# Técnica de Búsqueda para la Prestación de Servicios sobre Redes Superpuestas P2P No Estructuradas

# Documento de Anexos

**Alejandro Muñoz Andrade**
**Diego Eryk Muñoz Luna**

*Universidad del Cauca*
**Facultad de Ingeniería Electrónica y Telecomunicaciones**
**Programa en Ingeniería de Sistemas**
**Departamento de Sistemas**
**Grupo IDIS - Investigación y Desarrollo en Ingeniería de Software**
**Línea de Investigación Ingeniería de la Colaboración e Ingeniería del Software Basada en la Colaboración**
Popayán, Noviembre de 2010

# Técnica de Búsqueda para la Prestación de Servicios sobre Redes Superpuestas P2P No Estructuradas

# Documento de Anexos


**Alejandro Muñoz Andrade**
**Diego Eryk Muñoz Luna**

Monografía de Trabajo de Grado

Director
Ing. Pablo Augusto Magé Imbachí

*Universidad del Cauca*
**Facultad de Ingeniería Electrónica y Telecomunicaciones**
**Programa en Ingeniería de Sistemas**
**Departamento de Sistemas**
**Grupo IDIS - Investigación y Desarrollo en Ingeniería de Software**
**Línea de Investigación Ingeniería de la Colaboración e Ingeniería del Software Basada en la Colaboración**
Popayán, Noviembre de 2010

## TABLA DE CONTENIDO

**ANEXO A: ALCANCE DEL PROYECTO ..... 9**
1. Objetivos del Proyecto ..... 9
1.1. Objetivo General ..... 9
1.2. Objetivos Específicos ..... 9
2. Justificación del Proyecto ..... 9
3. Metodología de Trabajo ..... 9
4. Resumen de Entregables ..... 11
4.1. Relacionados con la Gestión del Trabajo de Grado ..... 11
4.2. Relacionados con el Prototipo Software ..... 11
5. Criterios de Éxito del Proyecto ..... 11

**ANEXO B: DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS SOFTWARE - SRS ..... 12**
1. Introducción ..... 12
1.1. Propósito ..... 12
1.2. Alcance ..... 12
1.3. Definiciones, Siglas y Abreviaciones ..... 12
2. Descripción Global ..... 12
2.1. Perspectiva del Producto ..... 12
2.2. Funciones del Producto ..... 14
2.3. Características del Usuario ..... 14
2.4. Atenciones y Dependencias ..... 14
2.5. Priorizar los Requerimientos ..... 14
3. Requerimientos Específicos ..... 15
3.1. Funcionales ..... 15
3.2. No Funcionales ..... 17
4. Rendimiento ..... 18
5. Restricciones de Diseño ..... 18

**ANEXO C: DOCUMENTO DE CASOS DE USO ..... 19**
1. Definición de Actores ..... 19
2. Casos de Uso en Formato de Alto Nivel ..... 19

2.1. Diagrama de Casos de Uso en Formato de Alto Nivel ..... 19
2.2. Descripción de Casos de Uso en Formato de Alto Nivel ..... 22
3. Casos de Uso en Formato Extendido ..... 26
3.1. Diagrama de Casos de Uso en Formato Extendido ..... 26
3.2. Descripción de Casos de Uso en Formato Extendido ..... 28

**ANEXO D: TECNOLOGÍAS PARA EL DESARROLLO DE REDES SUPERPUESTAS P2P Y SISTEMAS MULTI-AGENTE ..... 35**
1. JXTA ..... 35
1.1. ¿Qué es JXTA? ..... 35
1.2. ¿Por qué JXTA? ..... 36
1.3. Arquitectura JXTA ..... 37
1.4. Protocolos JXTA ..... 38
1.5. Servicios JXTA ..... 40
2. JADE ..... 41
2.1. ¿Qué es JADE? ..... 41
2.2. ¿Por qué JADE? ..... 42
2.3. Arquitectura JADE ..... 43
2.4. Comportamientos en JADE ..... 44
2.5. Servicios en JADE ..... 48

**ANEXO E: ESPECIFICACIÓN DE LOS EQUIPOS UTILIZADOS EN LAS PRUEBAS ..... 51**

**ANEXO F: MANUAL TÉCNICO Y DE INSTALACIÓN ..... 56**
1. Requerimientos Software de la Aplicación ..... 56
1.1. Instalación de Java ..... 56
1.2. Desactivación del Firewall ..... 56
2. Instalación de la Aplicación ..... 56
3. Ejecución de la Aplicación ..... 57
3.1. Cambiar el puerto de JADE ..... 57
3.2. Cambiar el puerto de JXTA ..... 57

**ANEXO G: MANUAL DE USUARIO ..... 58**
1. Ejecutar la Aplicación ..... 58
2. Menú Servidor ..... 59
2.1. Activar ..... 59
2.2. Desactivar ..... 59
2.3. Eliminar Mensajes ..... 59
2.4. Pestaña Servidor ..... 60
3. Menú Cliente ..... 60
3.1. Activar ..... 60
3.2. Desactivar ..... 60
3.3. Eliminar Mensajes Enviados ..... 61
3.4. Eliminar Mensajes Recibidos ..... 61
3.5. Eliminar todos los mensajes ..... 61
3.6. Pestaña Cliente ..... 61
4. Pestaña Información ..... 61
5. Salir de la Aplicación ..... 62

**REFERENCIAS BIBLIOGRÁFICAS ..... 64**

## LISTA DE FIGURAS

Figura 1. Diagrama de Casos de Uso - Paquetes ..... 20
Figura 2. Diagrama de Casos de Uso – Paquete Servicio ..... 21
Figura 3. Diagrama de Casos de Uso – Paquete Mensajes ..... 21
Figura 4. Diagrama de Casos de Uso – Paquete Opciones ..... 22
Figura 5. Diagrama de Casos de Uso Extendido – Paquete Servicio ..... 27
Figura 6. Diagrama de Casos de Uso Extendido – Paquete Mensajes ..... 27
Figura 7. Diagrama de Casos de Uso Extendido – Paquete Opciones ..... 28
Figura 8. Arquitectura JXTA ..... 37
Figura 9. Protocolos JXTA ..... 40
Figura 10. Arquitectura JADE ..... 43
Figura 11. Relación entre los elementos de la arquitectura principal de JADE ..... 44
Figura 12. Ruta de ejecución de un agente ..... 46
Figura 13. Servicio de Páginas Amarillas ..... 49
Figura 14. Aplicación del Servicio de Eco ..... 58
Figura 15. Servidor de la Aplicación de Eco ..... 59
Figura 16. Cliente de la Aplicación de Eco ..... 60
Figura 17. Información de la Aplicación de Eco ..... 62
Figura 18. Opción Salir de la Aplicación ..... 63
Figura 19. Pantalla de Salida de la Aplicación ..... 63

# LISTA DE TABLAS

Tabla 1. Requerimiento - Publicar el Servicio de Eco ..... 15
Tabla 2. Requerimiento - Buscar el Servicio de Eco ..... 15
Tabla 3. Requerimiento - Recuperar el Servicio de Eco..... 15
Tabla 4. Requerimiento - Enviar el Mensaje de Eco ..... 16
Tabla 5. Requerimiento - Transmitir el Mensaje de Eco..... 16
Tabla 6. Requerimiento - Recibir el Mensaje de Eco ..... 16
Tabla 7. Requerimiento - Iniciar el Servidor de Eco ..... 16
Tabla 8. Requerimiento - Detener el Servidor de Eco ..... 17
Tabla 9. Requerimiento - Iniciar el Cliente de Eco ..... 17
Tabla 10. Requerimiento - Detener el Cliente de Eco ..... 17
Tabla 11. Actor Sistema ..... 19
Tabla 12. Actor Usuario..... 19
Tabla 13. Caso de Uso en Formato de Alto Nivel - Publicar Servicio..... 22
Tabla 14. Caso de Uso en Formato de Alto Nivel – Buscar Servicio..... 23
Tabla 15. Caso de Uso en Formato de Alto Nivel – Recuperar Servicio ..... 23
Tabla 16. Caso de Uso en Formato de Alto Nivel – Enviar Mensaje..... 24
Tabla 17. Caso de Uso en Formato de Alto Nivel – Transmitir Mensaje ..... 24
Tabla 18. Caso de Uso en Formato de Alto Nivel – Recibir Mensaje..... 24
Tabla 19. Caso de Uso en Formato de Alto Nivel – Iniciar Servidor..... 25
Tabla 20. Caso de Uso en Formato de Alto Nivel – Detener Servidor ..... 25
Tabla 21. Caso de Uso en Formato de Alto Nivel – Iniciar Cliente..... 25
Tabla 22. Caso de Uso en Formato de Alto Nivel – Detener Cliente..... 26
Tabla 23. Caso de Uso en Formato Extendido – Publicar Servicio ..... 29
Tabla 24. Caso de Uso en Formato Extendido – Buscar Servicio..... 29
Tabla 25. Caso de Uso en Formato Extendido – Recuperar Servicio ..... 30
Tabla 26. Caso de Uso en Formato Extendido – Enviar Mensaje ..... 31
Tabla 27. Caso de Uso en Formato Extendido – Transmitir Mensaje ..... 31
Tabla 28. Caso de Uso en Formato Extendido – Recibir Mensaje ..... 32
Tabla 29. Caso de Uso en Formato Extendido – Iniciar Servidor..... 32
Tabla 30. Caso de Uso en Formato Extendido – Detener Servidor..... 33
Tabla 31. Caso de Uso en Formato Extendido – Iniciar Cliente ..... 34
Tabla 32. Caso de Uso en Formato Extendido – Detener Cliente..... 34
Tabla 33. Equipo PC-010 ..... 51
Tabla 34. Equipo PC-013 ..... 51
Tabla 35. Equipo PC-017 ..... 51
Tabla 36. Equipo PC-018 ..... 52
Tabla 37. Equipo PC-020 ..... 52
Tabla 38. Equipo PC-021 ..... 52
Tabla 39. Equipo PC-022 ..... 53
Tabla 40. Equipo PC-023 ..... 53
Tabla 41. Equipo PC-024 ..... 53
Tabla 42. Equipo PC-025 ..... 54
Tabla 43. Equipo PC-026 ..... 54

Tabla 44. Equipo PC-027 ..... 54
Tabla 45. Equipo PC-028 ..... 55
Tabla 46. Equipo PC-029 ..... 55

# ANEXO A: ALCANCE DEL PROYECTO

## 1. Objetivos del Proyecto

### 1.1. Objetivo General

- Proponer una Técnica de Búsqueda para la prestación de servicios sobre redes superpuestas P2P no estructuradas.

### 1.2. Objetivos Específicos

- Determinar y priorizar las principales características de búsqueda de servicios sobre redes P2P no estructuradas.
- Construir o adecuar una técnica de búsqueda para redes superpuestas P2P no estructuradas, que combine las características de búsqueda existentes.
- Desarrollar un prototipo software que permita verificar la técnica propuesta, dentro de una intranet, sobre un servicio específico.

## 2. Justificación del Proyecto

Debido a la gran acogida que han tenido las aplicaciones P2P en todo el mundo, y al poco conocimiento del funcionamiento real de estas redes, este proyecto trata de realizar una contribución significante a nivel investigativo en el área de las redes P2P, con la especificación de una Técnica de Búsqueda para la prestación de Servicios sobre Redes Superpuestas P2P no estructuradas. Entre los aportes se encuentran:

- Plantear una técnica de búsqueda que permita mejorar la prestación de servicios sobre estas redes.
- Establecer las principales características de búsqueda en este tipo de redes.
- Realizar una búsqueda eficiente de los nodos que prestan determinados servicios en entornos de Redes P2P no estructuradas.

Teniendo como principio que los servicios son considerados el centro de la gran mayoría de actividades hoy en día en cuanto a nivel de redes se refiere, se pretende con esta técnica de búsqueda ayudar en la forma en que estos son ofrecidos y a su vez explorar las funcionalidades prestadas por las plataformas de desarrollo para aplicaciones P2P.

## 3. Metodología de Trabajo

Teniendo en cuenta que el proyecto consta de tres partes esenciales que son: la investigación científica acerca de las técnicas que actualmente se usan para la

búsqueda en la prestación de servicios sobre redes P2P no estructuradas, la construcción o adecuación de la técnica y posteriormente el desarrollo de un prototipo software el cual permita verificar la técnica, ha surgido la necesidad de aplicar la metodología de investigación apoyada sobre los métodos de investigación de ingeniería [1], donde se definen las siguientes características:

Para la elaboración de la base conceptual y del estado del arte se recurrirá a la consulta e investigación documental a través de artículos, tesis, consultas a expertos, y otros medios de información relacionados con el tema de prestación de servicios sobre redes P2P, redes P2P no estructuradas, búsqueda sobre redes P2P no estructuradas, arquitecturas y sistemas de prestación de servicios sobre redes P2P no estructuradas y otros elementos que permitan mejorar el entendimiento del problema que se quiere resolver. Por último se realizará un documento de alcance con el fin de resolver los objetivos que se quieren cumplir en el proyecto de investigación.

Se realizará el estudio y análisis del funcionamiento de las redes P2P no estructuradas y la manera cómo se realiza la búsqueda para la prestación de servicios sobre estas redes. En esta fase se construirá o adecuará la técnica a partir de la investigación anteriormente realizada.

Posteriormente se realizará el diseño y la implementación de un prototipo software, que permita verificar el desempeño de la técnica anteriormente planteada en esta investigación. Para la implementación del prototipo software se usará la metodología de desarrollo Agile Unified Process (AUP) [2], utilizando sus fases y solo los artefactos que se consideren necesarios.

Las fases que se usarán son las de Inicio, Elaboración, Construcción y Transición, sin embargo, las disciplinas que se emplearán sólo comprenden el Diseño, Implementación y Pruebas. Así las fases de Elaboración y Construcción serán iterativas, usando el modelo Análisis, Diseño, Implementación y Pruebas [3].

Se llevarán a cabo un conjunto de actividades orientadas a la documentación del desarrollo del proyecto de investigación, conclusiones y resultados obtenidos a través de la elaboración de una monografía y se escribirá un artículo relacionado con el tema que será expuesto en un evento nacional o internacional o publicado en la página web de la Universidad del Cauca o en la página web personal de cualquiera de los integrantes de este trabajo de grado. Por último se realizará el proceso de sustentación del trabajo de grado ante los jurados definidos por la F.I.E.T. y se procederá a entregar el respectivo producto.

## 4. Resumen de Entregables

### 4.1. Relacionados con la Gestión del Trabajo de Grado

- Monografía del Trabajo de Grado
- Artículo de Investigación
- Documento de evaluación de los resultados obtenidos

### 4.2. Relacionados con el Prototipo Software

- Documento de Especificación de Requerimientos Software – SRS
- Lista de requerimientos funcionales y no funcionales
- Definición de la arquitectura del sistema
- Casos de uso en formato de alto nivel
- Casos de uso en formato extendido
- Diagrama de clases
- Diagramas de secuencia
- Manual de instalación
- Manual de usuario
- Prototipo software final que cumpla con los requerimientos establecidos

## 5. Criterios de Éxito del Proyecto

El proyecto será exitoso si como resultado genera:

- Los entregables relacionados con la gestión del trabajo de grado
- Los entregables relacionados con el prototipo software

# ANEXO B: DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS SOFTWARE - SRS

## 1. Introducción

### 1.1. Propósito

En este documento se pretende clasificar y describir los requerimientos que han sido identificados para el desarrollo del prototipo software que verificará la técnica de búsqueda propuesta.

### 1.2. Alcance

El prototipo software, que será desarrollado, pretende verificar la técnica de búsqueda propuesta sobre el servicio de envió y recepción de mensajes de eco. Por lo tanto, la mayoría de la funcionalidad del prototipo software se ejecutará de manera automática, sin ninguna interacción con el usuario final del sistema. El prototipo software permitirá ejecutar la funcionalidad tanto de cliente (solicitante del servicio) como de servidor (prestador del servicio) en una sola interfaz de usuario.

El prototipo software integrará dos tecnologías para el desarrollo de redes P2P y sistemas multi-agente, la primera, JXTA, para el desarrollo de Redes P2P; y la segunda, JADE, para el desarrollo de Sistemas Multi-Agente.

### 1.3. Definiciones, Siglas y Abreviaciones

- **Servicio de Mensajes de Eco:** Un servicio de mensajes de eco consiste en el envío y propagación de un mensaje hacia todos los nodos que estén conectados actualmente al nodo que presta el servicio en la red P2P.
- **GUI:** Interfaz Grafica de Usuario.
- **JDK:** Java Development Kit.
- **JRE:** Java Runtime Environment.

## 2. Descripción Global

### 2.1. Perspectiva del Producto

#### 2.1.1. Interfaces con el sistema

El prototipo software deberá integrarse con JADE y JXTA. Por lo tanto, deberá crearse un modulo que sirva de interfaz de comunicación entre el prototipo software y estas dos tecnologías de desarrollo.

