejecutara.
    try{ //Cerramos Statement y conexion con BD.
        if (stmt != null) stmt.close();
        if (con != null) con.close();
    }catch (SQLException e){
        System.out.println(a.getLocalName()+" : codi d'error "+
            e.getErrorCode()+
            ".SQLException -> "+e.getMessage().toString());
        return -1;
    }
}
}

/*****
Metodo que actualiza base de datos a partir de los datos introducidos
por el usuario.

Retorna:
    0 ->Si se ha podido introducir datos relativos al rechazo de un
        organo.
    -1 ->Si hay algun error de tipo SQLException.
*****/
private int ExtraerYTratarDatos (OrganRebutxat or)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;

    try{
        //Intentamos Conectar con la base de datos.
        con = DriverManager.getConnection (url, login, passwd);
        stmt=con.createStatement();

        //Modificamos estado 'Operado' a 'Rechazado'
        //en la relacion Necesita.
        ModificarFilaAssignacio(or,stmt);

        return 0;

    }catch (SQLException e){
        System.out.println(a.getLocalName()+" : codi d'error "+
            e.getErrorCode()+
            ".SQLException -> "+e.getMessage());
    }
}

```

```

        return -1;
    }finally{ //Este codigo siempre se ejecutara.
        try{ //Cerramos Statement y conexion con BD.
            if (stmt != null) stmt.close();
            if (con != null) con.close();
        }catch (SQLException e){
            System.out.println(a.getLocalName()+" codi d'error "+
            e.getErrorCode()+
            ".SQLException -> "+e.getMessage().toString());
            return -1;
        }
    }
}

/*****
Metodo que busca en BD si hay registrado un organo igual al que se pasa
por parametro en el objeto OrganRebutxat.

Retorna: 0 -> No hay coincidencia.
         1 -> Hay coincidencia.
*****/
private int ExisteixOrgan (OrganRebutxat or, Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT O.IdOrgan FROM OrgansDonats O"+
        " WHERE O.IdOrgan = '"+
        u.StringConEspaciosEnBlanco(or.get_0())+
        "'");

        if (result.next())
        {
            System.out.println(a.getLocalName()+"Existe Organo en BD.");
            return CORRECTE;
        }

        //No existe organo en BD con este IdOrgan.
        System.out.println(a.getLocalName()+"No existe Organo en BD.");
        return ORGANINEXISTENT;
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que modifica un registro en la tabla Necessita de la BD.
Cambia estado 'Operado'-'>'Rechazado', del paciente vivo que tenia
trasplantado el organo.
*****/
private void ModificarFilaAssignacio (OrganRebutxat or, Statement stmt)
throws SQLException
{
    int i=stmt.executeUpdate ("UPDATE Assignacio SET"+
    " Estat = 'Rechazado'"+
    " WHERE IdOrgan = '"+
    u.StringConEspaciosEnBlanco(or.get_0())+"'+
    " and Estat = 'Operado'");

    System.out.println(a.getLocalName()+
    "Se ha modificado registro en Necessita. i->"+i);
}

/*****
Metodo que busca en BD si un paciente esta registrado con el estado
Muerto, y se le ha trasplantado un organo previamente.
Este metodo sirve para no modificar estado del paciente en caso de
haya fallecido, servira para hacer estadisticas.
Retorna: 0 -> Si paciente esta registrado con el estado vivo.
*****/

```

```

        1 -> Si paciente esta registrado con el estado muerto.
        2 -> Si paciente no esta registrado en BD.
    *****/
    private int RegistratPacientMort(OrganRebutxat or,Statement stmt)
    throws SQLException
    {
        ResultSet result = null;
        try{
            result = stmt.executeQuery ("SELECT P.Estat FROM Pacients P"+
            " WHERE P.CodiSS IN (SELECT A.CodiSS FROM Assignacio A"+
            " WHERE A.IdOrgan = '"+
            u.StringConEspaciosEnBlanco(or.get_0())+"'"
            " and A.Estat = 'Operado')");

            //Miramos si el paciente con el organo trasplantado, esta vivo.
            if (result.next())
            {
                String estat=result.getString(1);

                if (estat.equals("Muerto"))
                {
                    return PACIENTMORT;
                }
                else
                {
                    return CORRECTE;
                }
            }

            System.out.println("El pacient no existeix a BD."+
            " Error greu de consistencia de BD.");
            return PACIENTINEXISTENT;

        }finally{ //Se necesita cerrar el tipo de objeto ResultSet
            if (result != null) result.close();
        }
    }

    /*****/
    Metodo que busca en BD si el organo ha estado trasplantado a un paciente.

    Retorna:
        0 -> Si organo esta trasplantado.
        4 -> Si organo no ha sido trasplantado a nadie.
    *****/
    private int OrganTrasplantat (OrganRebutxat or,Statement stmt)
    throws SQLException
    {
        ResultSet result = null;
        try{
            result = stmt.executeQuery ("SELECT A.CodiSS FROM Assignacio A"+
            " WHERE A.IdOrgan = '"+
            u.StringConEspaciosEnBlanco(or.get_0())+"'"
            " and A.Estat = 'Operado'");

            if (result.next())
            {
                System.out.println("Organo ha sido trasplantado a paciente");
                return CORRECTE;
            }
            else
            {
                System.out.println("Organo no ha sido trasplantado a ningun
paciente");
                return ORGANSENSETRASPLANTAR;
            }
        }finally{ //Se necesita cerrar el tipo de objeto ResultSet
            if (result!=null) result.close();
        }
    }

```



```

    }

/*****
Metodo que busca en BD si el organo ya consta como rechazado.

Retorna:
    0 -> Si organo no consta como rechazado en BD.
    5 -> Si organo ya consta como rechazado por algun paciente en BD.
*****/
private int OrganJaRebutxat (OrganRebutxat or,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT A.CodiSS FROM Assignacio A"+
            " WHERE A.IdOrgan = '"+
            u.StringConEspaciosEnBlanco(or.get_0())+"'" +
            " and A.Estat = 'Rechazado'");

        if (result.next())
        {
            System.out.println("Organo ya consta como rechazado en BD.");
            return ORGANJASHAREBUTXAT;
        }
        else
        {
            System.out.println("Organo no consta que estuviese rechazado.");
            return CORRECTE;
        }
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}

/*****
Metodo que busca en BD si hay registrado un hospital con NomHospital y
Ciutat igual al que se pasa por parametro en el objeto Hospital, y si
el paciente en cuestion esta ingresado alli.

Retorna: 0 -> si existe hospital => Hospital autorizado.
        6 -> no existe hospital => Hospital no autorizado..
*****/
private int HospitalAutorizado (OrganRebutxat or,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    ResultSet result2 = null;
    try{

        Hospital h = or.get_1();
        Adreza adr = h.getAdreza();
        String ciudad = u.StringConEspaciosEnBlanco(adr.getCiutat());

        //Solo habra un paciente al que se le ha trasplantado el organo.
        result =stmt.executeQuery ("SELECT A.CodiSS FROM Assignacio A"+
            " WHERE A.IdOrgan = '"+
            u.StringConEspaciosEnBlanco(or.get_0())+"'" +
            " and A.Estat = 'Operado'");

        //Estamos seguros de que result!=null, porque lo hemos comprobado
        //previamente.
        result.next();

        String cod = u.StringConEspaciosEnBlanco(result.getString(1));

        result2 = stmt.executeQuery ("SELECT H.NomHospital FROM HospitalIngres
H"+
            " WHERE H.NomHospital = '"+u.StringConEspaciosEnBlanco(h.getNom())+"'" +
            " and H.Ciutat = '"+ciudad+"'" +
            " and H.CodiSS = '"+cod+"'" );

```

```

        if (result2.next())
        {
            //Existe paciente ingresado en hospital; por lo tanto, esta autorizado
            //a solicitar accion.
            System.out.println(a.getLocalName()+"Hospital autorizado.");
            return CORRECTE;
        }
        //Paciente no ingresado en ese hospital y no esta autorizado
        //a solicitar accion.
        System.out.println(a.getLocalName()+"Hospital no autorizado.");
        return HOSPITALNOAUTORITZAT;

    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
        if (result2!=null) result2.close();
    }
}
}

```

#### 8.2.4.8 RespuestaAnotacionMuertePaciente.java

```

/*****
PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Clase que implementa el comportamiento que responde en el protocolo
request por parte de Agent0, para que AgentCTH tenga constancia,
de como ha ido la anotacion de la muerte de un paciente.
Ademas de realizar dicha accion.

