
# TREBALL DE FI DE CARRERA

**TÍTOL: Sistema de localización multimedia distribuido**

**AUTOR: Alberto José González Cela**

**DIRECTOR: Toni Oller Arcas**

**DATA: 20 de enero de 2005**

**Títol:** Sistema de localización multimedia distribuido

**Autor:** Alberto José González Cela

**Titulació:** Ingeniería Técnica de Telecomunicaciones

**Especialitat:** Telemática

**Pla:** 2000

**Director:** Toni Oller Arcas

**Departament:** Ingeniería Telemática

**Vist i plau,**

**Director del TFC**

**Registre:**

**Título:** Sistema de localización multimedia distribuido

**Autor:** Alberto José González Cela

**Director:** Toni Oller Arcas

**Fecha:** 20 de enero de 2005

**Resumen**

El proyecto se sitúa en un contexto donde hay multitud de recursos multimedia en una red heterogénea y un usuario quiere encontrarlos para poder reproducirlos. Por tanto, se debe definir un mecanismo para permitir al usuario localizar dichos recursos.

El objetivo que se persigue en este proyecto es el estudio de diferentes tecnologías para implementar una arquitectura de localización de recursos multimedia y por lo tanto de acceso a éstos.

Se contemplan 2 arquitecturas: centralizada y distribuida Peer to Peer (P2P).

En este proyecto se han estudiado principalmente 2 tecnologías que se implementarán en un entorno Java: Servicios Web para el escenario centralizado y JXTA para el entorno distribuido.

Para lograr nuestro objetivo se ha hecho un estudio de ambas tecnologías así como de sus componentes. Posteriormente se han realizado diferentes pruebas para comprobar su funcionamiento.

Una vez se tiene claro el funcionamiento de ambas tecnologías se ha diseñado la arquitectura centralizada y distribuida. La implementación del diseño se ha llevado a cabo de manera secuencial, de modo que a partir de un programa sencillo se han añadido funcionalidades.

Finalmente, se ha conseguido implementar una arquitectura centralizada en la que un usuario puede consultar, publicar y eliminar recursos y una arquitectura distribuida en la que un usuario puede descubrir a otros usuarios, crear grupos, publicar sus servicios, encontrar nuevos servicios y hablar con otros usuarios.

**Summary**

The context of this project is a heterogeneous network where a user wants to find multimedia resources in order to access to them.

The main goal of this project is the study of various technologies in order to implement a searching system of media resources.

We have seen 2 architectures: centralized and distributed (Peer to Peer).

In this project we have studied 2 technologies that will be implemented in a Java environment: Web Services for the centralized view and JXTA for the distributed view.

In order to achieve our objective, we have made a study of both technologies already mentioned and all of their components. After that, we ran some tests to ensure they worked properly.

Once we knew how the technologies worked, we designed the centralized and distributed architecture that we wanted to implement. The implementation has been sequential, that is, we started with a simple program and then we added new functionalities to it.

Finally, we achieved the implementation of a centralized architecture that enables us to search, publish and delete resources. Furthermore, we successfully implemented a distributed architecture where a user can discover other users, organize groups, publish his services, find new services and communicate with other peers.

# ÍNDICE

# CAPÍTULO 1. INTRODUCCIÓN

El estudio llevado a cabo en este trabajo viene motivado por un proyecto iniciado por la Fundación I2Cat.

I2Cat es una fundación privada cuya actividad es la investigación, innovación y desarrollo en el mundo de Internet y la transmisión digital de alta calidad de contenidos audiovisuales en banda ancha.

I2Cat pretende ser un foro para instituciones, organizaciones y empresas interesadas en formar parte de un proyecto global con el objetivo de desarrollar las infraestructuras y servicios de red de banda ancha de Cataluña.

La actividad principal de I2Cat para 2004 - 2005 gira entorno al proyecto que motiva el estudio presentado, denominado “Proyecto Integrado”.