### 2.1.2. Interfaces con el usuario

- El usuario interactuará con el prototipo software a través de una aplicación de escritorio.
- El prototipo software tendrá la funcionalidad de cliente, de servidor y de mensajes de depuración en la misma interfaz gráfica de usuario.
- La Interfaz gráfica de usuario deberá ser simple e intuitiva, permitiendo al usuario un fácil manejo de la aplicación.
- Los mensajes de información, depuración y mensajes de error serán mostrados en el panel de mensajes de información.
- Los mensajes de error graves serán desplegados en cajas de diálogos.

### 2.1.3. Interfaces con el hardware

El prototipo software se podrá ejecutar en computadores personales o de escritorio, que tengan una conexión a Internet.

### 2.1.4. Interfaces con el software

El prototipo software estará implementado en java, al igual que las tecnologías de desarrollo JADE y JXTA. Por la tanto, el computador donde se ejecute el prototipo deberá tener instalada una Maquina Virtual Java. Se recomienda la versión 1.6.0 del JDK o del JRE de Sun Microsystem.

### 2.1.5. Interfaces de comunicaciones

El prototipo software será soportado por los protocolos de comunicaciones JXTA y su respectiva arquitectura.

### 2.1.6. Restricciones de memoria

- Se necesitarán como mínimo 512 MB de memoria RAM y un procesador con una velocidad de 1.6 GHz para ejecutar la aplicación.
- Se recomienda una memoria RAM de 2GB o más y un procesador con una velocidad de 2.0 GHz o más para un rendimiento óptimo de la aplicación.

### 2.1.7. Requerimientos de adaptación

El prototipo software no tendrá en cuenta aspectos de seguridad, tales como nombre de usuarios o contraseñas, o sistemas de autenticación. Si se llegará a solicitar algún nombre de usuario, solo se hará con fines informativos.

## 2.2. Funciones del Producto

El prototipo software permitirá el envió y recepción de mensajes de eco. La propagación del mensaje se realizará mediante una conexión uno-a-uno (unicast), no por medio de la típica conexión uno-a-muchos (multicast). El envió de mensajes de eco se realizará cada 6 segundos, para tener un total de 10 mensajes por minuto. El contenido del mensaje de eco será la hora actual del sistema en el que se está ejecutando el cliente.

Además, el prototipo permitirá ejecutar la funcionalidad de cliente y servidor en una sola interfaz de usuario y mostrará mensajes de información y depuración del proceso que se está llevando a cabo.

## 2.3. Características del Usuario

La mayoría de la funcionalidad del prototipo software se ejecutará de manera automática, sin ninguna interacción con el usuario final del sistema. La única interacción del usuario final con el sistema, será la selección manual del tipo de funcionamiento del sistema, es decir el sistema puede comportarse como cliente, como servidor o como ambos. Sin embargo, el usuario podrá ver el envió y recepción de mensajes de eco, tanto del cliente como del servidor (si está activada la respectiva funcionalidad), y los mensajes de depuración del sistema.

## 2.4. Atenciones y Dependencias

Existen algunos factores que pueden afectar el desarrollo del prototipo, entre ellos se mencionan:

- Poco conocimiento de las tecnologías que se van a utilizar en el desarrollo del prototipo.
- Actividades no programadas, como por ejemplo capacitación y manejo de las tecnologías de desarrollo utilizadas.
- Cambios en las prioridades del equipo de trabajo.
- Resistencia a los cambios planteados.
- Fechas límites de entrega.

## 2.5. Priorizar los Requerimientos

- **Primera Iteración:** Desarrollar la funcionalidad del sistema multi-agente del prototipo software, de la técnica de búsqueda y el servicio de eco, mediante la plataforma de agentes JADE.

- **Segunda Iteración:** Desarrollar la funcionalidad del sistema y de las comunicaciones en la Red P2P del prototipo software, de la técnica de

búsqueda y el servicio de eco, mediante la plataforma JXTA. Desarrollar la interfaz de comunicaciones entre las dos plataformas, JADE y JXTA, y el prototipo software.

## 3. Requerimientos Específicos

### 3.1. Funcionales

| **REQUERIMIENTO: PUBLICAR EL SERVICIO DE ECO** | |
|---|---|
| **Código** | R-1 |
| **Nombre** | Publicar el Servicio de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [X] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe crear y publicar el servicio de eco. |
| **Prerrequisito** | Ninguno |
| **Manejo de Errores** | E1: El servicio de eco no se puede publicar. El sistema muestra un mensaje de error. |

**Tabla 1. Requerimiento - Publicar el Servicio de Eco**

| **REQUERIMIENTO: BUSCAR EL SERVICIO DE ECO** | |
|---|---|
| **Código** | R-2 |
| **Nombre** | Buscar el Servicio de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [X] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe buscar el servicio de eco. |
| **Prerrequisito** | Ninguno |
| **Manejo de Errores** | E1: El servicio de eco no se encuentra. El sistema muestra un mensaje de información y sigue buscando el servicio indefinidamente. |

**Tabla 2. Requerimiento - Buscar el Servicio de Eco**

| **REQUERIMIENTO: RECUPERAR EL SERVICIO DE ECO** | |
|---|---|
| **Código** | R-3 |
| **Nombre** | Recuperar el Servicio de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [X] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe recuperar el servicio de eco. El sistema hace uso del cache para recuperar el estado del envío de mensajes. |
| **Prerrequisito** | No se encuentra el nodo que prestaba el servicio de eco. |
| **Manejo de Errores** | E1: El servicio de eco no se encuentra. El sistema muestra un mensaje de información y sigue buscando el servicio indefinidamente, hasta poder recuperarlo. El sistema almacena los mensajes que no puede enviar en el cache. |

**Tabla 3. Requerimiento - Recuperar el Servicio de Eco**

| **REQUERIMIENTO: ENVIAR EL MENSAJE DE ECO** | |
|---|---|
| **Código** | R-4 |
| **Nombre** | Enviar el Mensaje de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [X] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe enviar un mensaje de eco. Esta funcionalidad pertenece |

| | |
|---|---|
| | a un nodo cliente. |
| **Prerrequisito** | El sistema debe estar conectado a un nodo que preste el servicio de eco. |
| **Manejo de Errores** | E1: El mensaje de eco no se puede enviar. El sistema muestra un mensaje de error.<br>E2: El nodo que presta el servicio de eco ya no está en la red. El sistema muestra un mensaje de error y busca indefinidamente hasta encontrar un nuevo nodo que preste el servicio. |

**Tabla 4. Requerimiento - Enviar el Mensaje de Eco**

| **REQUERIMIENTO: TRANSMITIR EL MENSAJE DE ECO** | |
|---|---|
| **Código** | R-5 |
| **Nombre** | Transmitir el Mensaje de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [X] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe transmitir un mensaje de eco. Esta funcionalidad pertenece a un nodo servidor. |
| **Prerrequisito** | El sistema debe recibir un mensaje de eco a ser transmitido. |
| **Manejo de Errores** | E1: El mensaje de eco no se puede enviar. El sistema muestra un mensaje de error. |

**Tabla 5. Requerimiento - Transmitir el Mensaje de Eco**

| **REQUERIMIENTO: RECIBIR EL MENSAJE DE ECO** | |
|---|---|
| **Código** | R-6 |
| **Nombre** | Recibir el Mensaje de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [X] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe recibir un mensaje de eco, que fue transmitido por un nodo servidor. Esta funcionalidad pertenece a un nodo cliente. |
| **Prerrequisito** | El sistema debe estar conectado a un nodo que preste el servicio de eco. |
| **Manejo de Errores** | E1: El mensaje de eco no se recibe correctamente, por lo tanto no se puede mostrar. El sistema muestra un mensaje de error. |

**Tabla 6. Requerimiento - Recibir el Mensaje de Eco**

| **REQUERIMIENTO: INICIAR EL SERVIDOR DE ECO** | |
|---|---|
| **Código** | R-7 |
| **Nombre** | Iniciar el Servidor de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [ ] Alta/Esencial [X] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe permitir al usuario iniciar el servidor de eco. |
| **Prerrequisito** | El servidor de eco debe estar detenido. |
| **Manejo de Errores** | E1: El servidor de eco no se puede iniciar. El sistema muestra un mensaje de error. |

**Tabla 7. Requerimiento - Iniciar el Servidor de Eco**

| **REQUERIMIENTO: DETENER EL SERVIDOR DE ECO** | |
|---|---|
| **Código** | R-8 |
| **Nombre** | Detener el Servidor de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [ ] Alta/Esencial [X] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe permitir al usuario detener el servidor de eco. |
| **Prerrequisito** | El servidor de eco debe estar funcionando. |

| | |
|---|---|
| **Manejo de Errores** | E1: El servidor de eco no se puede detener. El sistema muestra un mensaje de error. |

**Tabla 8. Requerimiento - Detener el Servidor de Eco**

| **REQUERIMIENTO: INICIAR EL CLIENTE DE ECO** | |
|---|---|
| **Código** | R-9 |
| **Nombre** | Iniciar el Cliente de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [ ] Alta/Esencial [X] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe permitir al usuario iniciar el cliente de eco. |
| **Prerrequisito** | El cliente de eco debe estar detenido. |
| **Manejo de Errores** | E1: El cliente de eco no se puede iniciar. El sistema muestra un mensaje de error. |

**Tabla 9. Requerimiento - Iniciar el Cliente de Eco**

| **REQUERIMIENTO: DETENER EL CLIENTE DE ECO** | |
|---|---|
| **Código** | R-10 |
| **Nombre** | Detener el Cliente de Eco |
| **Fuente** | Información recolectada en la fase de investigación. |
| **Prioridad** | [ ] Alta/Esencial [X] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | El sistema debe permitir al usuario detener el cliente de eco. |
| **Prerrequisito** | El cliente de eco debe estar funcionando. |
| **Manejo de Errores** | E1: El cliente de eco no se puede detener. El sistema muestra un mensaje de error. |

**Tabla 10. Requerimiento - Detener el Cliente de Eco**

## 3.2. No Funcionales

- **Fiabilidad:** El prototipo software debe ser tolerante a fallos. Debe poder recuperarse de los diferentes errores que puedan presentarse. Es importante decir que los fallos por componentes hardware pueden estar presentes y no se tienen en cuenta en el desarrollo de este prototipo, así como también es importante decir que los fallos de software siempre están latentes en una aplicación y algunas veces pueden ser inesperados.

- **Disponibilidad:** Una de las características de las redes P2P es que los nodos pueden entrar y salir de esta arbitrariamente, por lo tanto el grado de disponibilidad de la aplicación depende de la disponibilidad del servicio en la red P2P.

- **Seguridad:** El prototipo software no tendrá en cuenta aspectos de seguridad, tales como nombre de usuarios o contraseñas, o sistemas de autenticación. Si se llegará a solicitar algún nombre de usuario o contraseñas, solo se hará con fines informativos.

- **Consistencia:** La consistencia es un punto importante en las aplicaciones P2P, sin ella, el sistema no funcionará de una manera adecuada. Por lo tanto,

la aplicación debe garantizar consistencia en el envío y recepción de mensajes, y en actualización de interfaz de usuario.

- **Mantenimiento:** El desarrollo del prototipo software no incluye mantenimiento. Sin embargo, si hay una mejora en las actualizaciones de las herramientas de desarrollo o las plataformas utilizadas, se incluirán esas nuevas actualizaciones, siempre y cuando el prototipo no se vea afectado de una manera negativa.

- **Portabilidad:** El prototipo software se podrá ejecutar en computadores personales o de escritorio, que tengan una conexión a Internet. El prototipo software deberá poder ejecutarse en cualquier sistema operativo que tenga instalada una Maquina Virtual Java. Se recomienda la versión 1.6.0 del JDK o del JRE de Sun Microsystem.

## 4. Rendimiento

El prototipo software deberá cumplir con las siguientes características de rendimiento:

- El tiempo de respuesta para retransmitir el mensaje debe ser inferior al tiempo entre mensajes de eco, es decir debe ser menor a 6 segundos.

## 5. Restricciones de Diseño

No se ha identificado ninguna restricción de diseño.

# ANEXO C: DOCUMENTO DE CASOS DE USO

## 1. Definición de Actores

A continuación se presenta la definición de los actores que interactuarán con el prototipo software. La mayoría de la funcionalidad del prototipo software se ejecutará de manera automática, sin ninguna interacción con el usuario final del sistema (Ver Anexo: Documento de Especificación de Requerimientos Software - SRS). Por lo tanto, el sistema como tal se describirá como un actor.

| **ACTOR: SISTEMA** | |
|---|---|
| **Código** | ACT-1 |
| **Nombre** | Sistema |
| **Autores** | Diego Erik Muñoz Luna<br>Alejandro Muñoz Andrade |
| **Fuentes** | Documento de Especificación de Requerimientos Software - SRS |
| **Descripción** | El sistema será el encargado de publicar, buscar y recuperar el servicio de eco, y procesar el cache del cliente para la recuperación del servicio de eco. También será el encargado de enviar, transmitir y recibir los mensajes de eco, y de lanzar la funcionalidad inicial del servidor y el cliente del servicio de eco. |
| **Comentarios** | Ninguno |

**Tabla 11. Actor Sistema**

| **ACTOR: USUARIO** | |
|---|---|
| **Código** | ACT-2 |
| **Nombre** | Usuario |
| **Autores** | Diego Erik Muñoz Luna<br>Alejandro Muñoz Andrade |
| **Fuentes** | Documento de Especificación de Requerimientos Software - SRS |
| **Descripción** | El usuario podrá iniciar y detener el servidor de eco. El usuario también podrá iniciar y detener el cliente de eco. |
| **Comentarios** | Ninguno |

**Tabla 12. Actor Usuario**

## 2. Casos de Uso en Formato de Alto Nivel

### 2.1. Diagrama de Casos de Uso en Formato de Alto Nivel

Esta sección contiene los diferentes diagramas de casos de uso, que representan las funcionalidades del sistema relacionadas con los actores del mismo, los cuales se definieron anteriormente. La Figura 1 muestra el diagrama de casos de uso de una manera general, organizado por paquetes de acuerdo a la funcionalidad. Los paquetes están representados de la siguiente manera:

- **Paquete Servicio:** Agrupa los casos de uso relacionados con el servicio de eco. Incluye los casos de uso Publicar Servicio, Buscar Servicio y Recuperar Servicio.

- **Paquete Mensajes:** Agrupa los casos de uso relacionados con los mensajes de eco. Incluye los casos de uso Enviar Mensaje, Transmitir Mensaje y Recibir Mensaje.

- **Paquete Opciones:** Agrupa los casos de uso relacionados con las opciones que puede realizar el actor Usuario. Incluye los casos de uso Iniciar Servidor, Detener Servidor, Iniciar Cliente y Detener Cliente. Según la generalización de actores vista en la Figura 1, estos casos de uso también puede ser realizados por el actor Sistema.