*****/

package Agent0.agentes.A0;

//Importacion de clases necesarias para implementar un agente.

import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;
import jade.domain.FIPAAgentManagement.*;

import jade.proto.*;
import jade.onto.sl.sl.*;
import java.sql.*;
import java.util.*;

//Librerias necesarias para construir un mensaje.
import jade.lang.acl.*;
import jade.lang.sl.sl.*;

//Importacion de la ontologia.
import Agent0.agentes.A0.Ontology.*;

/*****
Clase que implementa un comportamiento Request:
Petición de AgentCTH para que Agent0 anote en su base de datos la
muerte de un paciente.
*****/
public class RespuestaAnotacionMuertePaciente extends
FipaRequestResponderBehaviour_sl.ActionHandler
implements FipaRequestResponderBehaviour_sl.Factory
{
    UtilidadesAgent0 u = new UtilidadesAgent0();
    AID AID_agent0 = new AID();
    Agent a=null;

    //Datos para conectarse con la base de datos.
    static String login ="db1";
    static String passwd ="A93652ZY";
    static String url="jdbc:odbc:db1";

    private final static int FALLOSQL=-1;
    private final static int CORRECTE=0;
    private final static int PACIENTMORT=1;
    private final static int PACIENTINEXISTENT=2;
    private final static int HOSPITALNOAUTORITZAT=3;

/*****
Constructor del comportamiento que se encarga de anotar la muerte de un
paciente en la BD.
*****/
    public RespuestaAnotacionMuertePaciente (Agent agent, ACLMessage respuesta)
    {
        super(agent,respuesta);
        a=agent;
        AID_agent0=a.getAID();
    }
}

```

```

        System.out.println(a.getLocalName() +
        ":Iniciando Request-Protocol para RESPONDER A anotar muerte de paciente"+
        " con "+ a.getAID());
    }

/*****
Implementacion de la interficie para manejar las acciones recibidas.
*****/
public FipaRequestResponderBehaviour_sl.ActionHandler create
(ACLMessage msg){

    System.out.println(AID_agent0+": Mensaje inicial :"+ msg.toString());

    //Instanciacion a este comportamiento.
    return new RespuestaAnotacionMuertePaciente (Ag0.agente0,msg);
}

/*****
Manejador de mensajes Request: Accion a realizar por este
comportamiento.
*****/
public void action ()
{
    try{

        //Obtenemos la peticion del mensaje recibido.
        ACLMessage msg=getRequest();

        System.out.println(a.getLocalName() +
        ": Mensaje de REQUEST recibido: \n"
        + msg.toString() + "\n");

        //Evaluacion del mensaje y composicion de las posibles respuestas.
        //Accion a realizar por Agent0.
        //A partir de aqui, comenzamos a realizar la accion pedida.
        System.out.println("Procesant IntroducirPacient...");

        //Extraemos contenido de mensaje inicial de Request.
        List l_ag = a.extractContent(msg);
        SL_Action SLa = (SL_Action) l_ag.get(0);

        //Sabemos que es de tipo AnotarMortPacient el objeto recibido.
        AnotarMortPacient amp = (AnotarMortPacient) SLa.get_1();

        //Lo hacemos aqui para escribirlo una sola vez.
        //Creamos un ACLMessage de respuesta.
        ACLMessage reply = msg.createReply();

        //Agent0 consulta BD para comprobar si se puede realizar accion
        //solicitada por AgentCTH.
        switch (EsPosibleExtraerYTratarDatos(amp))
        {
            //Se puede realizar accion.
            case CORRECTE:
                //Enviamos el mensaje: AGREE.
                //Obtenemos la accion.
                l_ag.clear();
                l_ag.add(SLa);
                l_ag.add(new SL_True());

                myAgent.fillContent(reply, l_ag);

                sendReply(ACLMessage.AGREE, reply.getContent());

                if ((ExtraerYTratarDatos(amp))==0)
                {
                    //Generamos un mensaje con accion realizada correctamente
                    //Lo llenamos con contenido de mensaje original
                    SL_Done d = new SL_Done();
                    d.set_0(SLa);

```

```

        l_ag.clear();
        l_ag.add(d);

        reply.setPerformative(ACLMessage.INFORM);
        myAgent.fillContent(reply, l_ag);

        myAgent.send(reply);
    }
    else
    {
        //Indicamos que es de refuse, porque no se puede realizar accion.
        reply.setPerformative(ACLMessage.FAILURE);

        //Creamos contenido del mensaje.

        l_ag.clear();
        l_ag.add(SLa);
        l_ag.add(new ErrorAccesBD());

        //Introducimos contenido en mensaje.
        myAgent.fillContent(reply, l_ag);

        //Enviamos mensaje.
        myAgent.send(reply);
    }
    break;

case PACIENTINEXISTENT:
    //Generamos un mensaje resultado indicando que organo,
    //con identificador introducido,no existe en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientInexistent());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);

    break;

case PACIENTMORT:
    //Generamos un mensaje resultado indicando que organo,
    //con identificador introducido,ya existe en BD.

    //Indicamos que es de refuse, porque no se puede realizar accion.
    reply.setPerformative(ACLMessage.REFUSE);

    //Creamos contenido del mensaje.

    l_ag.clear();
    l_ag.add(SLa);
    l_ag.add(new PacientMort());

    //Introducimos contenido en mensaje.
    myAgent.fillContent(reply, l_ag);

    //Enviamos mensaje.
    myAgent.send(reply);

    break;

```

```

        case HOSPITALNOAUTORIZAT:
            //Generamos un mensaje refuse indicando que el hospital no esta
            //autorizado a solicitar accion.

            //Indicamos que es de refuse, porque no se puede realizar accion.
            reply.setPerformative(ACLMessage.REFUSE);

            //Creamos contenido del mensaje.

            l_ag.clear();
            l_ag.add(SLa);
            l_ag.add(new HospitalSenseAutoritzar());

            //Introducimos contenido en mensaje.
            myAgent.fillContent(reply, l_ag);

            //Enviamos mensaje.
            myAgent.send(reply);

        break;

        case FALLOSQL:
            //Generamos un mensaje REFUSE indicando que se ha producido
            //un error al acceder a BD.

            reply.setPerformative(ACLMessage.REFUSE);

            //Creamos contenido del mensaje.

            l_ag.clear();

            l_ag.add(SLa);
            l_ag.add(new ErrorAccesBD());

            //Introducimos contenido en mensaje.
            myAgent.fillContent(reply, l_ag);

            //Enviamos mensaje.
            myAgent.send(reply);

        break;
    }

} catch (FIPAException fe){
    System.out.println(a.getName() +
        ": Error en el manegador de missatges:");
    System.out.println(fe.getMessage().toString());
}

}

/*****
Manejador de mensajes Request: Metodo done que indica si comportamiento
ha de seguir activo o no.
*****/
public boolean done()
{
    System.out.println(a.getLocalName() +

        ":Comportamiento finalizado correctamente");

    //Ninguna accion a realizar al finalizar el comportamiento.
    return true;
}

/*****
Manejador de mensajes Request: Metodo que reinicia comportamiento.
*****/
public void reset()

```

```

    {
        System.out.println(a.getLocalName() +
            ": Reinicializado el comportamiento");
    }
}

/*****
Metodo que consulta BD para ver si puede realizarse accion pedida
por AgentCTH.

Retorna:
    -1 -> Si se produce cualquier error de tipo SQLException.
    0 -> Si se pueden actualizar datos correctamente.
    1 -> Si paciente esta registrado en BD y esta muerto.
    2 -> Si paciente no esta registrado en BD.
    3 -> Si hospital no esta autorizado para hacer accion.
*****/
private int EsPosibleExtraerYTratarDatos (AnotarMortPacient amp)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;

    try{
        //Intentamos conectar con la base de datos.
        con = DriverManager.getConnection (url, login, passwd);
        stmt=con.createStatement();

        int correcte = 0;

        MortPacient mp = amp.get_0();
        String CodiSS =u.StringConEspaciosEnBlanco(mp.getCodiSS());
        Hospital hosp = amp.get_1();

        //Se consulta estado del paciente en BD.
        correcte=PacientViu(mp,stmt);

        //Se consulta si hospital esta autorizado para solicitar accion.
        if (correcte == 0) correcte = HospitalAutorizado(CodiSS,hosp,stmt);

        return correcte;

    }catch (SQLException e){
        System.out.println(a.getLocalName()+
            ": codi d'error "+e.getErrorCode()+
            ".SQLException -> "+e.getMessage());
        return -1;
    }finally{ //Este codigo siempre se ejecutara.
        try{ //Cerramos Statement y conexion con BD.
            if (stmt != null) stmt.close();
            if (con != null) con.close();
        }catch (SQLException e){
            System.out.println(a.getLocalName()+
                ": codi d'error "+e.getErrorCode()+
                ".SQLException -> "+e.getMessage().toString());
        }
    }
}

/*****
Metodo que actualiza BD segun la accion pedida por el usuario.

Retorna:
    -1 -> Si se produce cualquier error de tipo SQLException.
    0 -> Si todo ha ido bien.
*****/
private int ExtraerYTratarDatos (AnotarMortPacient amp)
{
    //Inicializacion de objeto para poder interactuar con BD.
    Statement stmt = null;
    Connection con = null;

```

```

try{
    //Intentamos Conectar con la base de datos.
    con = DriverManager.getConnection (url, login, passwd);
    stmt=con.createStatement();

    MortPacient mp = amp.get_0();

    int i=stmt.executeUpdate ("UPDATE Pacients SET "+
        "Estat = 'Muerto', "+
        "DiaMort = '"+mp.getDiaMort()+"', "+
        "MesMort = '"+mp.getMesMort()+"', "+
        "AnyMort = '"+mp.getAnyMort()+"', "+
        "HoraMort = '"+mp.getHoraMort()+"', "+
        "MinutsMort = '"+mp.getMinutMort()+"' "+
        "WHERE CodiSS = '"+u.StringConEspaciosEnBlanco(mp.getCodiSS())+"'");

    return 0;
}catch (SQLException e){
    System.out.println(a.getLocalName()+" codi d'error "+
        e.getErrorCode()+
        ".SQLException -> "+e.getMessage());
    return -1;
}finally{ //Este codigo siempre se ejecutara.
    try{ //Cerramos Statement y conexio con BD.
        if (stmt != null) stmt.close();
        if (con != null) con.close();
    }catch (SQLException e){
        System.out.println(a.getLocalName()+" codi d'error "+
            e.getErrorCode()+
            ".SQLException -> "+e.getMessage().toString());
        return -1;
    }
}
}

/*****
Metodo que busca en BD si hay registrado un paciente con CodiSS igual
al que se pasa por parametro en el objeto MortPacient y consta como
paciente vivo.