El “Proyecto Integrado” pretende crear un sistema mediante el cual un usuario puede acceder desde su terminal (fijo o móvil) a contenido audiovisual de alta calidad a partir de la integración de diferentes módulos que interaccionan entre sí de manera transparente para el usuario. El proyecto configura una red de entornos flexibles para la creación, edición y distribución de contenidos audiovisuales sobre IP para todo el mundo.

Para lograr el objetivo planteado, el proyecto centra sus esfuerzos en la integración de servicios, localización dinámica de recursos, heterogeneidad de los servicios, transparencia de la red, almacenamiento, transporte y procesado de contenido multimedia, transcodificación de contenidos, topología dinámica de la red, redes hechas a medida, gestión distribuida de los recursos de red y calidad de servicio. En definitiva, define nuevos modelos de red y nuevos modelos de negocio.

El trabajo realizado se centra en el proceso de localización de los recursos multimedia, englobado en el módulo de transcodificación de protocolos [Anexo 1].

## 1.1. Contexto

El proyecto se plantea la situación en la que un usuario desea acceder a un recurso multimedia que se encuentra en una red heterogénea y en la que existen multitud de recursos en ella. Por lo tanto se debe definir cómo se va a realizar la búsqueda de dichos recursos.

El proyecto pretende resolver la problemática que surge cuando un usuario desea reproducir un recurso multimedia y éste debe ser localizado en la red.

Para llegar a una solución, en primer lugar se deben plantear las diferentes posibilidades disponibles a la hora de localizar un recurso. En este proyecto se estudian 2 arquitecturas de búsqueda:

- Centralizada
- Distribuida (Peer to peer, P2P)

La búsqueda centralizada se basa en hacer peticiones a una máquina concreta que ofrece el servicio de localización. Ésta accede a un almacén de datos con la información de los recursos multimedia existentes en el sistema. La arquitectura centralizada, responde al esquema clásico de *n* clientes y 1 servidor (consumidor – productor). De ésta manera, un usuario es capaz de acceder fácilmente a una base de datos determinada para obtener toda la información de un recurso (meta información) que desea visualizar. No obstante, esta arquitectura plantea numerosos inconvenientes en cuanto a tolerancia de fallos, escalabilidad, etc., lo que motiva el estudio de la segunda arquitectura comentada, la distribuida.

Para llevar a cabo el estudio de esta arquitectura, se usarán Servicios Web a modo de interfaz de abastecimiento y representación del servicio junto con un registro UDDI para la publicación de estos servicios y por tanto puedan ser consultados por los usuarios.

Los Servicios Web, en término general, son servicios ofrecidos por una aplicación a otra aplicación. Los Servicios Web son descripciones de servicios que pueden ser publicados, localizados e invocados sobre la red. Utilizan WSDL para describir el servicio detalladamente (permite saber qué debe recibir y qué retorna un determinado servicio) con lo que se pueden implementar clientes.

La arquitectura distribuida, P2P, rompe con el esquema anterior, proponiendo un modelo en el que cada usuario del sistema puede ser cliente y servidor (productor y consumidor) a la vez.

En este tipo de arquitectura, los servicios no se encuentran en un solo servidor sino que están repartidos por la red. Esto supone grandes ventajas tanto en la reducción de congestión del tráfico en enlaces como en el equilibrio de carga entre servicios productores y consumidores. La función de los usuarios es publicar, de manera automática, sus servicios para que un usuario cualquiera los pueda localizar y usar en cualquier momento y en cualquier parte. Las posibilidades que ofrece este modelo en cuanto a robustez, tolerancia a fallos y escalabilidad son enormes. Sin embargo, el problema se encuentra en la heterogeneidad que conforma una estrategia distribuida. P2P intenta aprovechar de una forma más eficiente el acceso a la información (actualmente se produce mucha información que fluye por la red y el acceso a información útil, en tiempo real, cada vez es más difícil), el ancho de banda y la capacidad de cálculo de las redes.