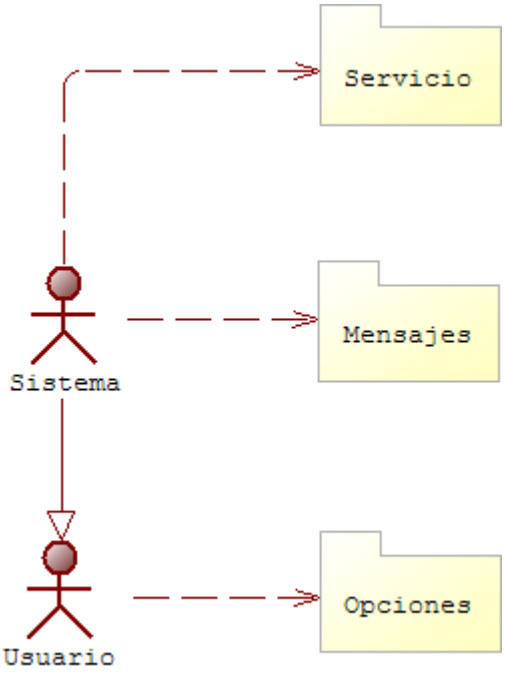


**Figura 1. Diagrama de Casos de Uso - Paquetes**

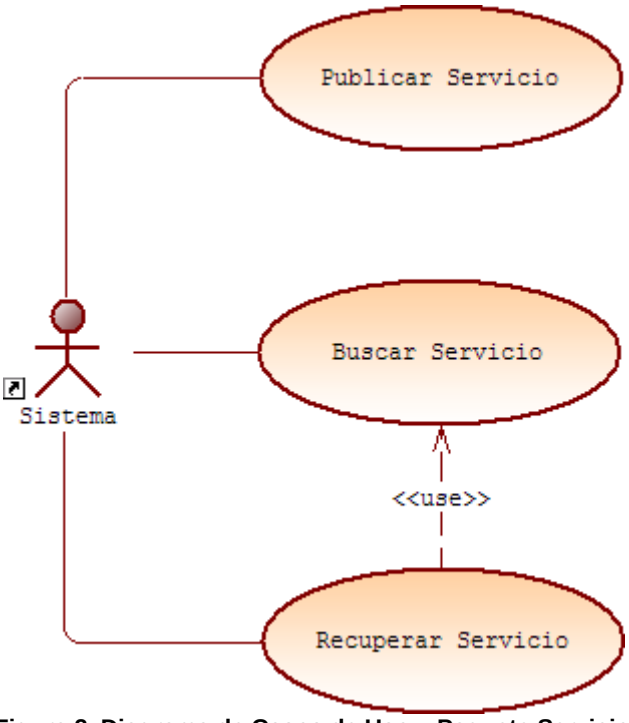


**Figura 2. Diagrama de Casos de Uso – Paquete Servicio**

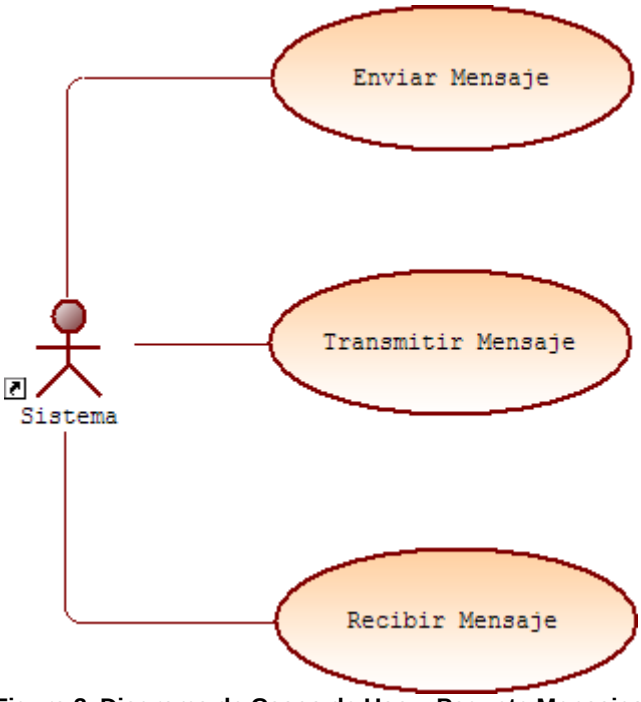


**Figura 3. Diagrama de Casos de Uso – Paquete Mensajes**

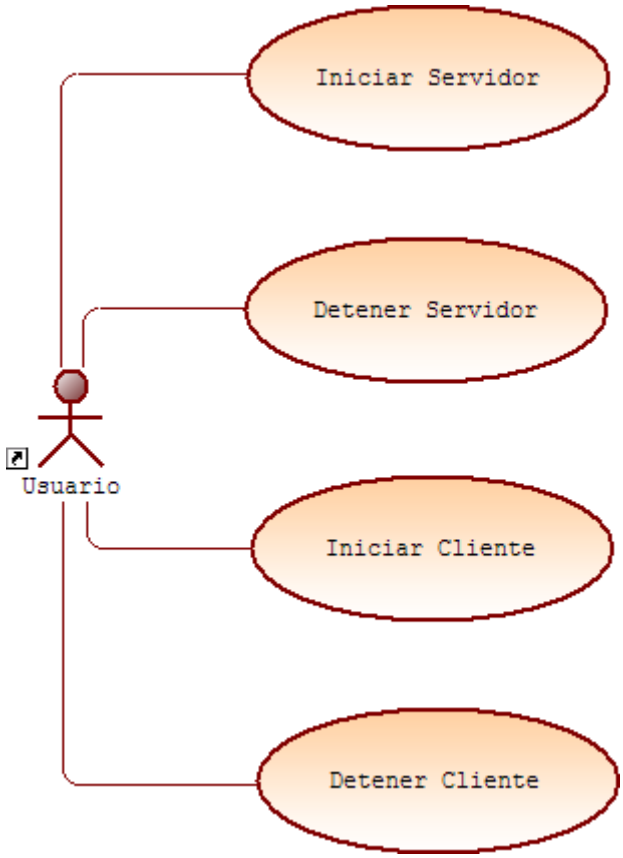


**Figura 4. Diagrama de Casos de Uso – Paquete Opciones**

## 2.2. Descripción de Casos de Uso en Formato de Alto Nivel

### 2.2.1. Casos de Uso del Paquete Servicio

| **CASO DE USO EN FORMATO DE ALTO NIVEL: PUBLICAR SERVICIO** | |
|---|---|
| **Código** | CU-1 |
| **Nombre** | Publicar Servicio |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario |
| **Descripción** | |
| El servicio de eco es creado y publicado tanto en el sistema multi-agente como en la red p2p. Para crear y publicar el servicio de eco se necesitan datos tales como el tipo de servicio y la descripción del servicio. El caso de uso finaliza. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso | Caso de Uso en Formato Extendido |

**Tabla 13. Caso de Uso en Formato de Alto Nivel - Publicar Servicio**

| CASO DE USO EN FORMATO DE ALTO NIVEL: BUSCAR SERVICIO | |
|---|---|
| **Código** | CU-2 |
| **Nombre** | Buscar Servicio |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario |
| **Descripción** | |
| El sistema busca un servicio, por el tipo de servicio, en este caso busca el servicio de eco. Si se encuentra un nodo que preste el servicio buscado, se almacena el nodo para futuras consultas y se envía una petición de conexión al nodo que presta el servicio. Cuando la petición llega al nodo que presta el servicio (servidor), este almacena el nodo que realizo la petición (cliente o servidor), con el fin de realizar futuras consultas y peticiones. El caso de uso finaliza. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso | Caso de Uso en Formato Extendido |

**Tabla 14. Caso de Uso en Formato de Alto Nivel – Buscar Servicio**

| CASO DE USO EN FORMATO DE ALTO NIVEL: RECUPERAR SERVICIO | |
|---|---|
| **Código** | CU-3 |
| **Nombre** | Recuperar Servicio |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario |
| **Descripción** | |
| Si por alguna razón no se pueden enviar peticiones a un nodo que este prestando el servicio de eco, este servidor se elimina. Posteriormente se procede a buscar un nuevo nodo que preste el servicio. Cuando el proceso de búsqueda y almacenamiento del servidor finalice (Caso de uso Buscar Servicio), se procede a recuperar los mensajes que no se han enviado, mediante el uso del cache de mensajes enviados y recibidos. El nodo procesa el cache, obtiene el último mensaje procesado por el servidor (mensaje recibido) y lo busca en la lista de mensajes enviados, para seleccionar los mensajes que no fueron procesados. A continuación el nodo procede a enviar los mensajes que no fueron procesados al nuevo servidor. El caso de uso finaliza. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso<br>Caso de Uso Buscar Servicio | Caso de Uso en Formato Extendido |

**Tabla 15. Caso de Uso en Formato de Alto Nivel – Recuperar Servicio**

## 2.2.2. Casos de Uso del Paquete Mensajes

| CASO DE USO EN FORMATO DE ALTO NIVEL: ENVIAR MENSAJE | |
|---|---|
| **Código** | CU-4 |
| **Nombre** | Enviar Mensaje |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario |
| **Descripción** | |
| Cuando un nodo está conectado a un nodo servidor, se empiezan a enviar los mensajes de eco. Un mensaje de eco es enviado cada 6 segundos, y contiene información tal como: Un código que debe ser único para cada mensaje, el nodo que envía el mensaje (remitente), el nodo al cual se le envía el mensaje (destino), y el contenido del mensaje, que será la hora local del remitente. El nodo que envía el mensaje lo almacena en el cache de mensajes enviados. El caso de uso | |

| | |
|---|---|
| finaliza. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso | Caso de Uso en Formato Extendido |

**Tabla 16. Caso de Uso en Formato de Alto Nivel – Enviar Mensaje**

| **CASO DE USO EN FORMATO DE ALTO NIVEL: TRANSMITIR MENSAJE** | |
|---|---|
| **Código** | CU-5 |
| **Nombre** | Transmitir Mensaje |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario |
| **Descripción** | |
| El caso de uso inicia cuando a un nodo servidor le llega un mensaje desde otro nodo. El nodo servidor recibe el mensaje y lo envía a todos los nodos que estén conectados a él, tanto clientes como servidores, incluyendo al nodo que le envío la petición de transmitir el mensaje, siempre y cuando este nodo sea un cliente. Así se transmite el mensaje a todos los clientes y servidores de eco que estén presentes en la red p2p. El servidor no almacena el mensaje recibido. El caso de uso finaliza. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso | Caso de Uso en Formato Extendido |

**Tabla 17. Caso de Uso en Formato de Alto Nivel – Transmitir Mensaje**

| **CASO DE USO EN FORMATO DE ALTO NIVEL: RECIBIR MENSAJE** | |
|---|---|
| **Código** | CU-6 |
| **Nombre** | Recibir Mensaje |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario |
| **Descripción** | |
| El caso de uso inicia cuando a un nodo cliente le llega un mensaje desde un nodo servidor. El nodo cliente recibe el mensaje, y compara si el mensaje fue enviado por él, es decir si él fue el remitente del mensaje. Si el nodo fue el remitente del mensaje, almacena el mensaje en el cache de mensajes recibidos. El caso de uso finaliza. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso | Caso de Uso en Formato Extendido |

**Tabla 18. Caso de Uso en Formato de Alto Nivel – Recibir Mensaje**

### 2.2.3. Casos de Uso del Paquete Opciones

| **CASO DE USO EN FORMATO DE ALTO NIVEL: INICIAR SERVIDOR** | |
|---|---|
| **Código** | CU-7 |
| **Nombre** | Iniciar Servidor |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario (Sistema), Secundario (Usuario) |
| **Descripción** | |
| El actor da la orden para iniciar el servidor, por lo tanto se inicia el servicio de eco y se busca un nodo que preste el servicio. | |

| |
|---|
| Para que el servidor se inicie pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se inicia. El caso de uso se inicia.<br>• El usuario desea iniciar el servidor, que fue detenido anteriormente. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores inicie el servidor. |

| **TRAZABILIDAD** | |
|---|---|
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso<br>Caso de Uso Publicar Servicio<br>Caso de Uso Buscar Servicio | Caso de Uso en Formato Extendido |

**Tabla 19. Caso de Uso en Formato de Alto Nivel – Iniciar Servidor**

| **CASO DE USO EN FORMATO DE ALTO NIVEL: DETENER SERVIDOR** | |
|---|---|
| **Código** | CU-8 |
| **Nombre** | Detener Servidor |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario (Sistema), Secundario (Usuario) |
| **Descripción** | |
| El actor da la orden para detener el servidor, por lo tanto se desconecta de la red p2p y el servicio que este estaba prestando ya no estará disponible.<br>Para que el servidor se detenga pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se cierra. El caso de uso se inicia.<br>• El usuario desea detener el servidor, que fue iniciado anteriormente o que está en funcionamiento. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores detenga el servidor. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso | Caso de Uso en Formato Extendido |

**Tabla 20. Caso de Uso en Formato de Alto Nivel – Detener Servidor**

| **CASO DE USO EN FORMATO DE ALTO NIVEL: INICIAR CLIENTE** | |
|---|---|
| **Código** | CU-9 |
| **Nombre** | Iniciar Cliente |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario (Sistema), Secundario (Usuario) |
| **Descripción** | |
| El actor da la orden para iniciar el cliente, por lo tanto se busca un nodo que preste el servicio. Si se encuentra un nodo servidor, se crea un nuevo cache que almacenará los mensajes enviados y recibidos.<br>Para que el cliente se inicie pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se inicia. El caso de uso se inicia.<br>• El usuario desea iniciar el cliente, que fue detenido anteriormente. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores inicie el cliente. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso<br>Caso de Uso Buscar Servicio | Caso de Uso en Formato Extendido |

**Tabla 21. Caso de Uso en Formato de Alto Nivel – Iniciar Cliente**

| CASO DE USO EN FORMATO DE ALTO NIVEL: DETENER CLIENTE | |
|---|---|
| **Código** | CU-10 |
| **Nombre** | Detener Cliente |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario (Sistema), Secundario (Usuario) |
| **Descripción** | |
| El actor da la orden para detener el cliente, por lo tanto se desconecta de la red p2p y deja de enviar mensaje de eco.<br>Para que el cliente se detenga pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se cierra. El caso de uso se inicia.<br>• El usuario desea detener el cliente, que fue iniciado anteriormente o que está en funcionamiento. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores detenga el cliente. | |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso | Caso de Uso en Formato Extendido |

**Tabla 22. Caso de Uso en Formato de Alto Nivel – Detener Cliente**

## 3. Casos de Uso en Formato Extendido

### 3.1. Diagrama de Casos de Uso en Formato Extendido

Esta sección contiene los diferentes diagramas de casos de uso en formato extendido, que representan las funcionalidades del sistema relacionadas con los actores del mismo, los cuales se definieron anteriormente. Los paquetes representados aquí son los mismos paquetes que se representaron en el diagrama de casos de uso de alto nivel.