Retorna: 0 -> si existe paciente en BD en el estado Vivo.
        1 -> si existe paciente en BD en el estado Muerto.
        2 -> si no esta registrado paciente en base de datos.
*****/
private int PacientViu (MortPacient p,Statement stmt) throws SQLException
{
    ResultSet result = null;
    try{
        result = stmt.executeQuery ("SELECT P.CodiSS, P.Estat FROM Pacients P"+
            " WHERE P.CodiSS = '"+u.StringConEspaciosEnBlanco(p.getCodiSS())+"'");

        if (result.next())
        {
            if (result.getString(2).equals("Vivo"))
            {
                System.out.println(a.getLocalName()+"Existe paciente vivo en BD.");
                return CORRECTE;
            }
            else
            {
                System.out.println(a.getLocalName()+"Existe paciente muerto en
BD.");
                return PACIENTMORT;
            }
        }
        //No existe paciente en BD con este CodiSS.
        System.out.println(a.getLocalName()+"No existe paciente en BD.");
        return PACIENTINEXISTENT;
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet

```



```

        if (result!=null) result.close();
    }
}
/*****
Metodo que busca en BD si hay registrado un hospital con NomHospital y
Ciutat igual al que se pasa por parametro en el objeto Hospital, y si
el paciente en cuestion esta ingresado alli.

Retorna: 0 -> si existe hospital => Hospital autorizado.
        3 -> no existe hospital => Hospital no autorizado..
*****/
private int HospitalAutorizado (String CodiSS,Hospital h,Statement stmt)
throws SQLException
{
    ResultSet result = null;
    try{
        Adreza adr = h.getAdreza();
        String ciudad = u.StringConEspaciosEnBlanco(adr.getCiutat());
        result = stmt.executeQuery ("SELECT H.NomHospital FROM HospitalIngres
H"+
        " WHERE H.NomHospital = '"+u.StringConEspaciosEnBlanco(h.getNom())+"'"+
        " and H.Ciutat = '"+ciudad+"'"+
        " and H.CodiSS = '"+CodiSS+"'");

        if (result.next())
        {
            //Existe paciente ingresado en hospital; por lo tanto, esta autorizado
            //a solicitar accion.
            System.out.println(a.getLocalName()+"Hospital autorizado.");
            return CORRECTE;
        }
        //Paciente no ingresado en ese hospital y no esta autorizado
        //a solicitar accion.
        System.out.println(a.getLocalName()+"Hospital no autorizado.");
        return HOSPITALNOAUTORITZAT;
    }finally{ //Se necesita cerrar el tipo de objeto ResultSet
        if (result!=null) result.close();
    }
}
}

```

### 8.2.4.9 UtilidadesAgent0.java

```

/*****
PFC Agent0.
Cristóbal Ledesma Oljas
8/10/02

Clase que implementa metodos utiles en algunos archivos del Ag0.
*****/

package Agent0.agentes.A0;

import java.util.*;

import java.awt.*;
import javax.swing.*;

public class UtilidadesAgent0
{
    /*****
    Constructor de la clase.
    *****/
    public UtilidadesAgent0(){}

    /*****
    Hace falta este metodo, ya que los Strings definidos en la ontologia
    no pueden contener caracteres en blanco, y a este agente les llega
    codificados y este metodo los decodifica.
    *****/
    public String StringConEspaciosEnBlanco (String s)
    {
        String prueba ="
        "+
        "
        ";
        char c[] = s.toCharArray();
        char t[] = prueba.toCharArray();
        String resultado="";

        int AlgunCasAnterior = 0;

        int j=0;

        for (int i=0;i<s.length();i++)
        {
            AlgunCasAnterior = 0;
            //Para recuperar espacios en blanco.

            //Para quitar el caracter en blanco inicial,
            //que introducimos antes para no tener problemas
            //si el usuario introducía un numero en
            //lugar de un String en la interficie grafica. Daria problemas
            //en la ontologia, segun como se haya definido el slot correspondiente.
            if ((c[i]=='/') && (c[i+1]=='_') && (i==0)) i=2;

            if ((c[i]=='/') && (c[i+1]=='_'))
            {
                t[j]=' ';
                j++;

                //Se sustituyen dos caracteres del string inicial por uno del
                //String final, por eso se incrementa la i.
                i++;

                AlgunCasAnterior = 1;
            }
        }
    }
}

```

```

if ((c[i]=='/') && (c[i+1]=='ú'))
{
    t[j]='ú';
    j++;

    //Se sustituyen dos caracteres del string inicial por uno del
    //String final, por eso se incrementa la i.
    i++;

    AlgunCasAnterior = 1;
}

if ((c[i]=='/') && (c[i+1]=='$'))
{
    t[j]='$';
    j++;

    //Se sustituyen dos caracteres del string inicial por uno del
    //String final, por eso se incrementa la i.
    i++;

    AlgunCasAnterior = 1;
}

if ((c[i]=='%') && (c[i+1]=='/'))
{
    t[j]='#';
    j++;

    //Se sustituyen dos caracteres del string inicial por uno del
    //String final, por eso se incrementa la i.
    i++;

    AlgunCasAnterior = 1;
}

if ((c[i]=='&') && (c[i+1]=='/'))
{
    t[j]='€';
    j++;

    //Se sustituyen dos caracteres del string inicial por uno del
    //String final, por eso se incrementa la i.
    i++;

    AlgunCasAnterior = 1;
}

if (AlgunCasAnterior == 0)
{
    t[j]=c[i];
    j+=1;
}
}
for (int h=0;h<j;h++)
{
    resultado+=t[h];
}
return (resultado);
}

/*****
Metodo que codifica los caracteres en blanco recogidos en la interficie
grafica, para poder almacenarlos en la ontologia.
*****/
public String StringSinEspaciosEnBlanco (String s)
{
    String prueba ="
    "+

```

```

        "
";
char c[] = s.toCharArray();
char t[] = prueba.toCharArray();
String resultado="";
int j=0;
int AlgunCasAnterior = 0;
for (int i=0;i<s.length();i++)
{
    AlgunCasAnterior = 0;
    //Para sustituir espacios en blanco.
    if (c[i]==' ')
    { //Substituim caracters en blanc per la cadena '/_'.
        t[j]='/';
        t[j+1]='_';
        j+=2;
        AlgunCasAnterior=1;
    }

    //Para sustituir caracteres ¢, ¥, ¢, y € y no tener
    //problema con Sniffer.
    if (c[i]=='¢')
    {
        t[j]='/';
        t[j+1]='ú';
        j+=2;
        AlgunCasAnterior=1;
    }

    if (c[i]=='¥')
    {
        t[j]='/';
        t[j+1]='$';
        j+=2;
        AlgunCasAnterior=1;
    }

    if (c[i]=='¢')
    {
        t[j]='%';
        t[j+1]='/';
        j+=2;
        AlgunCasAnterior=1;
    }

    if (c[i]=='€')
    {
        t[j]='&';
        t[j+1]='/';
        j+=2;
        AlgunCasAnterior=1;
    }

    if (AlgunCasAnterior==0)
    { //Insertamos caracter no conflictivo.
        t[j]=c[i];
        j+=1;
    }
}
for (int h=0;h<j;h++)
{
    resultado+=t[h];
}
return (resultado);
}

String DeDoubleAFloat (Double d)
{
    Float f;

```

```

f=new Float(d.intValue());
f=new Float(f.floatValue() + ( d.floatValue() % f.floatValue()) );

//Ahora convertimos el Float en cadena de caracteres sustituyendo el punto
//por una coma, para poderlo introducir en BD.
char s[]= f.toString().toCharArray();
String salida = new String();
for (int i=0;i<f.toString().length();i++)
{
    if (s[i]=='.') salida+=", ";
    else salida+=s[i];
}

return salida;
}

Double DeFloatADouble (Float f)
{
    Double d;
    d=new Double (f.intValue());
    d=new Double (d.doubleValue() + ( f.doubleValue() % d.doubleValue()) );
    return d;
}

//Final classe.
}

```

## 8.2.5 Impresión de código contenido en ./Agent0/agentes/A0/Ontology.

En este apartado se debería de volver a incluir todos los archivos del punto 8.2.3, exceptuando OntologiaCTH.java.

### 8.2.5.1 OntologiaAg0.java

```

/*****
    Agent0.agentes.A0.ontology.Ag0Ontology

    PFC Cristobal Ledesma Olias
    23/09/02

    Definición de la ontología: Objetos que representan
    frames y slots para que los agentes CTH y Agent0
    puedan intercambiar datos.
*****/

package Agent0.agentes.A0.Ontology;

import java.io.*;
import jade.onto.*;

import jade.onto.sl.sl.*;
import jade.onto.extensions.*;

//Definición de la ontología: Datos que han de recibir y enviar estos 2
agentes.
public class OntologiaAg0
{
    //nombre de la ontología.
    public static final String NAME="Ontologia_Trasplants_GruSMA";

    //*****Conceptos*****

    //Para datos en mensaje de CTH a Agent0.