Para construir aplicaciones P2P se dispone del framework JXTA, el cual engloba protocolos que permiten la comunicación y colaboración entre peers (nodos) conectados a la red. Se basa en Java y XML.

JXTA surge de un equipo de trabajo de Sun MicroSystems Inc,. con la finalidad de solventar ciertos problemas que aparecen el los sistemas distribuidos P2P. Dichos problemas son:

- Descripción de los servicios que puede proporcionar un peer y cómo pueden ser utilizados.
- Representación de la meta información. Cuando hablamos de meta información nos referimos a la información que describe un recurso multimedia. Para lograrlo se usará el framework MPEG 21.

MPEG 21 define principalmente cuáles son los participantes en una transacción de bienes (en nuestro caso, datos, concretamente Objetos Digitales) a través de una gran variedad de redes y dispositivos. Su base se sustenta en la definición del denominado Objeto Digital, el cuál permite especificar aspectos tales como la descripción del contenido del recurso (meta información), derechos de propiedad intelectual y utilización que cada usuario puede hacer del recurso multimedia en cuestión.

Por lo que respecta al almacenamiento de la información, se estudian 2 tipos de bases de datos:

- Relacionales (en concreto MySQL)
- XML (en concreto Xindice)

A pesar de las ventajas en términos de velocidad y escasez de recursos consumidos por una base de datos MySQL, también resulta interesante estudiar bases de datos XML (como Xindice) ya que nos basaremos en MPEG-21 para la representación de datos. Éste describe los Objetos Digitales mediante DIDL (Digital Item Description Language) y para ello utiliza XML.

## 1.2. Objetivos

Como ya se ha comentado en el contexto, este estudio pretende solventar la problemática de la localización de recursos multimedia en una red.

Para lograrlo, nuestros objetivos son el estudio de diferentes estrategias para implementar el servicio de localización. Las dos estrategias a estudiar son: centralizada y distribuida.

Uno de los objetivos como paso previo al desarrollo del estudio es el de definir detalladamente un prototipo, especificando las funcionalidades que ofrece.

Con el fin de poner en práctica el prototipo especificado, se implementará una herramienta que permita la localización de servicios de media en base a una arquitectura centralizada y distribuida.

Se debe crear una plataforma de trabajo basada en módulos heterogéneos que sean fácilmente integrables con otras partes del Proyecto Integrado.

Otro objetivo será el de crear una interfaz de usuario para acceder a la herramienta de localización.

En ambas arquitecturas, se debe solventar la manera de publicar nuestro servicio: localización de recursos de media. Para ello se estudiarán las diferentes herramientas que permiten llevar a cabo esta labor.

Una vez estudiadas e implementadas ambas arquitecturas, se extraerán las conclusiones pertinentes, incluyendo ventajas y desventajas de cada una.

# CAPÍTULO 2. ESPECIFICACIÓN

El objetivo principal del proyecto es definir un prototipo de localización de recursos de media. Dicho prototipo permitirá a los usuarios consultar, introducir y eliminar recursos de media en el sistema.

Cuando se habla de recursos de media nos referimos a:

- Audio bajo demanda: discos completos o canciones que se encuentren en el sistema.
- Video bajo demanda: películas completas o video clips que se encuentren en el sistema.
- Videoconferencia: videoconferencias / multiconferencias programadas en el sistema.
- Televisión: canales de televisión disponibles en la red.
- Radio: emisoras de radio disponibles en la red.

El módulo de localización, una vez encontrados los recursos que el usuario desea reproducir, pasará los datos pertinentes a un módulo de provisión de media para que éste interactúe con los diferentes procesadores de media con el fin de llevar al usuario el flujo multimedia adecuado.

A continuación, se describen las funcionalidades que se han especificado para el Proyecto Integrado [9] (dentro del cual está el módulo del sistema de localización al que se refiere el estudio realizado en este trabajo). El módulo de localización se encuentra dentro del módulo de transcodificación de protocolos.