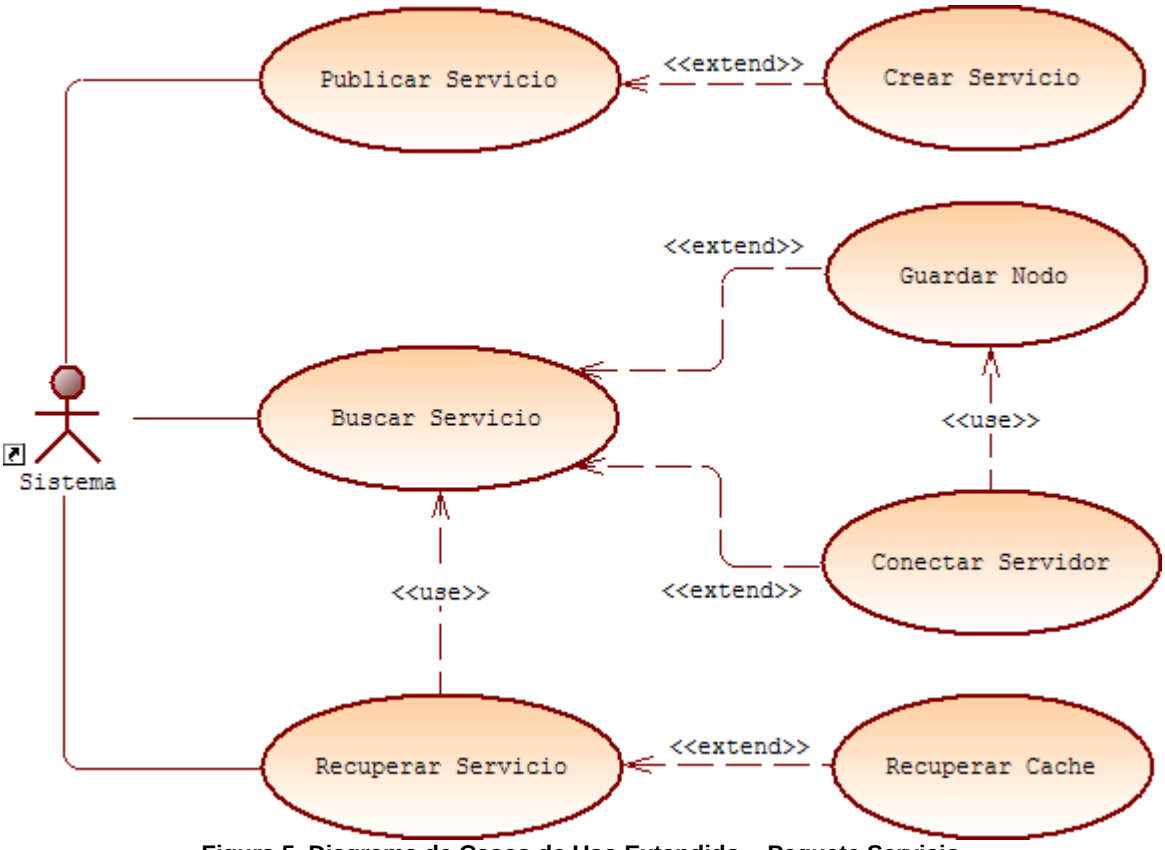


**Figura 5. Diagrama de Casos de Uso Extendido – Paquete Servicio**

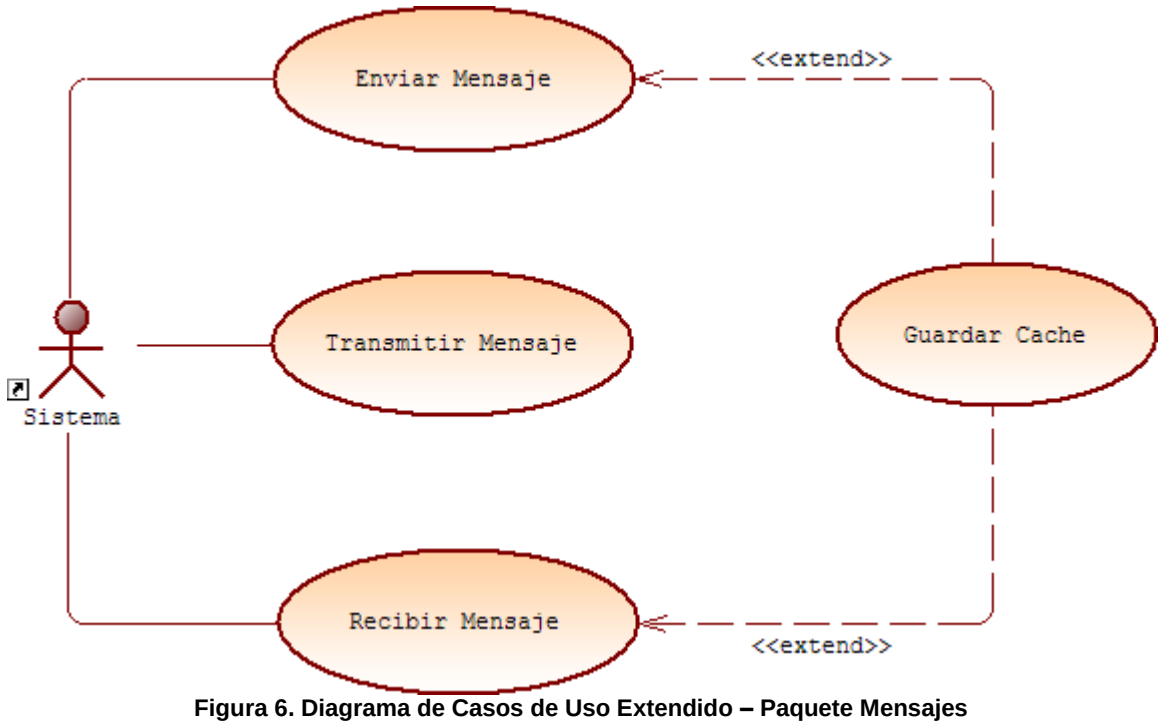


**Figura 6. Diagrama de Casos de Uso Extendido – Paquete Mensajes**

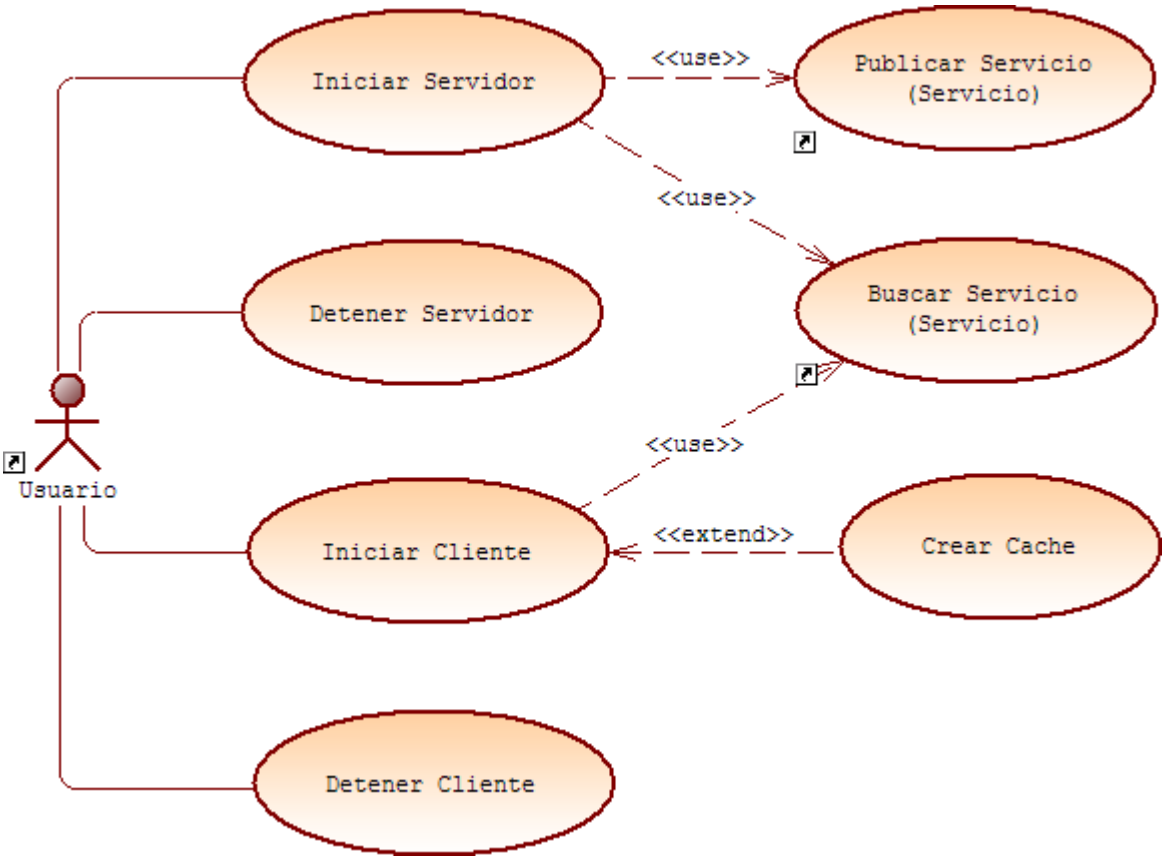


**Figura 7. Diagrama de Casos de Uso Extendido – Paquete Opciones**

## 3.2. Descripción de Casos de Uso en Formato Extendido

### 3.2.1. Casos de Uso del Paquete Servicio

| **CASO DE USO EN FORMATO EXTENDIDO: PUBLICAR SERVICIO** | |
|---|---|
| **Código** | CUEX-1 |
| **Nombre** | Publicar Servicio |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario y Esencial |
| **Precondición** | Ninguna |
| **Postcondiciones** | Ninguna |
| **Descripción** | |
| El servicio de eco es creado y publicado tanto en el sistema multi-agente como en la red p2p. Para crear y publicar el servicio de eco se necesitan datos tales como el tipo de servicio y la descripción del servicio. El caso de uso finaliza. | |
| **Curso Normal de Eventos** | 1. El caso de uso inicia cuando el sistema se inicia o el servidor de eco se inicia.<br>2. El sistema crea el servicio de eco, usando el tipo de servicio y la descripción del servicio.<br>3. El sistema publica el servicio de eco que fue creado, en el sistema multi-agente y en la red p2p. |

| | |
|---|---|
| | 4. El caso de uso finaliza. |
| **Excepciones** | 2.1. El sistema no puede crear el servicio. El sistema se cierra.<br>3.1. El sistema no puede publicar el servicio. El sistema se cierra. |
| **Flujos Alternativos** | Ninguno |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

**Tabla 23. Caso de Uso en Formato Extendido – Publicar Servicio**

| **CASO DE USO EN FORMATO EXTENDIDO: BUSCAR SERVICIO** | |
|---|---|
| **Código** | CUEX-2 |
| **Nombre** | Buscar Servicio |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario y Esencial |
| **Precondición** | Ninguna |
| **Postcondiciones** | Se empiezan a enviar los mensajes de eco. |
| **Descripción** | |
| El sistema busca un servicio, por el tipo de servicio, en este caso busca el servicio de eco. Si se encuentra un nodo que preste el servicio buscado, se almacena el nodo para futuras consultas y se envía una petición de conexión al nodo que presta el servicio. Cuando la petición llega al nodo que presta el servicio (servidor), este almacena el nodo que realizo la petición (cliente o servidor), con el fin de realizar futuras consultas y peticiones. Los nodos están conectados. El caso de uso finaliza. | |
| **Curso Normal de Eventos** | 1. El caso de uso inicia cuando el Sistema busca un servicio.<br>2. El sistema busca un servicio por el tipo de servicio.<br>3. Se encuentra un nodo que presta el servicio buscado (servidor).<br>4. Se almacena el nodo servidor.<br>5. El sistema envía una petición de conexión al servidor.<br>6. La petición de conexión llega al servidor.<br>7. El servidor almacena el nodo que realizó la petición de conexión.<br>8. Los nodos están conectados en la red p2p.<br>9. El caso de uso finaliza. |
| **Excepciones** | 3.1. El sistema no puede encontrar un servidor. El sistema sigue buscando indefinidamente por un servidor.<br>5.1. No se puede enviar la petición de conexión al servidor. El servidor se elimina. El caso de uso se reinicia en el paso 2 del curso normal de eventos. |
| **Flujos Alternativos** | Ninguno |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

**Tabla 24. Caso de Uso en Formato Extendido – Buscar Servicio**

| **CASO DE USO EN FORMATO EXTENDIDO: RECUPERAR SERVICIO** | |
|---|---|
| **Código** | CUEX-3 |
| **Nombre** | Recuperar Servicio |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario y Esencial |
| **Precondición** | El sistema no puede enviar un mensaje al servidor. |
| **Postcondiciones** | Se continua con el envió de mensajes de eco al servidor. |

| Descripción | |
|---|---|
| Si por alguna razón no se pueden enviar peticiones a un nodo que este prestando el servicio de eco, este servidor se elimina. Posteriormente se procede a buscar un nuevo nodo que preste el servicio. Cuando el proceso de búsqueda y almacenamiento del servidor finalice (Caso de uso Buscar Servicio), se procede a recuperar los mensajes que no se han enviado, mediante el uso del cache de mensajes enviados y recibidos. El nodo procesa el cache, obtiene el último mensaje procesado por el servidor (mensaje recibido) y lo busca en la lista de mensajes enviados, para seleccionar los mensajes que no fueron procesados. A continuación el nodo procede a enviar los mensajes que no fueron procesados al nuevo servidor. El caso de uso finaliza. | |
| **Curso Normal de Eventos** | 1. El caso de uso inicia cuando no se pueden enviar peticiones al servidor.<br>2. El sistema busca un nuevo servidor.<br>3. El sistema inicia el proceso de recuperación del servicio mediante el cache de mensajes enviados y recibidos.<br>4. El sistema obtiene el último mensaje recibido.<br>5. El sistema busca el último mensaje recibido en la lista de mensajes enviados, de acuerdo al código del mensaje.<br>6. El sistema selecciona los mensajes que no fueron procesados.<br>7. El sistema envía los mensajes que no fueron procesados al nuevo servidor.<br>8. El caso de uso finaliza. |
| **Excepciones** | 7.1. El sistema no puede enviar los mensajes al nuevo servidor. El caso de uso se reinicia en el paso 2 del curso normal de eventos. |
| **Flujos Alternativos** | 2.1. Se inicia el Caso de Uso Buscar Servicio |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido<br>Caso de Uso Buscar Servicio | Ninguno |

**Tabla 25. Caso de Uso en Formato Extendido – Recuperar Servicio**

## 3.2.2. Casos de Uso del Paquete Mensajes

| CASO DE USO EN FORMATO EXTENDIDO: ENVIAR MENSAJE | |
|---|---|
| **Código** | CUEX-4 |
| **Nombre** | Enviar Mensaje |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario y Esencial |
| **Precondición** | El nodo debe estar conectado a un servidor. |
| **Postcondiciones** | Se guarda el mensaje en el cache de mensajes enviados. |
| **Descripción** | |
| Cuando un nodo está conectado a un nodo servidor, se empiezan a enviar los mensajes de eco. Un mensaje de eco es enviado cada 6 segundos, y contiene información tal como: Un código que debe ser único para cada mensaje, el nodo que envía el mensaje (remitente), el nodo al cual se le envía el mensaje (destino), y el contenido del mensaje, que será la hora local del remitente. El nodo que envía el mensaje lo almacena en el cache de mensajes enviados. El caso de uso finaliza. | |
| **Curso Normal de Eventos** | 1. El caso de uso inicia cuando un nodo cliente se conecta a un nodo servidor.<br>2. El sistema crea el mensaje de eco que contiene la siguiente información: El código, el remitente, el destino, y el contenido del mensaje, que será la hora local del sistema. |

| | |
|---|---|
| | 3. El sistema envía el mensaje al servidor.<br>4. El sistema almacena el mensaje en la cache de mensajes enviados.<br>5. El caso de uso finaliza. |
| **Excepciones** | 3.1. El sistema no puede enviar el mensaje al servidor. Se inicia el Caso de Uso Recuperar Servicio. |
| **Flujos Alternativos** | Ninguno |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

**Tabla 26. Caso de Uso en Formato Extendido – Enviar Mensaje**

| **CASO DE USO EN FORMATO EXTENDIDO: TRANSMITIR MENSAJE** | |
|---|---|
| **Código** | CUEX-5 |
| **Nombre** | Transmitir Mensaje |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario y Esencial |
| **Precondición** | El servidor recibe un mensaje de eco. |
| **Postcondiciones** | Ninguna |
| **Descripción** | |
| El caso de uso inicia cuando a un nodo servidor le llega un mensaje desde otro nodo. El nodo servidor recibe el mensaje y lo envía a todos los nodos que estén conectados a él, tanto clientes como servidores, incluyendo al nodo que le envío la petición de transmitir el mensaje, siempre y cuando este nodo sea un cliente. Así se transmite el mensaje a todos los clientes y servidores de eco que estén presentes en la red p2p. El servidor no almacena el mensaje recibido. El caso de uso finaliza. | |
| **Curso Normal de Eventos** | 1. El caso de uso inicia cuando un nodo servidor recibe un mensaje de eco desde otro nodo.<br>2. El servidor transmite el mensaje a todos los nodos que estén conectados a él, tanto clientes como servidores. Se transmite un mensaje por cada nodo que esté conectado actualmente. El destino de cada mensaje cambia para contener al nodo al cual será transmitido el mensaje.<br>3. Si el nodo que envió el mensaje es un nodo cliente, el servidor le transmite también el mensaje.<br>4. El caso de uso finaliza. |
| **Excepciones** | 2.1. Si no se puede encontrar un nodo, el mensaje se descarta y el nodo se elimina.<br>3.1. Si no se encuentra el cliente, el mensaje se descarta y el cliente se elimina. |
| **Flujos Alternativos** | Ninguno |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

**Tabla 27. Caso de Uso en Formato Extendido – Transmitir Mensaje**

| **CASO DE USO EN FORMATO EXTENDIDO: RECIBIR MENSAJE** | |
|---|---|
| **Código** | CUEX-6 |
| **Nombre** | Recibir Mensaje |
| **Actores** | |
| Sistema | |
| **Tipo** | Primario y Esencial |

| | |
|---|---|
| **Precondición** | El cliente recibe un mensaje de eco. |
| **Postcondiciones** | Se guarda el mensaje en el cache de mensajes recibidos. |
| **Descripción** | |
| El caso de uso inicia cuando a un nodo cliente le llega un mensaje desde un nodo servidor. El nodo cliente recibe el mensaje, y compara si el mensaje fue enviado por él, es decir si él fue el remitente del mensaje. Si el nodo fue el remitente del mensaje, almacena el mensaje en el cache de mensajes recibidos. El caso de uso finaliza. | |
| **Curso Normal de Eventos** | 1. El caso de uso inicia cuando un nodo cliente recibe un mensaje de eco desde un servidor.<br>2. El cliente recibe el mensaje de eco.<br>3. Se guarda el mensaje en el cache de mensajes recibidos, siempre y cuando el mensaje de eco haya sido enviado por este cliente. Esto quiere decir que el cliente almacena los mensajes que han sido transmitidos exitosamente por el servidor en un cache de mensajes recibidos.<br>4. El caso de uso finaliza. |
| **Excepciones** | Ninguna |
| **Flujos Alternativos** | Ninguno |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

**Tabla 28. Caso de Uso en Formato Extendido – Recibir Mensaje**

### 3.2.3. Casos de Uso del Paquete Opciones

| | |
|---|---|
| **CASO DE USO EN FORMATO EXTENDIDO: INICIAR SERVIDOR** | |
| **Código** | CUEX-7 |
| **Nombre** | Iniciar Servidor |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario y Esencial (Sistema), Secundario y Deseado (Usuario) |
| **Precondición** | El servidor debe estar detenido. |
| **Postcondiciones** | El servicio de eco esta publicado. |
| **Descripción** | |
| El actor da la orden para iniciar el servidor, por lo tanto se inicia el servicio de eco y se busca un nodo que preste el servicio.<br>Para que el servidor se inicie pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se inicia. El caso de uso se inicia.<br>• El usuario desea iniciar el servidor, que fue detenido anteriormente. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores inicie el servidor. | |
| **Curso Normal de Eventos** | 1. El actor inicia el servidor.<br>2. Se publica el servicio de eco.<br>3. Se busca el servicio de eco.<br>4. El caso de uso finaliza. |
| **Excepciones** | 1.1. No se puede iniciar el servidor. Se muestra un mensaje de error. |
| **Flujos Alternativos** | 2.1. Se inicia el Caso de Uso Publicar Servicio<br>3.1. Se inicia el Caso de Uso Buscar Servicio |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido<br>Caso de Uso Publicar y Buscar Servicio | Ninguno |