    //Objectes definits en la ontología.
    public static final String PACIENT="pacient";
    public static final String DADESCLINIQUES="dadescliniques";
    public static final String HOSPITAL="hospital";
    public static final String ADREZA="adreza";
    public static final String ORGAN="organ";
    public static final String OPERACIO="operacio";
    public static final String
    CARACTERISTIQUESBUSQUEDA="caracteristiquesbusqueda";

    public static final String RESULTAT="resultat";
    public static final String GRAU="grau";

    public static final String ASSOCIACIO="associacio";
    public static final String RESULTATBUSQUEDA="resultatbusqueda";

    public static final String CONSULTAESPECIALISTA="consultaespecialista";
    public static final String DADESPACIENTS="dadespacients";
    public static final String PACIENTS="patients";
    public static final String PESOS="pesos";
    public static final String FUNCIO="funcio";

    public static final String MORTPACIENT="mortpatient";

    //Acciones
    public static final String INTRODUIRPACIENT="introduirpacient";
    public static final String ANOTARMORTPACIENT="anotarmortpatient";
    public static final String
    TROBARPACIENTSCOMPATIBLES="trobarpatientscompatibles";

```

```

public static final String ASSIGNACIOORGAN="assignacioorgan";
public static final String ORGANREBUTXAT="organrebutxat";
public static final String OPERACIOREALITZADA="operaciorealitzada";
public static final String
ACCIOCONSULTAESPECIALISTA="accioconsultaespecialista";

//Errores
public static final String PACIENTINEXISTENT="patientinexistent";
public static final String PACIENTMORT="patientmort";
public static final String PACIENTTEORGANASSIGNAT="patientteorganassignat";
public static final String PACIENTOPERAT="patientoperat";
public static final String ORGANINEXISTENT="organinexistent";
public static final String ORGANEXISTENT="organexistent";
public static final String
PACIENTSENSEORGANASSIGNAT="patientsenseorganassignat";
public static final String PACIENTSENSEOPERAR="patientsenseoperar";
public static final String ERRORACCESBD="erroraccesbd";
public static final String
PACIENTSENSEESPERARORGAN="patientsenseesperarorgan";
public static final String ORGANNODISPONIBLE="organnodisponible";
public static final String ORGANJAESTAVAREBUTXAT="organjaestavarebutxat";

public static final String HOSPITALSENSEAUTORITZAR="hospitalsenseautoritzar";

public static final String AGENTNOREGISTRAT="agentnoregistrat";
public static final String MASSA_OBJECTES="Massa-objectes";
public static final String MASSA_ATRIBUTS="Massa-atributs";
public static final String PESOS_NO_VALIDS="Pesos-no-valids";
public static final String CLASSES_INSUFICIENTS="Classes-insuficients";
public static final String ALGORITME_NO_IMPLEMENTAT="Algoritme-no-
implementat";

//Instancia de la ontologia para referenciar desde el agente
private static Ontology theInstanceAg0 = new SL_DefaultOntologyImp();
public static OntologyHelper ontoAg0_helper = new OntologyHelper();

static{
    initInstance();
}

//Para consultar los conceptos de la ontologia
public static Ontology instanceAg0()
{
    return theInstanceAg0;
}

//Constructor
public OntologiaAg0(){
}

//Inicializa la ontologia
private static void initInstance(){
    try{
        //Objeto patient: contiene todos los datos de un paciente
        theInstanceAg0.joinOntology(SL_Ontology.instance());
        theInstanceAg0.addRole(
            PACIENT,
            new SlotDescriptor[]{
                new
SlotDescriptor("CodiSS", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.
M),
                new
SlotDescriptor("Nom", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.M),
                new
SlotDescriptor("DiaEntradaSistema", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE,
Ontology.M),
                new

```

```

SlotDescriptor("MesEntradaSistema", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE,
Ontology.M),
        new

SlotDescriptor("AnyEntradaSistema", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE,
Ontology.M),
        new

SlotDescriptor("HoraEntradaSistema", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE
, Ontology.M),
        new

SlotDescriptor("MinutEntradaSistema", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYP
E, Ontology.M),
        new

SlotDescriptor("DiaMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology.0
),
        new

SlotDescriptor("MesMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology.0
),
        new

SlotDescriptor("AnyMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology.0
),
        new

SlotDescriptor("HoraMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology.
0),
        new

SlotDescriptor("MinutMort", Ontology.PRIMITIVE_SLOT, Ontology.LONG_TYPE, Ontology
.0),
        new

SlotDescriptor("Estat", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.M
),
        new
        SlotDescriptor("Organo", Ontology.FRAME_SLOT, ORGAN, Ontology.M),
        new

SlotDescriptor("HospitalIngres", Ontology.FRAME_SLOT, HOSPITAL, Ontology.M),
        new

SlotDescriptor("Dades", Ontology.FRAME_SLOT, DADESCLINIQUES, Ontology.M),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new Pacient();
            }
            public Class getClassForRole(){
                return Pacient.class;
            }
        }
    };

    //Objeto Organ: Contiene todos los datos referentes a un Organio.
    theInstanceAg0.addRole(
        ORGAN,
        new SlotDescriptor[]{
            new

SlotDescriptor("IdOrgan", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology
.0),
            new

SlotDescriptor("Tipus", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.M
),

```



```

        new
SlotDescriptor("Tamany",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.
M),
        new
SlotDescriptor("Estat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M
),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new Organ();
            }
            public Class getClassForRole(){
                return Organ.class;
            }
        }
    }
};
//Objeto DadesCliniques: Contiene todos los datos clinicos de un
paciente.
theInstanceAg0.addRole(
    DADESCLINIQUES,
    new SlotDescriptor[]{
        new
SlotDescriptor("DiaNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("MesNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("AnyNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("Sexe",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M)
,
        new
SlotDescriptor("Alzada",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.
M),
        new
SlotDescriptor("Pes",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
        new
SlotDescriptor("TipusSang",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontolo
gy.M),
        new
SlotDescriptor("B0",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("B1",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("Dr0",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("Dr1",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new DadesCliniques();
            }
            public Class getClassForRole(){
                return DadesCliniques.class;
            }
        }
    }
};

```

```

    }
);

//Objeto Hospital: Datos Referentes a un hospital.
theInstanceAg0.addRole(
    HOSPITAL,
    new SlotDescriptor[]{
        new
SlotDescriptor("Nom",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        new
SlotDescriptor("Adreza",Ontology.FRAME_SLOT,ADREZA,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Hospital();
        }
        public Class getClassForRole(){
            return Hospital.class;
        }
    }
);

//Objeto Adreza: Datos Referentes a la direccion de un hospital.
theInstanceAg0.addRole(
    ADREZA,
    new SlotDescriptor[]{
        new
SlotDescriptor("Carrer",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        new
SlotDescriptor("Numero",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        new
SlotDescriptor("Ciutat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
        new
SlotDescriptor("CodiPostal",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontol
ogy.M),
        new
SlotDescriptor("Provincia",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontolo
gy.M),
        new
SlotDescriptor("Pais",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M)
,
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Adreza();
        }
        public Class getClassForRole(){
            return Adreza.class;
        }
    }
);

//Objeto Operacio: Datos referentes a una operaci#n.
theInstanceAg0.addRole(
    OPERACIO,
    new SlotDescriptor[]{
        new
SlotDescriptor("DiaOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontolo
gy.M),

```

```

        new
SlotDescriptor("MesOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("AnyOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("HoraOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("MinutOperacio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("IdOrganTrasplantat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        new
SlotDescriptor("Hospital",Ontology.FRAME_SLOT,HOSPITAL,Ontology.M),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new Operacio();
            }
            public Class getClassForRole(){
                return Operacio.class;
            }
        }
    );

    //Objecto CaracteristiquesBusqueda: Objecto que contiene las
    //caracteristicas de un organo, pera encontrar pacientes compatibles
    //con dicho organo.

    theInstanceAg0.addRole(
        CARACTERISTIQUESBUSQUEDA,
        new SlotDescriptor[]{
            new
SlotDescriptor("IdOrgan",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
            new
SlotDescriptor("Distancia",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
            new
SlotDescriptor("Pes",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
            new
SlotDescriptor("Tamany",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
            new
SlotDescriptor("Antigens",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
            new
SlotDescriptor("Edat",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
            new
SlotDescriptor("TempsEspera",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
            new

```

```

SlotDescriptor("Funcio",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M)
,
    new
SlotDescriptor("Agente",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new CaracteristiquesBusqueda();
        }
        public Class getClassForRole(){
            return CaracteristiquesBusqueda.class;
        }
    }
);

//Estos objetos son los que recibe Ag0 de Especialista como
//resultado de una consulta.

//Objeto Resultat: contiene una lista de pacientes y el
// numero que contiene.
theInstanceAg0.addRole(
    RESULTAT,
    new SlotDescriptor[]{
        new SlotDescriptor("assignacio",Ontology.SET_SLOT,GRAU,Ontology.M),
        new
SlotDescriptor("nobjectes",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Resultat();
        }
        public Class getClassForRole(){
            return Resultat.class;
        }
    }
);

//Objeto Grau (de adecuacion al organo modelo):
//contiene el identificador del paciente y
//el valor que le corresponde despues del proceso de MCDM
theInstanceAg0.addRole(
    GRAU,
    new SlotDescriptor[]{
        new SlotDescriptor("valor-
grau",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        new SlotDescriptor("id-
objecte",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Grau();
        }
        public Class getClassForRole(){
            return Grau.class;
        }
    }
);

//Objeto Resultatdo que se devolver a AgentCTH despues de recibir
//de Agente Especialista el resultado de buscar pacientes compatibles
//con un determinado organo.
//Objeto Resultado de encontrar pacientes compatibles con un organo

theInstanceAg0.addRole(
    RESULTATBUSQUEDA,
    new SlotDescriptor[]{

```

```

        new
SlotDescriptor("Assignacio",Ontology.SET_SLOT,ASSOCIACIO,Ontology.M),
        new
SlotDescriptor("NPacients",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new ResultatBusqueda();
        }

        public Class getClassForRole(){
            return ResultatBusqueda.class;
        }
    }
);