Uno de los objetivos planteados en el Proyecto Integrado es la construcción de un prototipo de gestión, localización y provisión de recursos de media. Dicho prototipo permitirá a los usuarios consultar y acceder a servicios multimedia: audio bajo demanda, video bajo demanda, videoconferencia, televisión y radio ubicados en procesadores de media. Sobre esta plataforma se podrán implementar servicios de valor añadido como intercambio de mensajes y archivos entre los usuarios pertenecientes a un grupo.

## 2.1. Funcionalidades

Las funcionalidades se pueden ver con más detalle en el Anexo 7. A continuación se presenta un resumen de ellas.

### 2.1.1. Localización de recursos de media

El usuario puede obtener un listado de recursos multimedia según unos criterios de búsqueda. Los criterios de búsqueda se pueden introducir manualmente (especificar campo de búsqueda y valor) o se puede definir una

búsqueda automática asociada al perfil de usuario que se disponga dentro del sistema (por ejemplo, determinar que siempre se obtengan recursos de calidad máxima).

También se puede ver la disponibilidad de los servidores y hacer una valoración del recurso obtenido.

### 2.1.2. Ejecución de recursos de media

El usuario puede ejecutar los recursos localizados y tener un control sobre ellos. Puede hacer: play, stop, pause y control de volumen.

### 2.1.3. Gestión

Un usuario puede gestionar su perfil de búsqueda de recursos, según un mecanismo manual y otro automático.

- Automático: El sistema proporciona al usuario el recurso que mejor se adapte a las características previamente establecidas

- Manual: El usuario selecciona unos campos de búsqueda y el sistema le muestra todos los recursos disponibles según los campos completados.

### 2.1.4. Perfiles de usuario

El sistema define 3 perfiles de usuario: usuario registrado, usuario registrado en un grupo y administrador de grupo.

Las funcionalidades ofrecidas por el sistema se determinan en función del perfil de usuario.

## 2.2. Especificación formal

El sistema de localización es uno de los módulos que componen el Transcodificador de protocolos. Sin embargo, éste está formado por otros submódulos como el de autenticación, calidad de servicio, provisión de servicios, etc.

A continuación se muestran diferentes diagramas de casos de uso[1] que permiten ver con mayor claridad las funcionalidades que ofrece el sistema. También se describe la función de cada módulo y se las acciones que puede llevar a cabo cada usuario en función de su perfil.

[1] Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

### 2.2.1. Módulo de acceso al sistema

En el siguiente diagrama se puede ver que el sistema consta de 3 perfiles de usuario: Administrador de Grupo (ARG), Usuario Registrado en Grupo (URG) y Usuario (UR).

El ARG tiene funcionalidades de administración de grupo y de recursos multimedia. En cambio, el URG y UR no pueden desempeñar tareas de administración, pero sí de acceso a recursos multimedia.

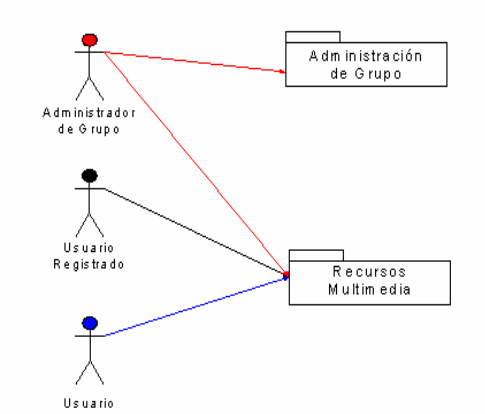



**Fig. 2.1** Diagrama casos de uso de acceso al sistema.

### 2.2.2. Módulo de localización de recursos

Este módulo tiene la responsabilidad de encontrar recursos multimedia (audio, video, videoconferencia, televisión, radio) a partir de unos parámetros de búsqueda. A continuación se muestra el acceso a las diferentes funcionalidades según el perfil de usuario con el que se accede al sistema.