**Tabla 29. Caso de Uso en Formato Extendido – Iniciar Servidor**

| **CASO DE USO EN FORMATO EXTENDIDO: DETENER SERVIDOR** | |
|---|---|
| **Código** | CUEX-8 |
| **Nombre** | Detener Servidor |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario y Esencial (Sistema), Secundario y Deseado (Usuario) |
| **Precondición** | El servidor debe estar iniciado. |
| **Postcondiciones** | El servicio se detiene. |
| **Descripción** | |
| El actor da la orden para detener el servidor, por lo tanto se desconecta de la red p2p y el servicio que este estaba prestando ya no estará disponible.<br>Para que el servidor se detenga pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se cierra. El caso de uso se inicia.<br>• El usuario desea detener el servidor, que fue iniciado anteriormente o que está en funcionamiento. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores detenga el servidor. | |
| **Curso Normal de Eventos** | 1. El actor detiene el servidor.<br>2. El caso de uso finaliza. |
| **Excepciones** | 1.1. No se puede detener el servidor. Se muestra un mensaje de error. |
| **Flujos Alternativos** | Ninguno |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

**Tabla 30. Caso de Uso en Formato Extendido – Detener Servidor**

| **CASO DE USO EN FORMATO EXTENDIDO: INICIAR CLIENTE** | |
|---|---|
| **Código** | CUEX-9 |
| **Nombre** | Iniciar Cliente |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario y Esencial (Sistema), Secundario y Deseado (Usuario) |
| **Precondición** | El cliente debe estar detenido. |
| **Postcondiciones** | El cliente se inicia. |
| **Descripción** | |
| El actor da la orden para iniciar el cliente, por lo tanto se busca un nodo que preste el servicio. Si se encuentra un nodo servidor, se crea un nuevo cache que almacenará los mensajes enviados y recibidos.<br>Para que el cliente se inicie pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se inicia. El caso de uso se inicia.<br>• El usuario desea iniciar el cliente, que fue detenido anteriormente. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores inicie el cliente. | |
| **Curso Normal de Eventos** | 1. El actor inicia el cliente.<br>2. Se busca el servicio de eco.<br>3. Se crea el cache que almacena los mensajes enviados y recibidos.<br>4. Se inicia el envío de mensajes de eco.<br>5. El caso de uso finaliza. |
| **Excepciones** | 1.1. No se puede iniciar el cliente. Se muestra un mensaje de error. |
| **Flujos Alternativos** | 2.1. Se inicia el Caso de Uso Buscar Servicio<br>3.1. Se inicia el Caso de Uso Enviar Mensajes |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

| | |
|---|---|
| Caso de Uso Buscar Servicio<br>Caso de Uso Enviar Mensajes | |

**Tabla 31. Caso de Uso en Formato Extendido – Iniciar Cliente**

| **CASO DE USO EN FORMATO EXTENDIDO: DETENER CLIENTE** | |
|---|---|
| **Código** | CUEX-10 |
| **Nombre** | Detener Cliente |
| **Actores** | |
| Sistema, Usuario | |
| **Tipo** | Primario y Esencial (Sistema), Secundario y Deseado (Usuario) |
| **Precondición** | El servidor debe estar iniciado. |
| **Postcondiciones** | El cliente se detiene. |
| **Descripción** | |
| El actor da la orden para detener el cliente, por lo tanto se desconecta de la red p2p y deja de enviar mensaje de eco.<br>Para que el cliente se detenga pueden ocurrir dos casos dependiendo del actor:<br>• El sistema se cierra. El caso de uso se inicia.<br>• El usuario desea detener el cliente, que fue iniciado anteriormente o que está en funcionamiento. El caso de uso se inicia.<br>El caso de uso finaliza después de que cualquiera de los actores detenga el cliente. | |
| **Curso Normal de Eventos** | 1. El actor detiene el cliente.<br>2. El caso de uso finaliza. |
| **Excepciones** | 1.1. No se puede detener el cliente. Se muestra un mensaje de error. |
| **Flujos Alternativos** | Ninguno |
| **TRAZABILIDAD** | |
| **Artefactos Anteriores** | **Artefactos Posteriores** |
| Diagrama de Casos de Uso Extendido | Ninguno |

**Tabla 32. Caso de Uso en Formato Extendido – Detener Cliente**

# ANEXO D: TECNOLOGÍAS PARA EL DESARROLLO DE REDES SUPERPUESTAS P2P Y SISTEMAS MULTI-AGENTE

En este capítulo se describen las tecnologías escogidas para la implementación de la red superpuesta P2P y el Sistema Multi-Agente en el prototipo software.

## 1. JXTA

### 1.1. ¿Qué es JXTA?

JXTA es un conjunto de protocolos P2P abiertos y generalizados, que permiten a cualquier dispositivo de red[1] comunicarse y colaborar mutuamente como peers o iguales. Los protocolos JXTA son independientes del lenguaje de programación, y existen múltiples implementaciones para diferentes entornos. El uso común de los protocolos JXTA significa que ellos son totalmente interoperables [4].

JXTA es una plataforma de computación de red abierta, diseñada para computación P2P, que provee los bloques de construcción básica y servicios requeridos para permitir conectividad, a cualquier aplicación donde quiera que esta se encuentre.

El nombre “JXTA” no es un acrónimo. Es una contracción para *yuxtapose*, que significa, colocado junto a algo o en posición inmediata a algo. Es bien conocido que P2P es yuxtapuesto, a la computación cliente-servidor, o a la computación basada en Web, las cuales son modelos tradicionales de computación distribuida hoy en día.

JXTA provee un conjunto común de protocolos abiertos, que cuentan con implementaciones de referencia, de código abierto para el desarrollo de aplicaciones P2P. Los protocolos JXTA estandarizan la forma en que sus nodos:

- Se descubren.
- Se auto-organizan en grupos de nodos.
- Publican y descubren recursos en la red.
- Se comunican entre sí.
- Se supervisan.

Los protocolos JXTA son diseñados para ser independientes de lenguajes de programación y protocolos de transporte. Los protocolos de comunicación pueden ser implementados en lenguaje de programación Java, C/C++, .NET, Ruby y muchos otros lenguajes. Por otra parte, pueden ser aplicados sobre TCP/IP,

[1] Entre los dispositivos de red podemos encontrar: Sensores, Teléfonos Celulares, PDAs, Computadores Portátiles y de Escritorio, Estaciones de Trabajo, Servidores y Supercomputadores.

HTTP, Bluetooth y otras redes de transporte, manteniendo interoperabilidad mundial.

## 1.2. ¿Por qué JXTA?

A medida que la web continúa creciendo en contenido y en número de dispositivos conectados, la computación P2P está llegando a ser cada vez más frecuente. Ejemplos populares incluyen intercambio de archivos, computación distribuida, y servicios de mensajería instantánea. Mientras cada una de estas aplicaciones realiza tareas diferentes, todas ellas comparten muchas de las mismas propiedades, tales como descubrimiento de nodos, búsqueda y transferencia de archivos o datos. Actualmente, el desarrollo de aplicaciones P2P es ineficiente, la mayoría de las aplicaciones son desarrolladas específicamente para una sola plataforma y no pueden comunicarse y compartir datos con otras aplicaciones [JXTA, 2010].

Un principio de diseño primario de JXTA es proveer una plataforma que contenga las funciones básicas de las redes superpuestas P2P. JXTA supera los defectos potenciales de muchos sistemas P2P existentes:

- **Interoperabilidad:** La tecnología JXTA, está diseñada para que un nodo pueda proveer servicios P2P, para localizar y comunicarse el uno con otro independientemente de direcciones de red y protocolos físicos.

- **Independiente de la Plataforma:** La tecnología JXTA, está diseñada para ser independiente del lenguaje de programación, protocolos de transporte de red y plataformas de despliegue.

- **Ubicuidad[2]:** La tecnología JXTA, está diseñada para ser accesible por cualquier dispositivo digital y en cualquier plataforma de despliegue.

Los protocolos JXTA, proporcionan a los desarrolladores la capacidad de construir y desplegar servicios y aplicaciones P2P interoperables. Debido a que los protocolos de JXTA son independientes del lenguaje de programación y los protocolos de transporte, varios dispositivos con diferente software pueden interoperar entre ellos.

Usando la tecnología JXTA los desarrolladores pueden escribir aplicaciones de red interoperables que pueden:

- Buscar a otros nodos en la red, mediante el descubrimiento dinámico, a través de firewalls y NATs.

[2] Ubicuidad significa que puede estar presente en cualquier parte.

- Compartir recursos fácilmente a través de la red.
- Buscar contenido disponible en los sitios de red.
- Crear un grupo de nodos que provean un servicio.
- Monitorear remotamente las actividades de los nodos.
- Comunicarse de forma segura con otros nodos en la red.

### 1.3. Arquitectura JXTA

La arquitectura software de JXTA está dividida en tres capas [4] [5], como se muestra en la Figura 8.

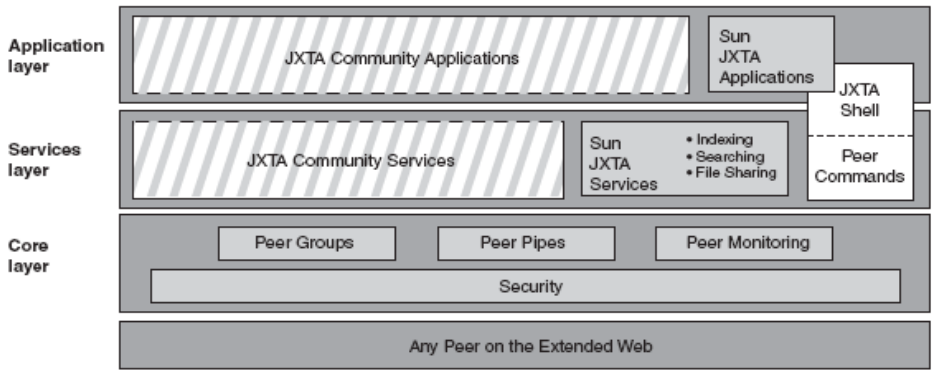


**Figura 8. Arquitectura JXTA**

- **Capa del Núcleo JXTA:** El núcleo de JXTA encapsula las primitivas mínimas y esenciales, que son comunes para la comunicación P2P. Esto incluye componentes básicos que permiten mecanismos claves para las aplicaciones P2P, incluyendo descubrimiento, transportes de comunicación, creación de nodos y grupos de nodos, y primitivas de seguridad. El núcleo es donde se encuentra el código para la implementación de los protocolos. Los protocolos proveen la funcionalidad para los nodos, grupos de nodos, seguridad y monitoreo, así como también el paso de mensajes y protocolos de red.

- **Capa de Servicios:** Un servicio es una funcionalidad, construida por encima del núcleo de JXTA, que usa los protocolos para cumplir una tarea asignada. La capa de servicios incluye los servicios de red que pueden no ser absolutamente necesarios para que una red superpuesta P2P opere, pero son comunes o deseables en un ambiente P2P. Ejemplos de servicios de red incluyen búsqueda e indexación, directorios, sistemas de almacenamiento, uso compartido de archivos, sistemas de archivos distribuidos, agregación y alquiler de recursos, protocolo de translación, autenticación y servicios de PKI[3].

[3] Public Key Infrastructure

- **Capa de Aplicaciones:** La capa de aplicaciones incluye implementación de aplicaciones integradas, como mensajería instantánea P2P, distribución de documentos y de recursos, gestión y entrega de contenidos de entretenimiento, sistemas de correo electrónico P2P y muchos otros.

Uno de los puntos importantes de la arquitectura JXTA, es que la línea entre las capas de la arquitectura no es rígida. Una aplicación de un cliente puede considerarse como un servicio para otro cliente. Si se desarrolla un nodo que provee alguna funcionalidad, otro nodo puede usar parte de esa funcionalidad como un servicio que se ajuste a lo que él necesita y le puede agregar otros módulos de software para complementarlo; pero un tercer nodo puede ver esa funcionalidad como una aplicación completa, sin necesidad de agregarle otros módulos. JXTA está diseñada para ser modular, permitiendo a los desarrolladores seleccionar y escoger una colección de servicios y aplicaciones de acuerdo a sus necesidades.

Existen cuatro aspectos esenciales de la arquitectura JXTA que la distinguen de otros modelos de red distribuida:

- El uso de documentos XML, llamados *advertisements*, para describir recursos de red.
- La abstracción de *pipes* a *nodos*, y de *nodos* a *endpoints*, sin depender de una autoridad central de nombramiento/direccionamiento, como es el caso de DNS.
- Un esquema de direccionamiento uniforme de nodos.
- Una infraestructura de búsqueda descentralizada basada sobre DHT[4], para indexación de recursos.

### 1.4. Protocolos JXTA

JXTA define una serie de mensajes XML, también conocidos como protocolos, para la comunicación entre nodos. Los nodos usan estos protocolos para descubrir otros nodos, anunciar y descubrir recursos de red y para la comunicación y el enrutamiento de mensajes [4].

JXTA define seis protocolos estándar [4]:

- **Peer Discovery Protocol (PDP):** Usado por los nodos para anunciar sus propios recursos y descubrir recursos de otros nodos. Cada recurso de un nodo es descrito y publicado usando un *advertisement*. Un recurso puede ser un nodo, un grupo, un pipe, un servicio o cualquier otro recurso que se tenga un advertisement. Este protocolo permite a un nodo buscar advertisements

[4] Distributed Hash Table

sobre la red. La actual implementación usa una combinación de IP multicast y el uso de súper-nodos mediante los nodos *rendezvous*.

- **Peer Information Protocol (PIP):** Usado por los nodos para obtener información del estado de otros nodos. Este protocolo provee un conjunto de mensajes para obtener la información de estado actual de los nodos en la red. Un mensaje *ping* es enviado a otro nodo para verificar si el nodo está vivo y obtener información acerca de este. El mensaje ping especifica que puede ser retronada una respuesta completa, mediante un advertisement, o una respuesta simple conocimiento, como por ejemplo *live* o *uptime*. Un mensaje *PeerInfo* es usado para enviar un mensaje de respuesta a un mensaje *ping*. El mensaje PingInfo contiene la credencial del remitente, el ID del nodo fuente, el ID del nodo destino, el tiempo de vida y el advertisement del nodo.

- **Peer Resolver Protocol (PRP):** Permite a los nodos enviar una consulta genérica a uno o más nodos y recibir una o múltiples respuestas a la consulta. Las consultas pueden ir dirigidas a todos los nodos en un grupo o a nodos específicos dentro del grupo. A diferencia de PDP y PIP, los cuales son usados para realizar consultas especificas con información predefinida, este protocolo permite a los nodos y servicios definir e intercambiar cualquier información arbitraria que necesiten. Este protocolo usa el Servicio Rendezvous para diseminar una consulta a múltiples nodos y usa mensajes unicast para enviar consultas a nodos específicos. Los protocolos PDP y PIP están construidos usando PRP.

- **Pipe Binding Protocol (PBP):** Usado por los nodos para establecer un canal de comunicación virtual, o *pipe*, entre uno o más nodos. Este protocolo es usado por un nodo para establecer dos o más puntos de conexión finales, también llamados *endpoints*. El enlace virtual del pipe puede encapsular cualquier cantidad de enlaces de transporte de redes físicas. Cada punto final del pipe trabaja para mantener el enlace virtual y para establecerlo de nuevo cuando sea necesario, mediante establecimiento o búsqueda de los endpoints de los pipes actuales.

- **Endpoint Routing Protocol (ERP):** Usado por los nodos para buscar rutas hacia puertos de destino en otros nodos. La información de enrutamiento incluye una secuencia ordenada de los IDs de los nodos que pueden ser usados para enviar un mensaje a su destino. Este protocolo permite a los nodos enviar mensajes a otros nodos sin que exista una conexión directa entre ellos. El mensaje es pasado a través de nodos intermediarios para alcanzar su destino final.

- **Rendezvous Protocol (RVP):** Usado por los nodos para encontrar recursos, propagar mensajes dentro de un grupo y anunciar recursos locales. Es usado

por los nodos *rendezvous* para organizarse con otros nodos *rendezvous*, compartir el espacio de direcciones de las DHT y propagar mensajes en un entorno controlado. Este protocolo posee mecanismos que permiten que la propagación de mensajes se ejecute de una manera controlada y eficiente.

En la Figura 9, se muestra cada uno de los protocolos JXTA y su relación con los otros protocolos [5].

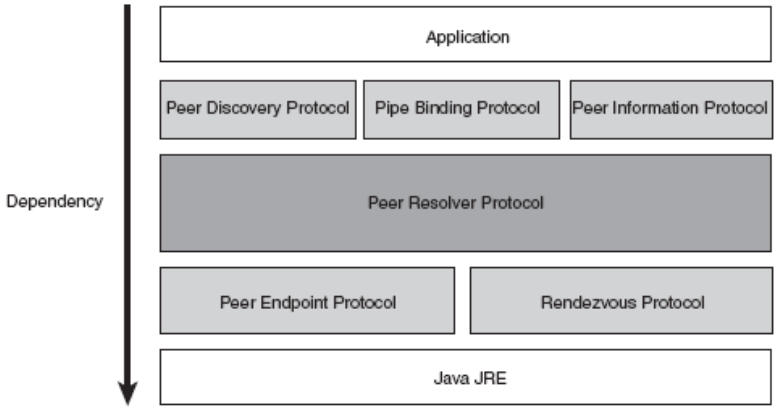


**Figura 9. Protocolos JXTA**

Todos los protocolos de JXTA son asíncronos y están basados en el modelo consulta/respuesta. Por ejemplo, un nodo JXTA usa uno de los protocolos para enviar una consulta a uno o más nodos dentro del grupo, pero este nodo puede recibir cero, una o más respuestas a la consulta realizada. No es necesario que los nodos JXTA implementen los seis protocolos, solo necesitan implementar los protocolos que van a usar [4].