//Objeto que indica un paciente y su grado de compatibilidad con
//un determinado organo.
theInstanceAg0.addRole(
    ASSOCIACIO,
    new SlotDescriptor[]{
        new
SlotDescriptor("Grau",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M)
,
        new
SlotDescriptor("CodiSS",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.
M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new Associacio();
        }
        public Class getClassForRole(){
            return Associacio.class;
        }
    }
);

//Objecto para pedir servicio a Agente Especialista

//Objeto DadesPacients
theInstanceAg0.addRole(
    DADESPACIENTS,
    new SlotDescriptor[]{
        new
SlotDescriptor("nPacients",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.M),
        new
SlotDescriptor("LlistaPacients",Ontology.SET_SLOT,PACIENTS,Ontology.M),
        new
SlotDescriptor("TipusOrgan",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontol
ogy.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new DadesPacients();
        }
        public Class getClassForRole(){
            return DadesPacients.class;
        }
    }
);

//Objeto Pacients

```

```

        theInstanceAg0.addRole(
            PACIENTS,
            new SlotDescriptor[]{
                new
SlotDescriptor("NomHospital",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Onto
logy.M),
                new
SlotDescriptor("Localitat",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontolo
gy.M),
                new
SlotDescriptor("Provincia",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontolo
gy.M),
                new
SlotDescriptor("TipusSang",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontolo
gy.M),
                new
SlotDescriptor("PesPersona",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontol
ogy.M),
                new
SlotDescriptor("TamanyOrgan",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Onto
logy.M),
                new
SlotDescriptor("B0",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
                new
SlotDescriptor("B1",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
                new
SlotDescriptor("Dr0",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
                new
SlotDescriptor("Dr1",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
                new
SlotDescriptor("DiaNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
                new
SlotDescriptor("MesNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
                new
SlotDescriptor("AnyNaixement",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontol
ogy.M),
                new
SlotDescriptor("DiaEspera",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.O),
                new
SlotDescriptor("MesEspera",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.O),
                new
SlotDescriptor("AnyEspera",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology
.O),
            },
            new RoleEntityFactory(){
                public Object create (Frame f){
                    return new Pacients();
                }
                public Class getClassForRole(){
                    return Pacients.class;
                }
            }
        );

        //Objeto Pesos
        theInstanceAg0.addRole(
            PESOS,
            new SlotDescriptor[]{
                new
SlotDescriptor("Pes",Ontology.PRIMITIVE_SLOT,Ontology.DOUBLE_TYPE,Ontology.M),
                new
SlotDescriptor("Atribut",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology
.M),
            },
            new RoleEntityFactory(){
                public Object create (Frame f){

```

```

        return new Pesos();
    }
    public Class getClassForRole(){
        return Pesos.class;
    }
}
);

//Objecte CONSULTAESPECIALISTA:
//conte les dades a passar al agent ESPECIALISTA:
theInstanceAg0.addRole(
    CONSULTAESPECIALISTA,
    new SlotDescriptor[]{
new
SlotDescriptor("Funcio_dist",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        new
SlotDescriptor("Pacientes",Ontology.FRAME_SLOT,DAESPACIENTS,Ontology.M),
        new
SlotDescriptor("LlistaPesos",Ontology.SET_SLOT,PESOS,Ontology.M),
        new
SlotDescriptor("Agente",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new ConsultaEspecialista();
        }
        public Class getClassForRole(){
            return ConsultaEspecialista.class;
        }
    }
);

//Objecto MortPacient: Indica paciente y momento de su muerte.
theInstanceAg0.addRole(
    MORTPACIENT,
    new SlotDescriptor[]{
        //NUEVO SLOT: CodiSS.
        new
SlotDescriptor("CodiSS",Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
        //NUEVO SLOT: DiaMort
        new
SlotDescriptor("DiaMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        //NUEVO SLOT: MesMort
        new
SlotDescriptor("MesMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        //NUEVO SLOT: AnyMort
        new
SlotDescriptor("AnyMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        //NUEVO SLOT: HoraMort
        new
SlotDescriptor("HoraMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
        //NUEVO SLOT: MinutMort
        new
SlotDescriptor("MinutMort",Ontology.PRIMITIVE_SLOT,Ontology.LONG_TYPE,Ontology.M),
    },
);

```

```

    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new MortPacient();
        }
        public Class getClassForRole(){
            return MortPacient.class;
        }
    }
);

//Objeto que indica un error porque paciente no existe.
theInstanceAg0.addRole(
    PACIENTINEXISTENT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientInexistent();
        }
        public Class getClassForRole(){
            return PacientInexistent.class;
        }
    }
);

//Objeto que indica un error porque organo ya existe.
theInstanceAg0.addRole(
    ORGANEXISTENT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganExistent();
        }
        public Class getClassForRole(){
            return OrganExistent.class;
        }
    }
);

//Objeto que indica un error porque paciente est muerto.
theInstanceAg0.addRole(
    PACIENTMORT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientMort();
        }
        public Class getClassForRole(){
            return PacientMort.class;
        }
    }
);

//Objeto que indica error porque paciente tiene organo assignado.
theInstanceAg0.addRole(
    PACIENTTEORGANASSIGNAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientTeOrganAssignat();
        }
        public Class getClassForRole(){
            return PacientTeOrganAssignat.class;
        }
    }
);

```



```

    }
);
//Objeto que indica error porque paciente no espera organo.
theInstanceAg0.addRole(
    PACIENTSENSEESPERARORGAN,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientSenseEsperarOrgan();
        }
        public Class getClassForRole(){
            return PacientSenseEsperarOrgan.class;
        }
    }
);

//Objeto que indica error porque paciente ya estaba operado.
theInstanceAg0.addRole(
    PACIENTOPERAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientOperat();
        }
        public Class getClassForRole(){
            return PacientOperat.class;
        }
    }
);

//Objeto que indica error porque no existe el organo.
theInstanceAg0.addRole(
    ORGANINEXISTENT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganInexistent();
        }
        public Class getClassForRole(){
            return OrganInexistent.class;
        }
    }
);

//Objeto que indica error porque organo esta asignado.
theInstanceAg0.addRole(
    ORGANNODISPONIBLE,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganNoDisponible();
        }
        public Class getClassForRole(){
            return OrganNoDisponible.class;
        }
    }
);

//Objeto que indica error porque paciente no tiene organo asignado.
theInstanceAg0.addRole(
    PACIENTSENSEORGANASSIGNAT,
    new SlotDescriptor[]{
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientSenseOrganAssignat();
        }
    }
);

```

```

        }
        public Class getClassForRole(){
            return PacientSenseOrganAssignat.class;
        }
    }
};

//Objeto que indica error porque paciente est sin operar.
theInstanceAg0.addRole(
    PACIENTSENSEOPERAR,
    new SlotDescriptor[]{
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PacientSenseOperar();
        }
        public Class getClassForRole(){
            return PacientSenseOperar.class;
        }
    }
);

//Objeto que indica error porque organo ya consta como rechazado.
theInstanceAg0.addRole(
    ORGANJAESTAVAREBUTXAT,
    new SlotDescriptor[]{
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new OrganJaEstavaRebutxat();
        }
        public Class getClassForRole(){
            return OrganJaEstavaRebutxat.class;
        }
    }
);

//Objeto que indica error porque hospital no esta autorizado a solicitar
//accion.
theInstanceAg0.addRole(
    HOSPITALSENSEAUTORITZAR,
    new SlotDescriptor[]{
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new HospitalSenseAutoritzar();
        }
        public Class getClassForRole(){
            return HospitalSenseAutoritzar.class;
        }
    }
);

//CLUSDM: Objeto que indica un error por pesos no validos (no suman 100)
theInstanceAg0.addRole(
    PESOS_NO_VALIDS,
    new SlotDescriptor[]{
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new PesosNoValids();
        }
        public Class getClassForRole(){
            return PesosNoValids.class;
        }
    }
);

//CLUSDM: Objeto que indica un error por numero de pacientes excesivo
theInstanceAg0.addRole(

```

```

        MASSA_OBJECTES,
        new SlotDescriptor[]{
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new MassaObjectes();
            }
            public Class getClassForRole(){
                return MassaObjectes.class;
            }
        }
    );

    //CLUSDM: Objeto que indica un error per numero de atributos excesivo
    theInstanceAg0.addRole(
        MASSA_ATRIBUTS,
        new SlotDescriptor[]{
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new MassaAtributs();
            }
            public Class getClassForRole(){
                return MassaAtributs.class;
            }
        }
    );

    //CLUSDM: Objeto que indica un error: no se han obtenido suficientes
    clases
    theInstanceAg0.addRole(
        CLASSES_INSUFICIENTS,
        new SlotDescriptor[]{
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new ClassesInsuficients();
            }
            public Class getClassForRole(){
                return ClassesInsuficients.class;
            }
        }
    );

    //CLUSDM: Objeto que indica un error: no se ha implementado algoritmo
    theInstanceAg0.addRole(
        ALGORITME_NO_IMPLEMENTAT,
        new SlotDescriptor[]{
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new AlgoritmeNoImplementat();
            }
            public Class getClassForRole(){
                return AlgoritmeNoImplementat.class;
            }
        }
    );

    //CLUSDM: Objeto que indica un error: no se ha implementado algoritmo
    theInstanceAg0.addRole(
        AGENTNOREGISTRAT,
        new SlotDescriptor[]{
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new AgentNoRegistrat();
            }
            public Class getClassForRole(){

```

```

        return AgentNoRegistrat.class;
    }
}
);

//Objeto que indica error en acceso a BD.
theInstanceAg0.addRole(
    ERRORACCESBD,
    new SlotDescriptor[]{
        },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new ErrorAccesBD();
        }
        public Class getClassForRole(){
            return ErrorAccesBD.class;
        }
    }
);