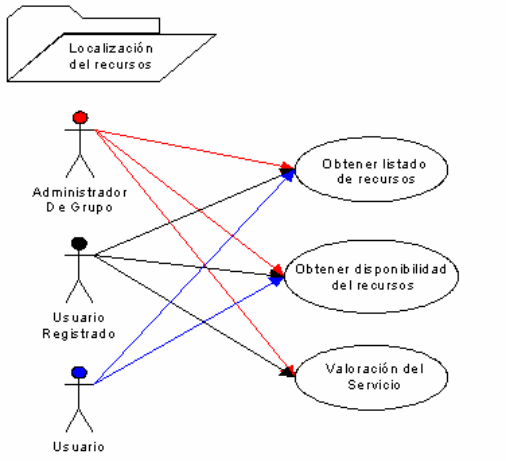



**Fig. 2.2** Diagrama casos de uso de localización de recursos.

Seguidamente se detalla el proceso a seguir para obtener el listado de recursos multimedia desde el punto de vista de usuario.

**Tabla 2.1**

| | | | | | |
|---|---|---|---|---|---|
| FUNCIONALIDAD | Obtención de listado de recursos multimedia. | | | | |
| Propósito | Permitir al Administrador de grupo, usuario registrado o usuario obtener una lista de aquellos recursos multimedia que están disponibles / no disponibles. | | | | |
| Precondiciones | Tener configurado correctamente los parámetros de búsqueda según modo manual o automático. | | | | |
| Poscondiciones | Posibilidad de acceso a los recursos multimedia disponibles. | | | | |
| Actores | Administrador del Grupo (AG), Usuario Registrado en el Grupo (URG), Usuario (UR) | | | | |
| DESCRIPCIÓN | Acción del Actor | | | Respuesta del Sistema | |
| | Paso | Actor | Acción | Paso | Respuesta |
| | 1 | AG / URG / UR | Solicitar al sistema comenzar el proceso. | 2 | El sistema retorna un listado de recursos multimedia :<br>- disponibles<br>- no disponibles |
| ALTERNATIVAS | Paso | Condición que provoca una respuesta alternativa | | Acción | |
| | 3 | El AG / URG / UR cancela el proceso | | Fin de la funcionalidad | |
| | 4 | El AG / URG / UR finaliza el proceso | | Fin de la funcionalidad | |
| | 5 | El sistema detecta que los datos no son correctos | | Se informa al AG / URG / UR y finaliza la funcionalidad. También se puede valorar el recurso o servicio obtenido. | |

Esquema de interacción con el sistema para la obtención del listado:


**Fig. 2.3** Obtención de listado de recursos multimedia.

### 2.2.3. Módulo de localización de procesadores de media

Este módulo se encarga de encontrar diferentes procesadores de media (Asterisk, Darwin, VideoLAN, etc.). A continuación se muestra el acceso a las diferentes funcionalidades según el perfil de usuario con el que se accede al sistema.

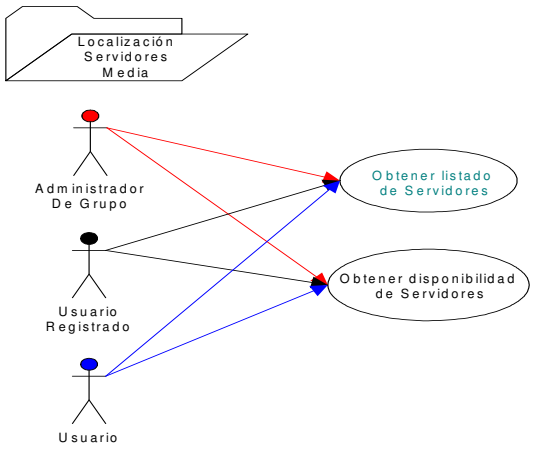



**Fig. 2.4** Diagrama casos de uso de localización de procesadores de media.

### 2.2.4. Módulo de provisión de servicios

Este módulo especifica las acciones que puede llevar a cabo un usuario sobre cada uno de los recursos multimedia.