### 1.5. Servicios JXTA

Todos los protocolos estándar de JXTA son implementados como servicios, llamados servicios del núcleo o *core services*, los cuales son los siguientes [6]:

- Discovery Service
- Peer Info Service
- Resolver Service
- Pipe Service
- Endpoint Service
- Rendezvous Service

Una instancia de un servicio es asociada con un grupo de nodos específico. Solo los nodos que son miembros del mismo grupo son capaces de comunicarse con otros nodos del grupo por medio de estos servicios. Por defecto, todos los nodos

tiene un grupo común llamado *Net Peer Group*, este grupo permite que todos los nodos y sus advertisements sean descubiertos. Los servicios proveen a los desarrolladores un nivel de abstracción, ocultando para ellos algunas cosas que no son necesarias conocer o manipular de los protocolos. [Wilson, 2002].

## 2. JADE

### 2.1. ¿Qué es JADE?

JADE[5] es un framework software totalmente implementado en lenguaje Java. Simplifica la implementación de Sistemas Multi-Agente a través de un middleware que cumple con las especificaciones FIPA y a través de un conjunto de herramientas gráficas que soportan las fases de depuración y despliegue. La plataforma de Agentes puede ser distribuida de una máquina a otra, sin la necesidad de compartir el mismo sistema operativo, y la configuración puede ser controlada por medio de una interfaz gráfica de usuario remota. La configuración se puede cambiar incluso en tiempo de ejecución, mediante el movimiento de los agentes de una máquina a otra. JADE está totalmente implementado en lenguaje Java y el requerimiento mínimo del sistema es la versión 1.4 de Java [7].

JADE es un framework para desarrollar aplicaciones distribuidas basadas en agentes, en conformidad con las especificaciones FIPA, para la interoperabilidad inteligente de Sistemas Multi-Agente. El objetivo es simplificar el desarrollo, mientras se asegura el cumplimiento de los estándares, a través de un amplio conjunto de servicios del sistema y de los agentes. JADE puede ser considerado un agente middleware, que implementa una plataforma de agentes y un framework de desarrollo. Este se ocupa de todos aquellos aspectos que no son propios de los agentes internos a la plataforma y que son independientes de las aplicaciones, tales como el transporte de mensajes, codificación y análisis, o el ciclo de vida de los agentes.

Una aplicación basada en JADE, está compuesta de un conjunto de componentes llamados *agentes*, los cuales implementan las piezas de la funcionalidad requerida por la aplicación. JADE, proporciona principalmente las abstracciones del Agente y del comportamiento del agente, la distribución transparente de los agentes a través de una amplia gama de dispositivos, la comunicación peer-to-peer entre agentes y un mecanismo de descubrimiento publicación-suscripción, que permite a los agentes encontrar a otros. Además, JADE proporciona una serie de características adicionales, como la movilidad de agentes, soporte al contenido de lenguajes y ontologías, tolerancia a fallos e integración de servicios web y un variado conjunto de herramientas gráficas que facilitan la administración de una aplicación basada en JADE.

[5] Java Agent Development Framework

## 2.2. ¿Por qué JADE?

El objetivo de JADE es simplificar el desarrollo de Sistemas Multi-Agente, que cumplan estándares, a través de un amplio conjunto de servicios del sistema y de los agentes en el cumplimiento de las especificaciones FIPA: Servicio de nombres y servicio de páginas amarillas, transporte de mensajes y servicio de análisis, y una biblioteca de protocolos de interacción FIPA lista para ser utilizada [7].

La plataforma de agentes JADE, cumple con las especificaciones FIPA e incluye todos los elementos obligatorios para administrar la plataforma, los cuales son el AMS, el DF y el MTS. Toda la comunicación de agentes se realiza a través del paso de mensajes, donde FIPA-ACL es el lenguaje para representar los mensajes.

La plataforma de agentes puede ser distribuida en varios equipos. Sólo una aplicación Java, y por lo tanto sólo una Máquina Virtual Java (JVM) es ejecutada en cada host. Cada JVM es básicamente un contenedor de agentes que proporciona un completo entorno de ejecución para la ejecución del agente y permite que varios agentes se ejecuten simultáneamente en el mismo host.

La arquitectura de comunicación ofrece una mensajería flexible y eficiente, donde JADE crea y gestiona una cola de mensajes ACL entrantes, privada para cada agente; los agentes pueden tener acceso a su cola mediante una combinación de varios modos: bloqueo (blocking), votación (polling), tiempo de espera (timeout) y coincidencia basada en patrones (pattern matching based). El modelo completo de comunicaciones FIPA ha sido implementado y sus componentes han sido claramente diferenciados y totalmente integrados: protocolos de interacción, envolturas, ACL, lenguajes de contenido, esquemas de codificación, ontologías y finalmente, protocolos de transporte. El mecanismo de transporte, en particular, es como un camaleón, porque se adapta a cada situación, mediante la elección de forma transparente del mejor protocolo disponible. Java RMI, notificación de eventos, IIOP y HTTP son usados actualmente, pero más protocolos pueden ser fácilmente añadidos a través de las interfaces IMTP JADE y MTP. La mayoría de los protocolos de interacción definidos por FIPA ya están disponibles y pueden ser instanciados después de definir el comportamiento dependiente de la aplicación para cada estado del protocolo. SL y el agente de gestión de ontologías también han sido implementados, así como también el soporte para el contenido de los lenguajes y ontologías que puede definir el usuario pueden ser implementados, registrados con los agentes y usados automáticamente por el framework.

Básicamente, los agentes son implementados como un hilo por cada agente. Pero muchas veces, los agentes necesitan ejecutar tareas paralelas. Además de la solución multi-hilo (multi-thread), ofrecida directamente por el lenguaje Java, JADE soporta también la programación de comportamientos cooperativos, donde JADE programa estas tareas de una forma ligera y eficaz. El tiempo de ejecución (runtime) incluye también algunos comportamientos listos para usar, para las

tareas más comunes en la programación de agentes, tales como protocolos de interacción FIPA, despertar bajo una condición determinada, y estructurar tareas complejas como agregaciones de otras más simples.

### 2.3. Arquitectura JADE

La Figura 10, muestra los elementos de la arquitectura principal de la plataforma JADE [8]. Una plataforma JADE está compuesta de contenedores de agentes que pueden estar distribuidos sobre la red. Los agentes viven en contenedores, los cuales son procesos de Java, que proveen el tiempo de ejecución de JADE y todos los servicios necesarios para almacenar y ejecutar agentes. Hay un contenedor especial, llamado Contenedor Principal, el cual representa el punto de arranque de una plataforma: Este es el primer contenedor a ser ejecutado y todos los otros contenedores deben unirse a un contenedor principal para ser registrados con él.

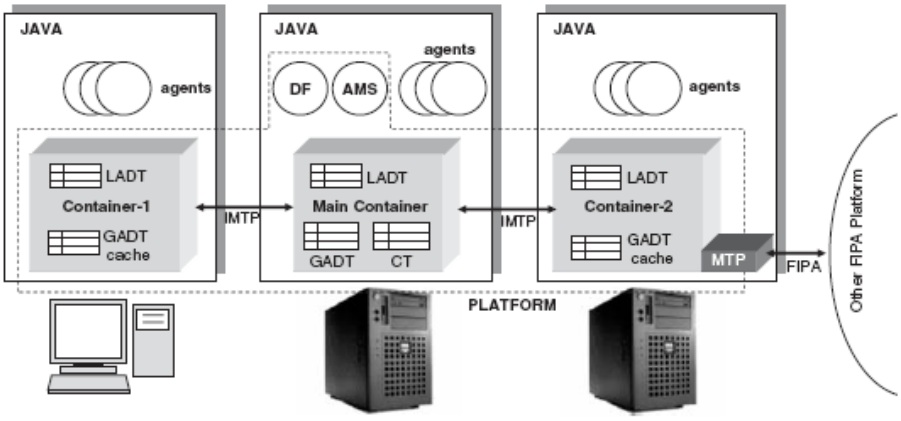


**Figura 10. Arquitectura JADE**

- **Container Table (CT):** Es el registro de las referencias de objeto y las direcciones de transporte de todos los nodos contenedores que componen la plataforma.

- **Global Agent Descriptor Table (GADT):** Es el registro de todos los agentes presentes en la plataforma, incluyendo su estado actual y su localización.

- **Local Agent Descriptor Table (LADT):** Es el registro de todos los agentes presentes en el nodo, incluyendo su estado actual y su localización.

- **Agent Management System (AMS):** Es un agente especial que provee la gestión de agentes y el servicio de páginas blancas de la plataforma.

- **Directory Facilitator (DF):** Es un agente especial que provee el servicio de páginas amarillas de la plataforma.

- **Internal Message Transport Protocol (IMTP):** Es el encargado de la comunicación entre diferentes contenedores de agentes dentro de la misma plataforma.

- **Messages Transport Protocol (MTP):** Es el encargado de la comunicación entre las plataformas, el cual envía y recibe mensajes FIPA.

El diagrama UML de la Figura 11, muestra el esquema de las relaciones entre los elementos principales de la arquitectura JADE [8].

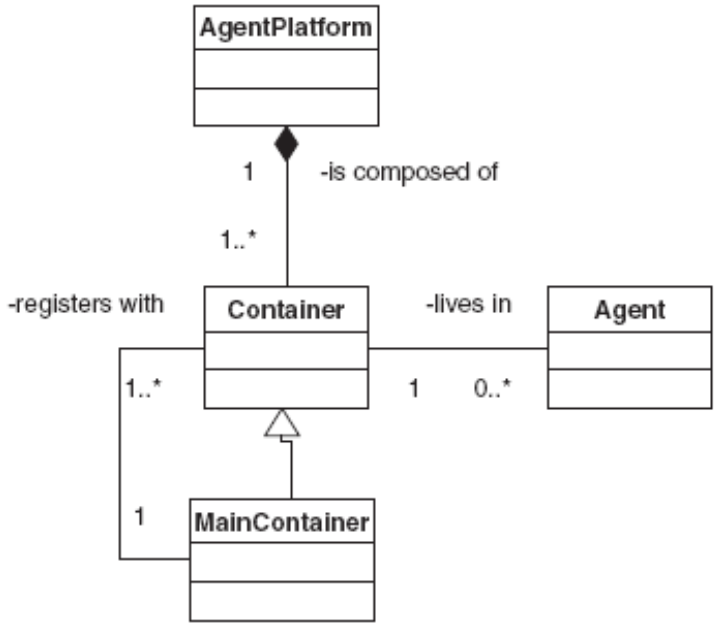


**Figura 11. Relación entre los elementos de la arquitectura principal de JADE**

## 2.4. Comportamientos en JADE

Un comportamiento, representa una tarea que un agente puede ejecutar en un momento dado. Para que un agente ejecute la tarea asignada, el comportamiento debe ser agregado al agente. Un comportamiento puede ser agregado cuando se inicia el agente, dentro de otros comportamientos ya iniciados o por respuesta a un evento ejecutado por un usuario del sistema [8].

Cada comportamiento en JADE debe implementar dos métodos. El método *action()* define las operaciones a ser realizadas cuando el comportamiento se está ejecutando. El método *done()* retorna un valor booleano para indicar cuando un

comportamiento ha sido completado y debe ser removido del conjunto (pool) de comportamientos que un agente está ejecutando [8].

Un comportamiento puede ser terminado cuando se necesite, mediante la llamada al método *removeBehaviour()* de la clase *Agent*. Una llamada a este método remueve la referencia al comportamiento del conjunto de comportamientos que están siendo actualmente ejecutados por el agente [8].

Cada comportamiento tiene una variable miembro llamada *myAgent* que apunta al agente que ejecuta el comportamiento. Esta variable provee una vía fácil para acceder a los recursos del agente dentro del comportamiento [8].

Un agente puede ejecutar varios comportamientos concurrentemente. Sin embargo, es importante notar que la planificación de comportamientos no es pre-vacía, como los hilos java, pero si es cooperativa. Esto significa que cuando un comportamiento es planificado para su ejecución, su método *action()* es llamado y se ejecuta hasta su retorno. Sin embargo, el programador es el que define cuando un agente cambia entre la ejecución de un comportamiento a la ejecución de otro comportamiento [8].

Este enfoque tiene varias ventajas:

- Permite un solo thread java por agente, lo cual es muy importante en entornos con recursos limitados, como por ejemplo teléfonos celulares.

- Provee un rendimiento mejorado, debido a que el cambio entre comportamientos es más rápido que el cambio entre threads java.

- Elimina todos los problemas de sincronización entre comportamientos concurrentes que acceden a los mismos recursos, debido a que todos los comportamientos son ejecutados en el mismo thread java. Esto ofrece una mejora en el rendimiento.

- Cuando ocurre un cambio de comportamiento, el estado de un agente no incluye ninguna información de pila, esto implica que es posible tomar un *'snapshot'* de este. Esto permite la implementación de algunas características avanzadas importantes, como por ejemplo guardar el estado del agente en un medio persistente para una posterior reanudación (Persistencia de agentes), o transferir el agente a otro contenedor para una ejecución remota (Movilidad de agentes).

La ruta de ejecución de un agente con sus comportamientos, es mostrada en la Figura 12 [8].

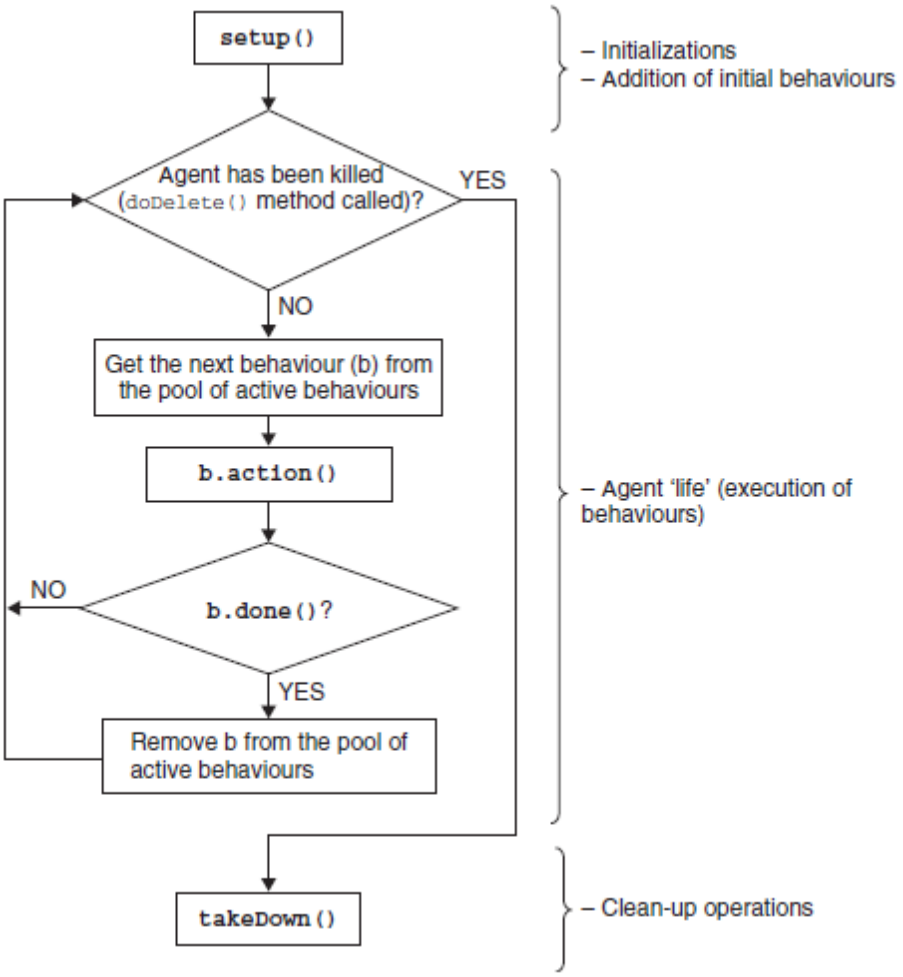


**Figura 12. Ruta de ejecución de un agente**

Cuando no hay comportamientos disponibles para ejecutar, el thread java del agente se duerme (sleep), para no consumir recursos ni tiempo de CPU. El thread es levantado (wakeup) de nuevo cuando exista un comportamiento que esté disponible para ejecutarse.