//Accio Introducir Paciente: Objeto que contiene informaci n
//que pasara AgentCTH a Agent0 para realizar la accion
//introducir paciente
theInstanceAg0.addRole(
    INTRODUIRPACIENT,
    new SlotDescriptor[]{
        //NUEVO SLOT: Pacient.
        new
        SlotDescriptor(Ontology.FRAME_SLOT,PACIENT,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new IntroduirPacient();
        }
        public Class getClassForRole(){
            return IntroduirPacient.class;
        }
    }
);

//Objeto de la ontologia que contiene datos, que enviar AgentCTH a
//Agent0, para que se realice la accion Anotar Muerte del Paciente.

theInstanceAg0.addRole(
    ANOTARMORTPACIENT,
    new SlotDescriptor[]{
        //NUEVO SLOT: MortPacient.
        new
        SlotDescriptor(Ontology.FRAME_SLOT,MORTPACIENT,Ontology.M),
        new
        SlotDescriptor(Ontology.FRAME_SLOT,HOSPITAL,Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new AnotarMortPacient();
        }
        public Class getClassForRole(){
            return AnotarMortPacient.class;
        }
    }
);

//Objeto que enviar AgentCTH a Agent0, para que este realice la
//accion de encontrar pacientes compatibles con un organo.

theInstanceAg0.addRole(
    TROBARPACIENTSCOMPATIBLES,
    new SlotDescriptor[]{
        //NUEVO SLOT: CaracteristiquesBusqueda.
        new

```

```

SlotDescriptor(Ontology.FRAME_SLOT, CARACTERISTIQUESBUSQUEDA, Ontology.M),
    new
    SlotDescriptor(Ontology.FRAME_SLOT, HOSPITAL, Ontology.M),
},
new RoleEntityFactory(){
    public Object create (Frame f){
        return new TrobarPacientsCompatibles();
    }
    public Class getClassForRole(){
        return TrobarPacientsCompatibles.class;
    }
}
);

//Accio Consultar Especialista: Objeto que contiene informaci n
//que pasara Agent0 a Agent Especialista para realizar la accion
//encontrar pacientes compatibles.
theInstanceAg0.addRole(
    ACCIOCONSULTAESPECIALISTA,
    new SlotDescriptor[]{
        //NUEVO SLOT: Patient.
        new
        SlotDescriptor(Ontology.FRAME_SLOT, CONSULTAESPECIALISTA, Ontology.M),
    },
    new RoleEntityFactory(){
        public Object create (Frame f){
            return new AccioConsultaEspecialista();
        }
        public Class getClassForRole(){
            return AccioConsultaEspecialista.class;
        }
    }
);

//Objeto AsignacioOrgan: Indica paciente y organo que se le ha
asignado.
//Cuando AgentCTH envíe este objeto a Agent0, Agent0 realizar la
//asignacion del organo en Base de Datos.

theInstanceAg0.addRole(
    ASIGNACIOORGAN,
    new SlotDescriptor[]{
        //NUEVO SLOT: CodiSS
        new
        SlotDescriptor(Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.M),
        //NUEVO SLOT: IdOrgan
        new
        SlotDescriptor(Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE, Ontology.M),
    },
    new
    SlotDescriptor(Ontology.FRAME_SLOT, HOSPITAL, Ontology.M),
},
new RoleEntityFactory(){
    public Object create (Frame f){
        return new AsignacioOrgan();
    }
    public Class getClassForRole(){
        return AsignacioOrgan.class;
    }
}
);

//Objeto OrganRebutxat: Indica organo que ha sido rechazado.
//Cuando AgentCTH envíe este objeto a Agent0, Agent0 anotara en
//en su base de datos que el organo a sido rechazado por paciente.
theInstanceAg0.addRole(

```

```

        ORGANREBUTXAT,
        new SlotDescriptor[]{
            //NUEVO SLOT: IdOrgan
            new
SlotDescriptor(Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,Ontology.M),
            new
SlotDescriptor(Ontology.FRAME_SLOT,HOSPITAL,Ontology.M),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new OrganRebutxat();
            }
            public Class getClassForRole(){
                return OrganRebutxat.class;
            }
        }
    );

    //Objeto que enviara AgentCTH a Agent0, indicandole que se ha realizado
    //un trasplante de un organo.
    theInstanceAg0.addRole(
        OPERACIOREALITZADA,
        new SlotDescriptor[]{
            //NUEVO SLOT: Operacio
            new
SlotDescriptor(Ontology.FRAME_SLOT,OPERACIO,Ontology.M),
        },
        new RoleEntityFactory(){
            public Object create (Frame f){
                return new OperacioRealitzada();
            }
            public Class getClassForRole(){
                return OperacioRealitzada.class;
            }
        }
    );

}catch (OntologyException e){
    System.err.println("Error generant la Ontologia");
    e.printStackTrace();
}
}
}

```

### 8.2.5.2 AccioConsultaEspecialista.

/\*\*\*\*\*  
CTHAg0Ontology.AccioConsultaEspecialista

PFC Cristóbal Ledesma Oljas  
25/09/02

Implementación del objeto AccioConsultaEspecialista definido en la ontología  
\*\*\*\*\*/

```
package Agent0.agentes.A0.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

public class AccioConsultaEspecialista extends Domain_OntologyObj
{
    //Variables
    private ConsultaEspecialista consulta;

    //Constructor
    public AccioConsultaEspecialista()
    {
        super("AccioConsultaEspecialista");
        consulta=new ConsultaEspecialista();
    }
    public AccioConsultaEspecialista(ConsultaEspecialista p)
    {
        super("AccioConsultaEspecialista");
        consulta=p;
    }

    //Metodos SET
    public void set_0(ConsultaEspecialista p)
    {
        this.consulta=p;
    }

    //Metodos GET
    public ConsultaEspecialista get_0()
    {
        return this.consulta;
    }
}
```

### 8.2.5.3 ConsultaEspecialista.java

```
/*
*****
ontology.ConsultaEspecialista

PFC
Cristobal Ledesma Olias
09/12/2002

Implementacion del objeto: ConsultaEspecialista definido en la ontologia
*****
*****/

package Agent0.agentes.A0.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

public class ConsultaEspecialista extends Domain_OntologyObj
{
    //Variables
    private List LlistaPesos=new ArrayList();
    private DadesPacients Pacientes;
    private Long Funcio_dist;
    private String Agente;

    //Construtor
    public ConsultaEspecialista()
    {
        super("ConsultaEspecialista");
        Funcio_dist=new Long(0);
        Pacientes=new DadesPacients();
        LlistaPesos=new ArrayList();
        Agente="no_inicializado";
    }

    //Metodos SET
    public void addLlistaPesos (Pesos p)
    {
        (this.LlistaPesos).add(p);
    }
    public void clearLlistaPesos()
    {
        this.LlistaPesos=new ArrayList();
    }
    public void setPacientes (DadesPacients p)
    {
        this.Pacientes=p;
    }
    public void setFuncio_dist (Long f)
    {
        this.Funcio_dist=f;
    }
    public void setAgente (String f)
    {
        this.Agente=f;
    }

    //Metodos GET
    public Iterator getAllLlistaPesos()
    {
        return(this.LlistaPesos).iterator();
    }
    public DadesPacients getPacientes()
    {
        return this.Pacientes;
    }
}
```



```
    public Long getFuncio_dist()
    {
        return this.Funcio_dist;
    }

    public String getAgente()
    {
        return this.Agente;
    }
}
```

#### 8.2.5.4 DadesPacients.java

```
/*
*****
ontology.DadesPacients

PFC
Cristobal Ledesma Olias
09/12/2002

Implementacion del objeto: DadesPacients definido en la ontologia
*****
*****/

package Agent0.agentes.A0.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

//Implementacio de l'objecte: DadesPacients definit a l'ontologia
public class DadesPacients extends Domain_OntologyObj
{
    //Variables
    private List LlistaPacients;
    private Long nPacients;
    private String TipusOrgan;

    //Construtor
    public DadesPacients()
    {
        super("DadesPacients");
        nPacients=new Long(0);
        TipusOrgan="no_inicializado";
        LlistaPacients=new ArrayList();
    }

    public DadesPacients(Long n, String s)
    {
        super("DadesPacients");
        nPacients=n;
        TipusOrgan=s;
        LlistaPacients=new ArrayList();
    }

    //Metodos SET
    public void addLlistaPacients (Pacients p)
    {
        (this.LlistaPacients).add(p);
    }
    public void clearLlistaPacients()
    {
        this.LlistaPacients=new ArrayList();
    }
    public void setnPacients (Long n)
    {
        this.nPacients=n;
    }

    public void setTipusOrgan (String n)
    {
        this.TipusOrgan=n;
    }

    //Metodos GET
    public Iterator getAllLlistaPacients()
    {
        return(this.LlistaPacients).iterator();
    }
}
```

```
        public Long getnPacients()
        {
            return this.nPacients;
        }

        public String getTipusOrgan()
        {
            return this.TipusOrgan;
        }
    }
```

### 8.2.5.5 Grau.java

```

/*****
*****
ontology.Grau

PFC
David Sanchez Ruenes
28-7-2001

Implementacio de l'objecte: Grau definit a l'ontologia
*****/

package Agent0.agentes.A0.Ontology;

import jade.onto.sl.sl.*;

//Implementacio de l'objecte: Grau definit a l'ontologia
public class Grau extends Domain_OntologyObj
{
    //Variables
    private String valor_grau=new String("");
    private Long id_objecte;