### 2.4.1. Tipos de Comportamientos

A continuación se describen los tipos de comportamientos que implementa JADE [8] [7]:

- **One-Shot Behaviour:** Son diseñados para ser completados en una sola fase de ejecución; es decir su método *action()* es ejecutado solo una vez. La clase *jade.core.behaviours.OneShotBehaviour* implementa el método *done()* para que retorne el valor booleano *true* y puede ser extendida convenientemente para implementar nuevos comportamientos one-shot.

- **Cyclic Behaviour:** Son diseñados para nunca ser completados; es decir su método *action()* ejecuta las mismas operaciones cada vez que es llamado, hasta que la ejecución del agente que lo contiene termine. La clase *jade.core.behaviours.CyclicBehaviour* implementa el método *done()* para que retorne el valor booleano *false* y puede ser extendida convenientemente para implementar nuevos comportamientos cíclicos (cyclic).

- **Generic Behaviour:** Los comportamientos genéricos, llamados simplemente comportamientos, implementan un disparador de estado y ejecutan varias operaciones dependiendo del valor del estado. Estos comportamientos terminan cuando una condición es cumplida, la cual debe ser comprobada en el método *done()*. Las clases que extiendan *jade.core.behaviours.Behaviour* deben sobrescribir el método *done()* para terminar la ejecución del comportamiento cuando una condición se cumpla.

- **Waker Behaviour:** Este comportamiento pre-implementa los métodos *action()* y *done()*, para ejecutar el método abstracto *onWake()* después de que un determinado tiempo pase. El tiempo de espera debe ser debe ser especificado en el constructor. Después de la ejecución del método *onWake()* el comportamiento se completa.

- **Ticker Behaviour:** Este comportamiento pre-implementa los métodos *action()* y *done()*, para ejecutar el método abstracto onTick() repetidamente, esperando que un determinado tiempo pase después de cada ejecución. El tiempo del periodo debe ser debe ser especificado en el constructor. Un *TickerBehaviour* nunca se completa, a no ser que sea explícitamente removido o su método *stop()* sea llamado.

- **Composite Behaviour:** Un enfoque simple para implementar tareas complejas en JADE es la composición de comportamientos, esto quiere decir crear tareas complejas mediante varios comportamientos simples. Un comportamiento compuesto contiene un número de sub-comportamientos hijo. La clase *CompositeBehaviour* implementa el método *action()* para que cada vez que sea llamado, este invoque el método *action()* de cada uno de los comportamientos hijo.

JADE provee tres tipos de comportamientos compuestos: *SequentialBehaviour*, *FSMBehaviour* y *ParallelBehaviour*.

- **Sequential Behaviour:** La clase *SequentialBehaviour* implementa un comportamiento compuesto que organiza sus hijos de acuerdo a una simple política secuencial. El comportamiento inicia la ejecución de su primer hijo, cuando este termina, continua la ejecución del segundo hijo, y así

sucesivamente hasta que todos sus hijos se hayan ejecutado. Cuando su último hijo es completado, el comportamiento secuencial entero termina. Los sub-comportamientos en un comportamiento secuencial son agregados mediante el método *addSubBehaviour()*. El orden en el cual los sub-comportamientos son agregados determinan el orden en el cual los comportamientos son organizados en el comportamiento secuencial.

- **FSM Behaviour:** La clase *FSMBehaviour* implementa un comportamiento compuesto que organiza sus hijos de acuerdo a una maquina de estados finitos (FSM), en donde cada estado corresponde a uno de los comportamientos hijo. La clase *FSMBehaviour* provee métodos para registrar los sub-comportamientos como estados FSM y para registrar las transiciones entre los estados.

- **Parallel Behaviour:** La clase *ParallelBehaviour* implementa un comportamiento compuesto que organiza sus hijos en paralelo. La organización es cooperativa, esto significa que cada vez que se el método *action()* de un comportamiento paralelo es ejecutado, este invoca el método *action()* del hijo actual y mueve el puntero al siguiente hijo sin importar si este último fue completado o no. Los sub-comportamientos en un comportamiento paralelo son agregados mediante el método *addSubBehaviour()*. La política de terminación del comportamiento es seleccionada en tiempo de instanciación, especificando en el constructor una de las constantes definidas en la clase *ParallelBehaviour*: WHEN_ALL o WHEN_ANY, es decir “cuando todos terminen” o “cuando cualquiera termine”.

## 2.5. Servicios en JADE

JADE, provee un servicio de páginas amarillas, que permite a los agentes descubrir dinámicamente otros agentes que se encuentran disponibles en la plataforma [8].

El servicio de páginas amarillas, permite a los agentes publicar las descripciones de uno o más servicios que ellos proveen, para que otros agentes puedan fácilmente descubrir y explotar estos servicios. El servicio de páginas amarillas se muestra en la Figura 13 [8].

Cualquier agente puede publicar y buscar servicios. La publicación, eliminación, modificación y búsqueda de servicios puede ser ejecutada en cualquier momento durante el tiempo de vida del agente [8].

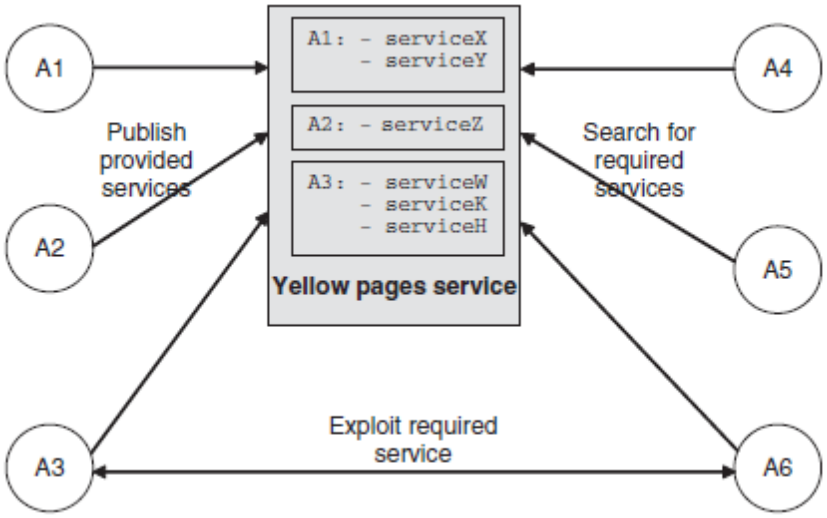


**Figura 13. Servicio de Páginas Amarillas**

El servicio de páginas amarillas en JADE, en concordancia con las especificaciones FIPA, es suministrado por un agente especializado llamado Facilitador de Directorio o DF. Cada plataforma JADE tiene un agente DF por defecto. Como el DF es un agente, es posible que otros agentes interactúen con él mediante el intercambio de mensajes FIPA-ACL, usando un lenguaje de contenido y una ontología propia. Para simplificar estas interacciones JADE provee la clase *jade.domain.DFService* con la cual es posible publicar y buscar servicios a través de una amplia variedad de métodos [8].

### 2.5.1. Publicando Servicios

Un agente que desee publicar uno o más servicios, debe proveer al DF una descripción que incluye: Su propio AID, una lista de los servicios que provee y opcionalmente una lista de lenguajes y ontologías que otros agentes deben usar para interactuar con él. Cada descripción del servicio publicado debe incluir el tipo de servicio, el nombre del servicio, los lenguajes y ontologías requeridos para usar el servicio y una colección de propiedades específicas del servicio, expresadas en pares de la forma clave-valor. Las clases *DFAgentDescription*, *ServiceDescription* y *Property*, incluidas en el paquete *jade.domain.FIPAAgentManagement* representan estas abstracciones [8].

Para publicar un servicio, un agente debe crear una descripción propia (por ejemplo una instancia de la clase *DFAgentDescription*) y llamar al método estático *register()* de la clase *DFService* [8].

### 2.5.2. Buscando Servicios

Un agente que desee buscar servicios debe proveer al DF una plantilla (template) de la descripción del servicio que desea buscar y llamar al método estático *search()* de la clase *DFService*. El resultado de la búsqueda es una lista de todas las descripciones que coinciden con la plantilla que se pasó al agente DF. De acuerdo a las especificaciones FIPA, una descripción del servicio coincide con la plantilla si todos los campos especificados en la plantilla están presentes en la descripción del servicio y todos sus valores coinciden [8].

El agente DF de JADE también provee un mecanismo de suscripción, que permite a los agentes ser notificados tan pronto como otros agentes publiquen o despubliquen sus servicios [8].

# ANEXO E: ESPECIFICACIÓN DE LOS EQUIPOS UTILIZADOS EN LAS PRUEBAS

A continuación se muestra la ficha técnica con las respectivas especificaciones de cada equipo usado en las pruebas y la verificación de la técnica de búsqueda.

| **EQUIPO** | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-010** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | WDC WD400EB-00CPF0 | **CAPACIDAD** | | 37 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 256 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS XP | | | | |

**Tabla 33. Equipo PC-010**

| **EQUIPO** | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-013** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | SDRAM-PC2700 DDR | **CAPACIDAD** | | 256 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 34. Equipo PC-013**

| **EQUIPO** | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-017** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST360021A ATA | **CAPACIDAD** | | 60 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON D310 | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS XP | | | | |

**Tabla 35. Equipo PC-017**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-018** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | MAXTOR 6E04LO ATA | **CAPACIDAD** | | 40 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 2100 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 36. Equipo PC-018**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-020** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 37. Equipo PC-020**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-021** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 38. Equipo PC-021**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-022** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | SDRAM-PC2700 DDR | **CAPACIDAD** | | 256 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 39. Equipo PC-022**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-023** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | HDS728080PLAT20 ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 2100 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 40. Equipo PC-023**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-024** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ULTRA-ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 41. Equipo PC-024**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-025** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ULTRA-ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 42. Equipo PC-025**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-026** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1600 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS XP | | | | |

**Tabla 43. Equipo PC-026**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-027** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1612 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 44. Equipo PC-027**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-028** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON | **VELOCIDAD** | | 1800 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 45. Equipo PC-028**

| EQUIPO | | | | | |
|---|---|---|---|---|---|
| **SERIAL** | **PC-029** | | | | |
| **ESTADO ACTUAL** | **En Funcionamiento** | **SI** | | **Sin Funcionamiento** | |
| **DISCO DURO** | ST3802110A ATA | **CAPACIDAD** | | 80 GB | |
| **MEMORIA RAM** | DDR-SDRAM | **CAPACIDAD** | | 512 MB | |
| **TIPO PROCESADOR** | INTEL CELERON D310 | **VELOCIDAD** | | 2133 MHZ | |
| **SISTEMA OPERATIVO** | WINDOWS 7 | | | | |

**Tabla 46. Equipo PC-029**

# ANEXO F: MANUAL TÉCNICO Y DE INSTALACIÓN

## 1. Requerimientos Software de la Aplicación

### 1.1. Instalación de Java

Es necesario instalar Java en el equipo en el cual se va ejecutar la aplicación. Solo es necesario instalar el Entorno de Ejecución Java (JRE), pero si se desea puede instalarse el Java Development Kit (JDK). La versión de Java a instalar será la versión 1.6 (JRE o JDK). Se recomienda la versión de Java 1.6.18 (Java 6.0 Update 18).

### 1.2. Desactivación del Firewall

Para que no existan problemas en las comunicaciones entre los nodos de la red superpuesta P2P, se debe desactivar el firewall de cada equipo dentro de la red.

Para desactivar el firewall del equipo, abra el '*Panel de Control*', y en la sección '*Firewall de Windows*', desactívelo.

## 2. Instalación de la Aplicación

La aplicación no necesita instalarse. Solo necesita copiarse en un disco que permita la lectura y escritura de archivos, debido a que tanto JADE como JXTA, crean archivos temporales en el directorio de ejecución de la aplicación. Se recomienda copiar el directorio completo de la aplicación en la raíz de uno de los discos duros, por ejemplo en 'C:\Echo'.

La estructura del directorio quedaría así:

```
+ Echo
     EchoService.jar
     + lib
          bcprov-jdk14.jar
          commons-codec-1.3.jar
          http.jar
          iiop.jar
          jade.jar
          jade-misc.jar
          jadeTools.jar
          javax.servlet.jar
          jxta.jar
          org.mortbay.jetty.jar
```

## 3. Ejecución de la Aplicación

Después de copiar el directorio con todos los archivos a un disco que permita la lectura y escritura, ejecute el archivo `EchoService.jar`, para iniciar la aplicación.

Si la aplicación no se ejecuta automáticamente, desde la línea de comandos digite:

```
java -jar EchoService.jar
```

### 3.1. Cambiar el puerto de JADE

Por defecto JADE se ejecuta en el puerto 1099. Si desea cambiar el puerto en el cual se ejecuta JADE, desde la línea de comandos agregue la opción -jadeport <numero de puerto>, por ejemplo:

```
java -jar EchoService.jar -jadeport 1101
```

### 3.2. Cambiar el puerto de JXTA

Por defecto JXTA se ejecuta en el puerto 9701. Si desea cambiar el puerto en el cual se ejecuta JXTA, desde la línea de comandos agregue la opción -jxtaport <numero de puerto>, por ejemplo:

```
java -jar EchoService.jar -jxtaport 9702 -jadeport 1102
```

# ANEXO G: MANUAL DE USUARIO

## 1. Ejecutar la Aplicación

Ejecute el archivo `EchoService.jar`, para iniciar la aplicación.

Si la aplicación no se ejecuta automáticamente, desde la línea de comandos digite:

```
java -jar EchoService.jar
```

Aparecerá la siguiente ventana:

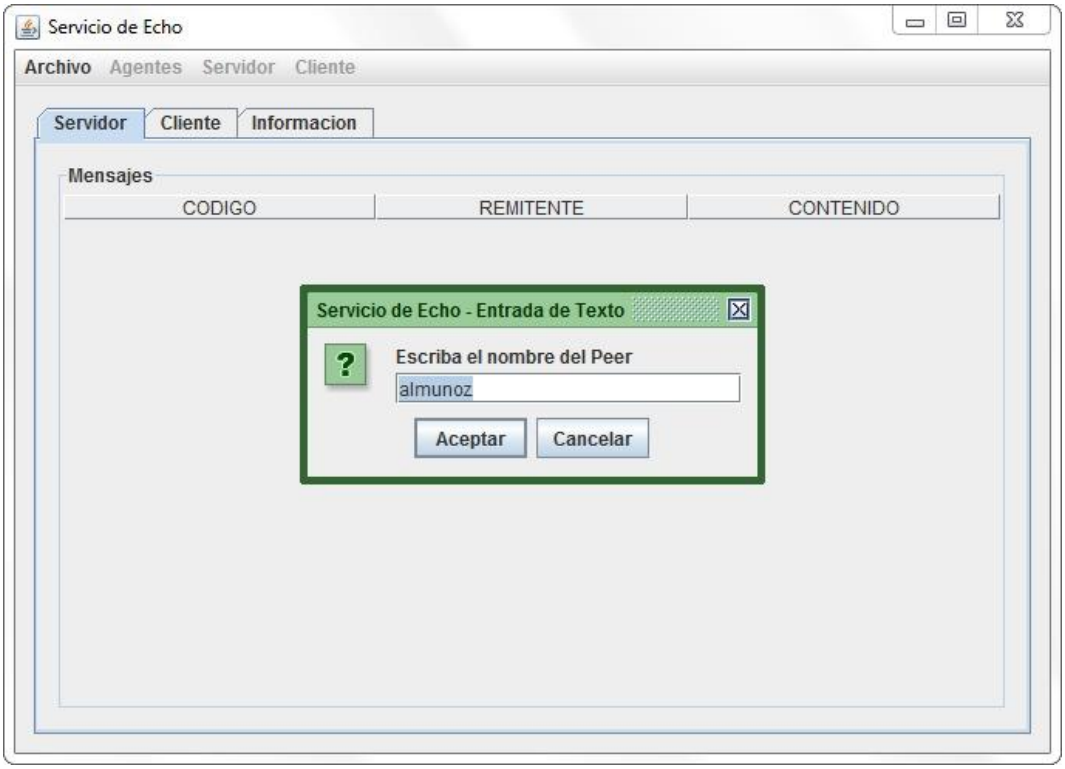


**Figura 14. Aplicación del Servicio de Eco**

Escriba el nombre del Peer, y presione el botón `Aceptar`.

Espere a que se habiliten los menús superiores: `Agentes`, `Servidor`, `Cliente`. Cuando los menús estén habilitados ya puede empezar a utilizar la aplicación.

Por defecto la funcionalidad del nodo servidor y del nodo cliente se ejecutan automáticamente al inicio de la aplicación.

## 2. Menú Servidor

El menú `Servidor` presenta la funcionalidad relacionada con el nodo servidor. El nodo servidor puede Activarse, Desactivarse y pueden eliminarse los mensajes que están actualmente en la pestaña de información del servidor.

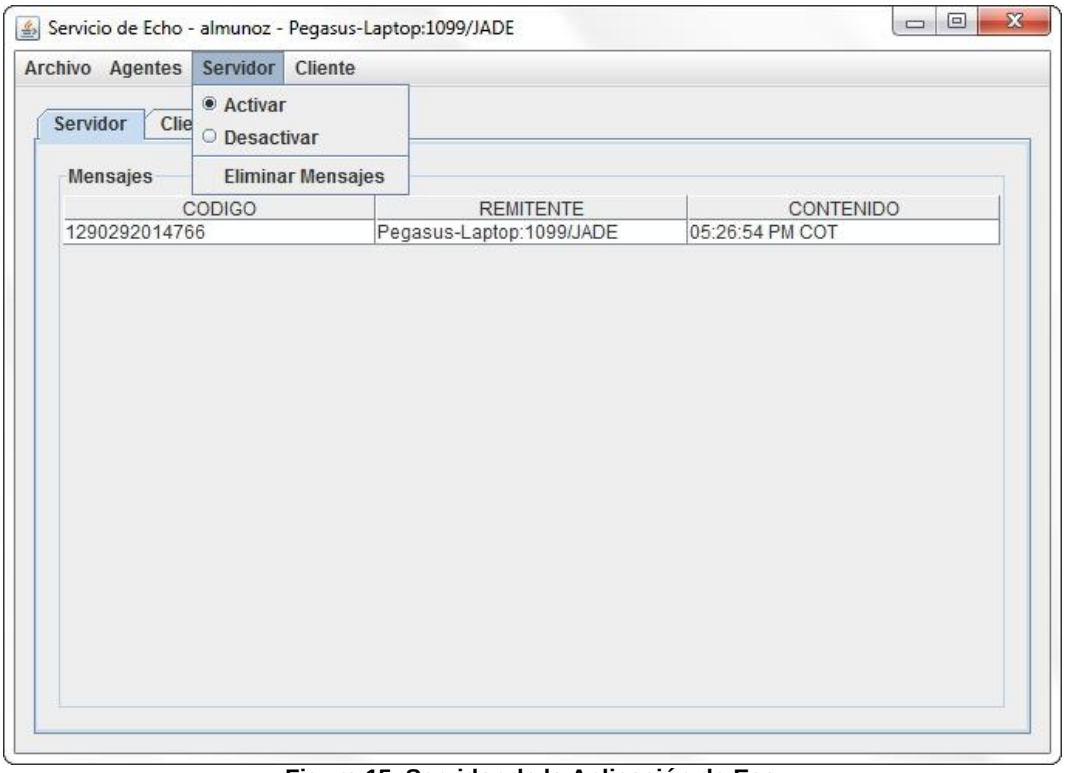


**Figura 15. Servidor de la Aplicación de Eco**

En la Figura 15, se puede ver el menú Servidor con sus respectivas opciones.

### 2.1. Activar

Esta opción activa el nodo servidor, siempre y cuando el servidor este desactivado.

### 2.2. Desactivar

Esta opción desactiva el nodo servidor, siempre y cuando el servidor este activado.

### 2.3. Eliminar Mensajes

Esta opción elimina los mensajes que se visualizan actualmente en la pestaña `Servidor`.

### 2.4. Pestaña Servidor

Muestra la información de los mensajes que recibe el nodo servidor para ser procesados.

## 3. Menú Cliente

El menú `Cliente` presenta la funcionalidad relacionada con el nodo cliente. El nodo cliente puede Activarse, Desactivarse y pueden eliminarse los mensajes recibidos, enviados y ambos, que están actualmente en la pestaña de información del cliente.

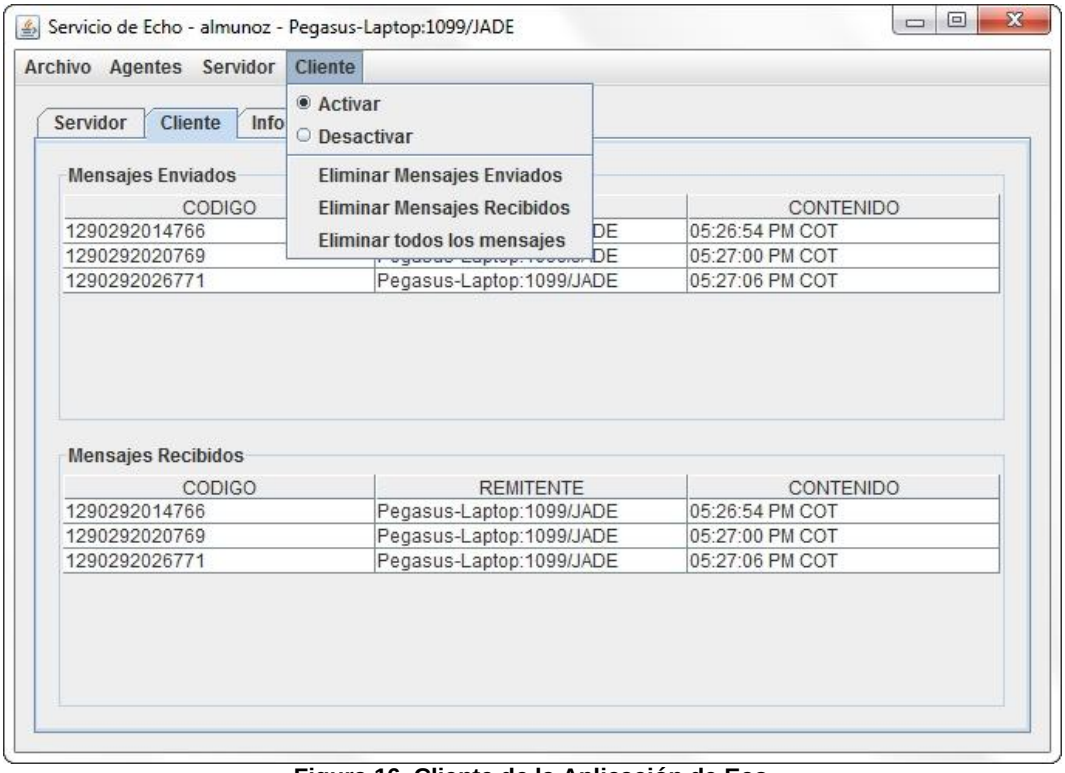


**Figura 16. Cliente de la Aplicación de Eco**

En la Figura 16, se puede ver el menú Cliente con sus respectivas opciones.

### 3.1. Activar

Esta opción activa el nodo cliente, siempre y cuando el cliente este desactivado.

### 3.2. Desactivar

Esta opción desactiva el nodo cliente, siempre y cuando el cliente este activado.

### 3.3. Eliminar Mensajes Enviados

Esta opción elimina los mensajes enviados que se visualizan actualmente en la pestaña `Cliente`.

### 3.4. Eliminar Mensajes Recibidos

Esta opción elimina los mensajes recibidos que se visualizan actualmente en la pestaña `Cliente`.

### 3.5. Eliminar todos los mensajes

Esta opción elimina los mensajes enviados y recibidos que se visualizan actualmente en la pestaña `Cliente`.

### 3.6. Pestaña Cliente

Muestra la información de los mensajes que envía y recibe el nodo cliente.

## 4. Pestaña Información

La pestaña `Información` muestra los mensajes en tiempo real, que describen las funciones que se están ejecutando actualmente en la aplicación. Los mensajes describen desde la creación de agentes, así como también la búsqueda del servicio, el tiempo de respuesta del servidor, el tiempo de procesamiento de los mensajes, entre otros.

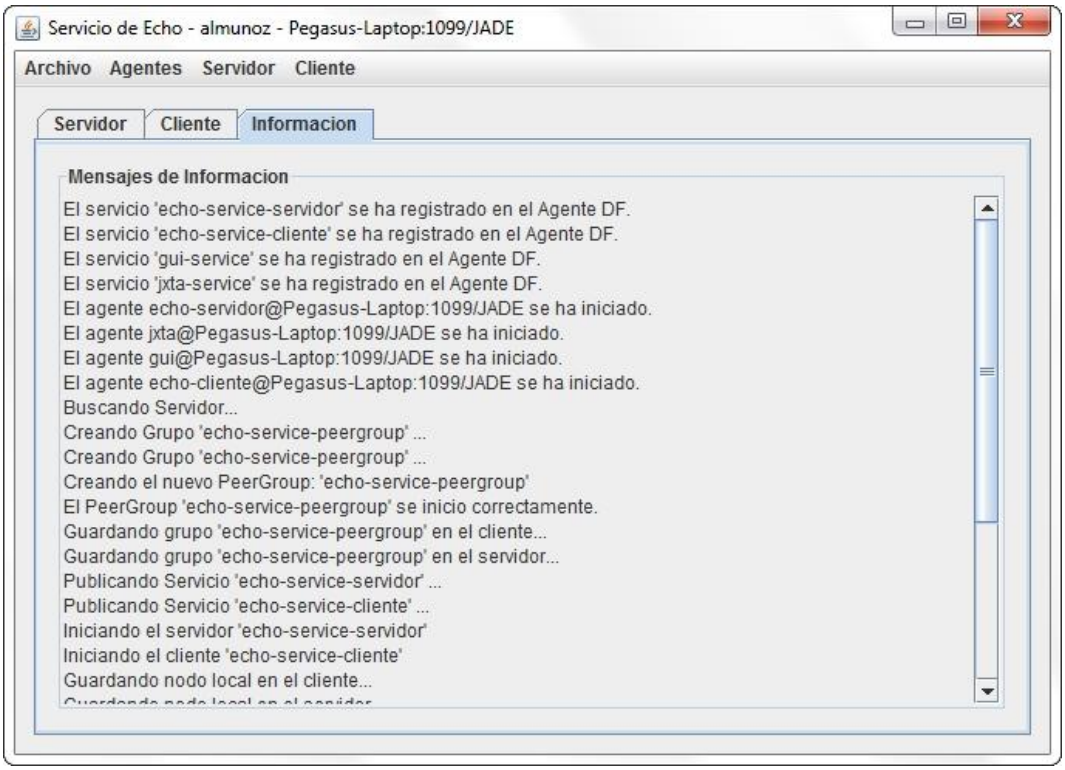


**Figura 17. Información de la Aplicación de Eco**

En la Figura 17, se muestra la pestaña de información con algunos de los mensajes que se pueden visualizar en ella.

## 5. Salir de la Aplicación

Para salir de la aplicación, puede cerrar la ventana o desde el menú `Archivo`, seleccione la opción `Salir`, como se muestra en la Figura 18.

Se le preguntará si desea salir de la aplicación, si desea salir presione el botón `Aceptar`, si no desea salir, presione el botón `Cancelar`. En la Figura 19, se puede ver la pantalla de salida de la aplicación.

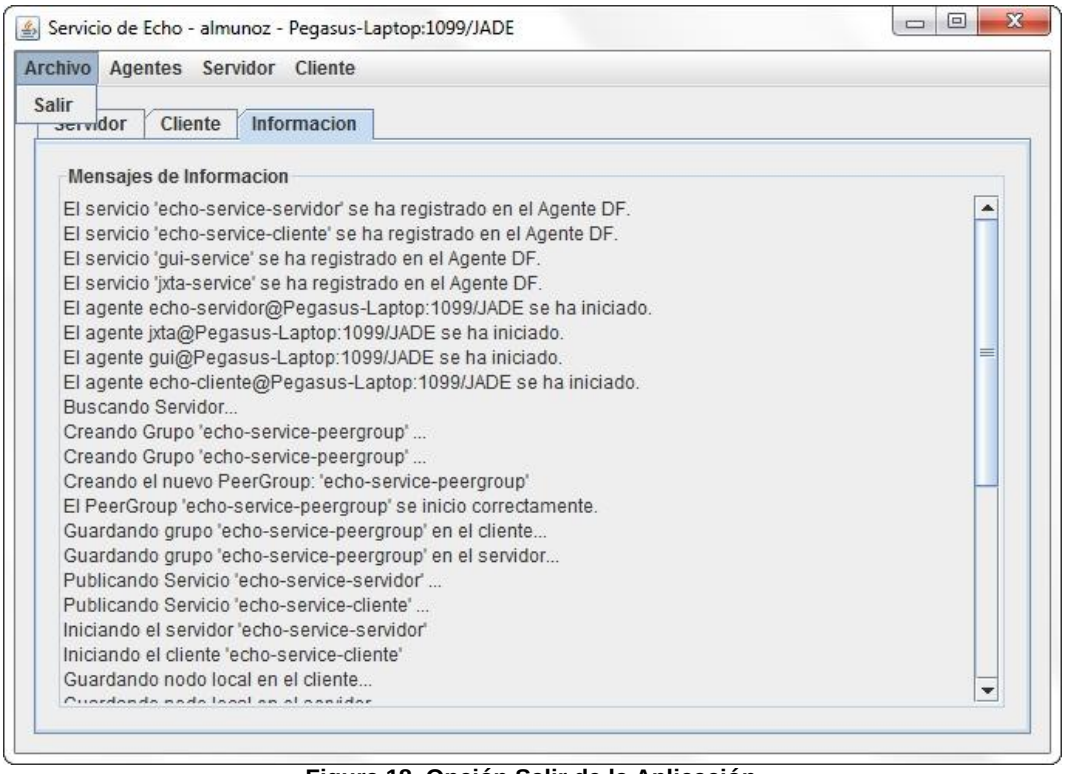


**Figura 18. Opción Salir de la Aplicación**

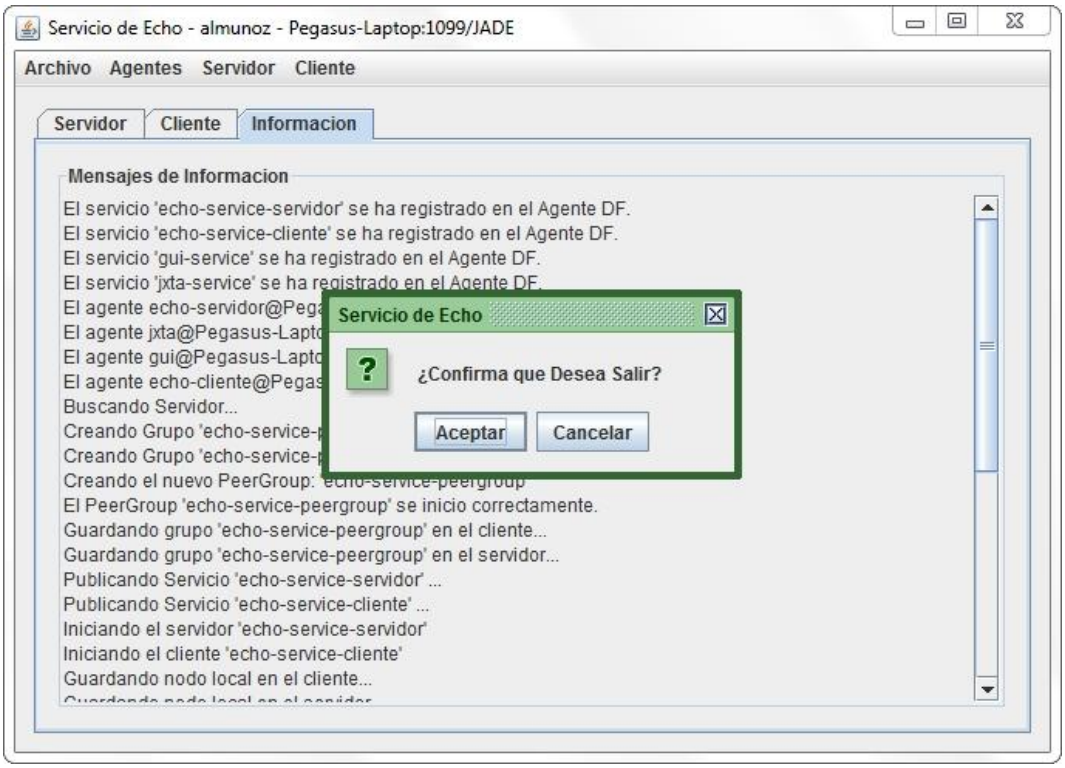


**Figura 19. Pantalla de Salida de la Aplicación**

# REFERENCIAS BIBLIOGRÁFICAS

[1] A. Finkelstein y J. Kramer, "Software Engineering: A Roadmap", 2009.

[2] AUP. *The Agile Unified Process v1.1*, 2006. Disponible: http://www.ambysoft.com/unifiedprocess/agileUP.html

[3] R. Pressman, *Ingeniería del Software: Un enfoque Práctico*, 5ª Edición. Capítulo 2 ed.: McGraw Hill, 2002.

[4] JXTA. *JXTA Community Project*, 2010. Disponible: http://www.jxta.org/

[5] J. D. Gradecki, *Mastering Jxta: Building Java Peer-to-Peer Applications*: John Wiley & Sons, Inc., 2002.

[6] B. J. Wilson, *JXTA Book*: New Riders Publishing, 2002.

[7] JADE. *Java Agent Development Framework*, 2010. Disponible: http://jade.tilab.com/

[8] F. L. Bellifemine, G. Caire, y D. Greenwood, *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*: John Wiley & Sons, Ltd., 2007.