    //Constructor
    public Grau()
    {
        super("Grau");
    }

    //Metodos SET
    public void setvalorGrau (String s)
    {
        this.valor_grau=s;
    }

    public void setidObjecte (Long id)
    {
        this.id_objecte=id;
    }

    //Metodes GET
    public String getvalorGrau()
    {
        return this.valor_grau;
    }

    public Long getidObjecte()
    {
        return this.id_objecte;
    }
}

```

### 8.2.5.6 *OrdenacioImpossible.java*

```

/*****
CTHAg0Ontology.OrdenacioImpossible

PFC Cristóbal Ledesma Oljas
25/09/02

Implementación del objeto OrdenacioImpossible definido en la ontología
*****/

package Agent0.agentes.A0.Ontology;

import jade.onto.sl.sl.*;

public class OrdenacioImpossible extends Domain_OntologyObj
{
    //Constructor
    public OrdenacioImpossible()
    {
        super("OrdenacioImpossible");
    }
}

```

### 8.2.5.7 Pacients.java

```

/*****
*****
ontology.Pacients

PFC
David Sanchez Ruenes
28-7-2001

Implementacio de l'objecte Pacients definit a l'ontologia
*****/

package Agent0.agentes.A0.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

//Implementacio de l'objecte Funcio definit a l'ontologia
public class Pacients extends Domain_OntologyObj
{

    //Variables
    private String NomHospital;
    private String Localitat;
    private String Provincia;
    private String TipusSang;
    private Double PesPersona;
    private Double TamanyOrgan;
    private Long B0;
    private Long B1;
    private Long Dr0;
    private Long Dr1;
    private Long DiaNaixement;
    private Long MesNaixement;
    private Long AnyNaixement;
    private Long DiaEspera;
    private Long MesEspera;

    private Long AnyEspera;

    //Constructor
    public Pacients()
    {
        super("Pacients");
        this.NomHospital="no_inicializado";
        this.Localitat="no_inicializado";
        this.Provincia="no_inicializado";
        this.TipusSang="no_inicializado";
        this.PesPersona=new Double(0);
        this.TamanyOrgan=new Double(0);
        this.B0=new Long(0);
        this.B1=new Long(0);
        this.Dr0=new Long(0);
        this.Dr1=new Long(0);
        this.DiaNaixement=new Long(0);
        this.MesNaixement=new Long(0);
        this.AnyNaixement=new Long(0);
        this.DiaEspera=null;
        this.MesEspera=null;
        this.AnyEspera=null;
    }

    public Pacients(String n,String l, String p, String t,
    Double pp, Double to,Long b0, Long b1,Long dr0, Long dr1,
    Long dn, Long mn, Long an, Long de, Long me, Long ae)

```

```

{
    super("Pacients");
    this.NomHospital=n;
    this.Localitat=l;
    this.Provincia=p;
    this.TipusSang=t;
    this.PesPersona=pp;
    this.TamanyOrgan=to;
    this.B0=b0;
    this.B1=b1;
    this.Dr0=dr0;
    this.Dr1=dr1;
    this.DiaNaixement=dn;
    this.MesNaixement=mn;
    this.AnyNaixement=an;
    this.DiaEspera=de;
    this.MesEspera=me;
    this.AnyEspera=ae;
}

public Pacients(String n,String l, String p, String t,
Double pp, Double to,Long b0, Long b1,Long dr0, Long dr1,
Long dn, Long mn, Long an)
{
    super("Pacients");
    this.NomHospital=n;
    this.Localitat=l;
    this.Provincia=p;
    this.TipusSang=t;
    this.PesPersona=pp;
    this.TamanyOrgan=to;
    this.B0=b0;
    this.B1=b1;
    this.Dr0=dr0;
    this.Dr1=dr1;
    this.DiaNaixement=dn;
    this.MesNaixement=mn;
    this.AnyNaixement=an;
    this.DiaEspera=null;
    this.MesEspera=null;
    this.AnyEspera=null;
}

//Metodes SET
public void setNomHospital (String n)
{
    this.NomHospital=n;
}

public void setLocalitat (String n)
{
    this.Localitat=n;
}
public void setProvincia (String n)
{
    this.Provincia=n;
}

public void setTipusSang (String n)
{
    this.TipusSang=n;
}

public void setPesPersona (Double n)
{
    this.PesPersona=n;
}

```

```

    public void setTamanyOrgan (Double n)
    {
        this.TamanyOrgan=n;
    }

    public void setB0 (Long n)
    {
        this.B0=n;
    }

    public void setB1 (Long n)
    {
        this.B1=n;
    }

    public void setDr0 (Long n)
    {
        this.Dr0=n;
    }

    public void setDr1 (Long n)
    {
        this.Dr1=n;
    }

    public void setDiaNaixement (Long n)
    {
        this.DiaNaixement=n;
    }

    public void setMesNaixement (Long n)
    {
        this.MesNaixement=n;
    }

    public void setAnyNaixement (Long n)
    {
        this.AnyNaixement=n;
    }

    public void setDiaEspera (Long n)
    {
        this.DiaEspera=n;
    }

    public void setMesEspera (Long n)
    {
        this.MesEspera=n;
    }

    public void setAnyEspera (Long n)
    {
        this.AnyEspera=n;
    }

    //Metodes GET
    public String getNomHospital ()
    {
        return this.NomHospital;
    }

    public String getLocalitat ()
    {
        return this.Localitat;
    }
    public String getProvincia ()
    {
        return this.Provincia;
    }

```



```

    public String getTipusSang ()
    {
        return this.TipusSang;
    }

    public Double getPesPersona ()
    {
        return this.PesPersona;
    }

    public Double getTamanyOrgan ()
    {
        return this.TamanyOrgan;
    }

    public Long getB0 ()
    {
        return this.B0;
    }

    public Long getB1 ()
    {
        return this.B1;
    }

    public Long getDr0 ()
    {
        return this.Dr0;
    }

    public Long getDr1 ()
    {
        return this.Dr1;
    }

    public Long getDiaNaixement ()
    {
        return this.DiaNaixement;
    }

    public Long getMesNaixement ()
    {
        return this.MesNaixement;
    }

    public Long getAnyNaixement ()
    {
        return this.AnyNaixement;
    }

    public Long getDiaEspera ()
    {
        return this.DiaEspera;
    }

    public Long getMesEspera ()
    {
        return this.MesEspera;
    }

    public Long getAnyEspera ()
    {
        return this.AnyEspera;
    }
}

```

### 8.2.5.8 Pesos.java

```
/******  
ontology.Pesos  
  
PFC  
David Sanchez Ruenes  
28-7-2001  
  
Implementacio de l'objecte Pesos definit a l'ontologia  
*****/  
  
package Agent0.agentes.A0.Ontology;  
  
import jade.onto.sl.sl.*;  
import java.util.*;  
  
//Implementacio de l'objecte Pesos definit a l'ontologia  
public class Pesos extends Domain_OntologyObj  
{  
  
    //Variables  
    private Double Pes;  
    private String Atribut;  
  
    //Constructor  
    public Pesos()  
    {  
        super("Pesos");  
        this.Pes=new Double(0);  
        this.Atribut="no_inicializado";  
    }  
  
    public Pesos(Double l,String a)  
    {  
        super("Pesos");  
        this.Pes=l;  
        this.Atribut=a;  
    }  
  
    //Metodes SET  
    public void setPes (Double l)  
    {  
        this.Pes=l;  
    }  
  
    public void setAtribut (String a)  
    {  
        this.Atribut=a;  
    }  
  
    //Metodes GET  
    public Double getPes()  
    {  
        return(this.Pes);  
    }  
    public String getAtribut()  
    {  
        return(this.Atribut);  
    }  
}
```

### 8.2.5.9 Resultat.java

```

/*****
*****
ontology.Resultat

PFC
David Sanchez Ruenes
28-7-2001

Classe necessaria per Agent Ag0.

Implementacio de l'objecte: Resultat definit a l'ontologia
*****
*****/

package Agent0.agentes.A0.Ontology;

import jade.onto.sl.sl.*;
import java.util.*;

//Implementacio de l'objecte: Resultat definit a l'ontologia
public class Resultat extends Domain_OntologyObj
{
    //Variables
    private List assignacio=new ArrayList();
    private Long nobjectes;

    //Construtor
    public Resultat()
    {
        super("Resultat");
    }

    //Metodos SET
    public void addassignacio (Object grau)
    {
        (this.assignacio).add(grau);
    }
    public void clearassignacio ()
    {
        assignacio=new ArrayList();
    }
    public void setnobjectes (Long n)
    {
        this.nobjectes=n;
    }

    //Metodos GET
    public Iterator getAllassignacio()
    {
        return(this.assignacio).iterator();
    }
    public Long getnobjectes()
    {
        return this.nobjectes;
    }
}

```



## **9. ¿Cómo se pone en marcha el sistema?**

Para ejecutar el sistema multi-agente que se ha presentado en esta documentación, se han de seguir los siguientes pasos:

-Instalar el software **jdk-1\_2\_2\_013**, que corresponde al lenguaje de programación java, porque es el lenguaje utilizado para programar los agentes que componen este sistema y Jade. (Tiene que ser la versión **\_013** o posterior, para que pueda funcionar sobre una plataforma Intel Pentium IV. Teniendo en cuenta que ha de utilizarse jdk-1.2).

-Instalar el software **Jade**, que proporciona la plataforma básica sobre la que se ejecutarán los agentes, y herramientas para desarrollarlos.

-Actualizar las variables de entorno **CLASSPATH** y **PATH**. Esto sólo es válido para una plataforma Windows. Para el ordenador en el que se ha implementado este proyecto, las variables de entorno se han actualizado con el archivo entorno.bat, que tiene el siguiente contenido:

```
set PATH=c:\WINNT\SYSTEM32;E:\WINNT;c:\WINNT\SYSTEM32\WBEM;c:\ARCHIVOS DE
PROGRAMA\ARCHIVOS COMUNES\ADAPTEC SHARED\SYSTEM;c:\JDK1.2.2\BIN;;
set
CLASSPATH=c:\.;c:\JADE\LIB\JADE.JAR;c:\JADE\LIB\JADETOOLS.JAR;c:\JADE\LIB\BASE64.JAR;c:\JADE\
CLASSES\JADE_SL.JAR;c:\pfc_ei\ontology;
```

-Configurar el **controlador ODBC** de windows, para poder acceder a la base de datos desde Ag0. Se han utilizado los siguientes valores para la configuración:

En la pestaña DSN de usuario y de sistema, ha de aparecer el nombre db1 con el controlador Driver do Microsoft Access. (\*.mdb).

Nombre de origen de datos: db1.

Descripción: Datos de Agent0.

Base de datos: .\Agent0\A0\BDAgent0\db1.mdb.

Nombre de inicio de sesión: db1.

Contraseña: A93652ZY.

-Para compilar todo el sistema multi-agente se ha utilizado el archivo **compila.bat**, que tiene el siguiente contenido:

```
cd agentes
cd CTH
del *.class
javac -encoding 850 *.java
cd interficie
del *.class
javac -encoding 850 *.java
cd ..
cd ontology
del *.class
javac -encoding 850 *.java
cd ..
cd ..
cd A0
del *.class
javac -encoding 850 *.java
cd ontology
del *.class
javac -encoding 850 *.java
```

```
cd ..
cd ..
cd especial
del *.class
javac -encoding 850 *.java
cd ontology
del *.class
javac -encoding 850 *.java
cd ..
cd ..
cd ..
```

-Para ejecutar jade y cargar los agentes se han utilizado los archivos **executa.bat** y **executa2.bat**, que tienen el siguiente contenido:

executa.bat:

```
java jade.Boot -platform -gui h:Agent0.agentes.CTH.AgentCTH o:Agent0.agentes.A0.Ag0
e:Agent0.agentes.especial.EspecialistaAgent c:Agent0.agentes.especial.ClusDMResponderAgent ;
```

executa2.bat:

```
java jade.Boot -platform -gui h:Agent0.agentes.CTH.AgentCTH o:Agent0.agentes.A0.Ag0
e:Agent0.agentes.especial.EspecialistaAgent c:Agent0.agentes.especial.ClusDMResponderAgent
f:Agent0.agentes.especial.DummyFuzzyAgent t:Agent0.agentes.especial.Dummy2TupleAgent
ord:Agent0.agentes.especial.DummyOrdinalAgent ;
```

## **10. Linea de trabajos futuros.**

Como se ha dicho al iniciar esta documentación, este proyecto forma parte de uno más general, que involucra a varios sistemas multi-agente. Este sistema multi-agente general, se puede ir ampliando con relativa facilidad, ya sea incorporando nuevos agentes ó proporcionando más funcionalidad a los existentes. La tendencia de futuros trabajos en este campo, puede ser imprevisible.

Las tareas que se deberían realizar en un futuro próximo, son las siguientes:

- Unificar todos los sistemas multi-agentes que se han implementado hasta el momento, en uno sólo. El concepto de sistema multi-agente es relativamente nuevo, y está evolucionando muy deprisa. Los sistemas multi-agentes comentados en la introducción, utilizan distintas versiones de software, y esto complica su conexión para que funcione cómo una unidad.
- Implementar un sistema multi-agente que accediera a la información del sistema general, y que facilitara datos estadísticos que permitieran tener una visión más amplia de cómo se está gestionando todo lo relacionado con el trasplante de órganos.
- Debido a las características del sistema multi-agente general, se debería probar que el sistema funcionase de una forma distribuida, conectando varias plataformas de agentes (Agent Platform). Otra opción, es ejecutar el sistema sobre una única plataforma de agentes que estuviera compartida por varios ordenadores, y en donde estuviesen cargados todos los agentes.
- Probar el sistema general adaptándolo a la realidad.

## **11. Conclusiones**

El concepto de sistema multi-agente es relativamente nuevo. Gracias a organismos sin ánimo de lucro como la FIPA, que ha ido proporcionando estándares, se ha facilitado la evolución y la implementación de los sistemas multi-agentes.

Un sistema multi-agente en general, proporciona una opción bastante interesante en el diseño de aplicaciones distribuidas, que se puede aplicar a cualquier disciplina. Pero sobre todo, es una gran aportación que se ha hecho a la inteligencia artificial, por la capacidad que tienen los agentes de compartir conocimiento, y por la capacidad de aprendizaje y toma de decisiones, que pueden adquirir para funcionar con un alto grado de independencia.

En el aspecto personal, este proyecto me ha permitido aprender unos conceptos informáticos nuevos para mí. Pienso que en un mundo tan apasionante y cambiante como es la informática, tener una mentalidad abierta a conocer nuevas tecnologías es importante.

Académicamente hablando, me ha permitido introducirme en el campo de los agentes y sistemas multi-agente. También me ha permitido conocer la herramienta de desarrollo de agentes JADE, profundizar en la programación orientada a objetos (Concretamente en java), y poner en práctica los conocimientos relacionados con el diseño de una base de datos y su gestión desde java.

Quiero agradecer el asesoramiento que me han proporcionado Aïda Valls y Toni Moreno, para poder implementar el proyecto. Mi agradecimiento a David Isern y a Ivan López, que me hicieron una introducción a los agentes y sistemas multi-agentes, y las tendencias que hay en este campo. También agradezco a los compañeros del grupo GruSMA en general por su aportación de proyectos.



## **Bibliografia**

### **[Banzai]**

Banzai, Grup de recerca en Intel·ligència Artificial, DEIM, Universitat Rovira i Virgili, URL: <http://www.etse.urv.es/recerca/banzai>.

### **[Bocio, 00]**

Jaime Bocio, Diseño e Implementación de un Sistema Multi-Agente para la gestión de equipos médicos, Proyecto Final de Carrera de Ingeniería Técnica en Informática de Sistemas, URV, 2000,

URL: <http://www.etse.urv.es/recerca/banzai/toni/MAS/>

### **[David Sánchez, 2001]**

Un Sistema Multi-Agent d'ajuda a l'assignació d'òrgans en trasplantaments. El mètode ClusDM amb criteris lingüístics. Proyecto Final de Carrera de Ingeniería Técnica en Informática de Sistemas, URV, 2001,

URL: <http://www.etse.urv.es/recerca/banzai/toni/MAS/>

### **[FIPA]**

Foundation for Intelligent Physical Agents, web page: <http://www.fipa.org>

### **[FIPA, 1998a]**

Foundation for Intelligent Physical Agents (1998), Agent Communication Language, FIPA

### **[FIPA, 1998b]**

Foundation for Intelligent Physical Agents (1998), Agent Management, FIPA

### **[FIPA, 1998c]**

Foundation for Intelligent Agents (1998), FIPA97 Developers Guide, FIPA

### **[FIPA, 2000a]**

Foundation for Intelligent Agents (2000), FIPA ACL Message Structure, FIPA

### **[FIPA, 2000b]**

Foundation for Intelligent Agents (2000), FIPA Agent Management Specification, FIPA

### **[FIPA, 2000c]**

Foundation for Intelligent Agents (2000), FIPA SL Content Language Specification, FIPA

### **[FIPA 2001]**

Foundation for Intelligent Physical Agents (2001), FIPA Device Ontology, FIPA

### **[GruSMA]**

GruSMA, Grup de Sistemes Multi-Agent, DEIM, Universitat Rovira i Virgili, plana web: <http://www.etse.urv.es/recerca/banzai/toni/MAS>

### **[GruSMA, 2000a]**

GruSMA (2000), Agents i Sistemes Multi-Agents, Universitat Rovira i Virgili

**[GruSMA, 2000b]**

GruSMA (2000), Especificación FIPA de la Gestión de Agentes, Universitat Rovira i Virgili

**[GruSMA, 2000c]**

GruSMA (2000), Jade 1.3, Universitat Rovira i Virgili

**[GruSMA, 2000d]**

GruSMA (2000), Lenguaje de comunicación entre agentes FIPA, Universitat Rovira i Virgili

**[Isern, 2001]**

Isern, D. (2001), JADE: Conceptes fonamentals, apunts GruSMA, Universitat Rovira i Virgili

**[Isern,2000]**

Isern, D. (2000), Desenvolopament de Sistemes Multiagent: JADE, apunts GruSMA, Universitat Rovira i Virgili

**[Isern, 1999]**

Isern, D. (1999), Avaluació d'entorns de desenvolopament de sistemes Multiagent, Projecte Final de Carrera, Enginyeria Técnica en Informática de Sistemas, Universitat Rovira i Virgili

**[Isern et al., 2001]**

Isern, D., Moreno, A., Valls, A. (2001), Desenvolopament de sistemes Multi-Agent en JADE, Report de Recerca DEIM-RR-01-003, Dept. Enginyeria Informática i Matemàtiques, Universitat Rovira i Virgili

**[ONT]**

Organización Nacional de Trasplantes, Ministerio de Sanida y Consumo, <http://www.msc.es/ont/esp/home.htm>

**[Ribes, 00]**

Albert Ribes, SMA d'ajuda a la gestió dels trasplants d'òrgans entre hospitals. Logística del transport dels òrgans. Proyecto Final de Carrera de Ingeniería Técnica en Informática de Sistemas, URV,2000,

URL: <http://www.etse.urv.es/recerca/banzai/toni/MAS/>