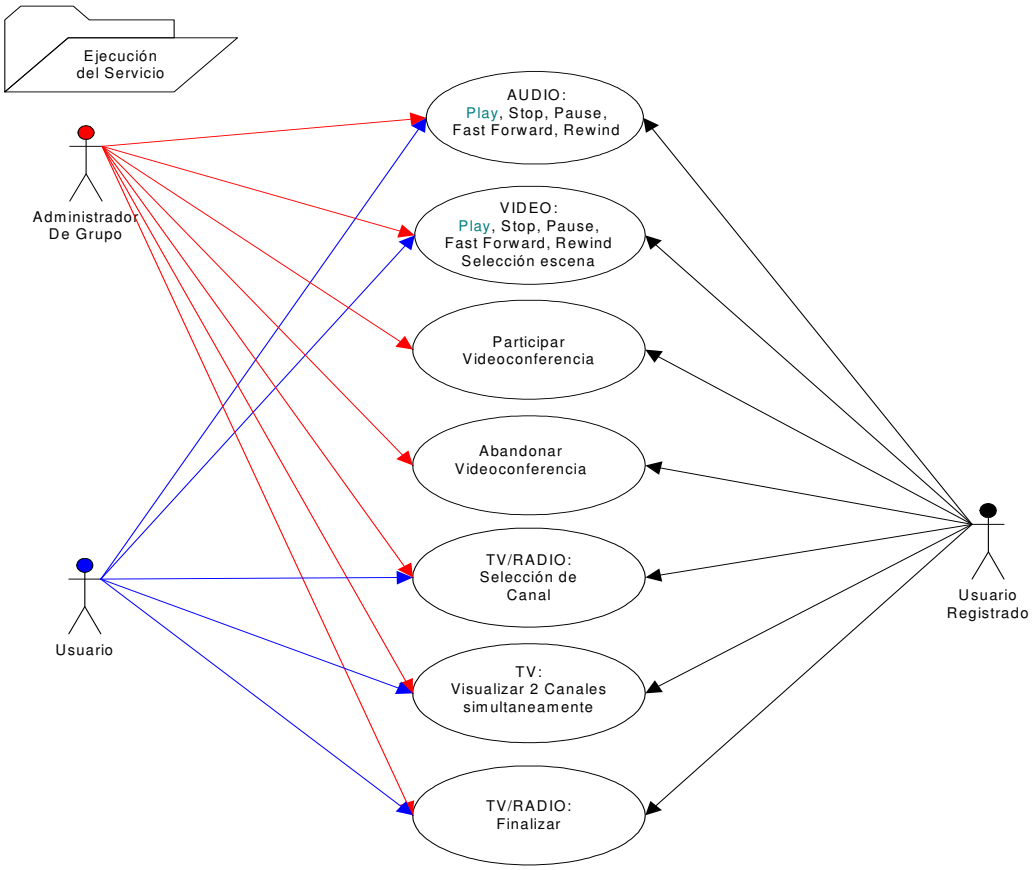



**Fig. 2.5** Diagrama casos de uso de provisión de servicios.

Seguidamente se detalla el proceso a seguir para reproducir un recurso multimedia desde el punto de vista de usuario.

**Tabla 2.2**

<table>
<tr><td>FUNCIONALIDAD</td><td colspan="5">Reproducción (Play) de audio / video.</td></tr>
<tr><td>Propósito</td><td colspan="5">Permitir a un usuario reproducir cualquier fuente de audio / video que se encuentre disponible en el sistema.</td></tr>
<tr><td>Precondiciones</td><td colspan="5">Haber localizado el recurso previamente y que éste se encuentre disponible.</td></tr>
<tr><td>Poscondiciones</td><td colspan="5">Al final de la reproducción se puede valorar tanto el recurso como el servicio obtenido en caso de ser usuario registrado o administrador.</td></tr>
<tr><td>Actores</td><td colspan="5">Administrador del Grupo (AG), Usuario Registrado en el Grupo (URG), Usuario (UR)</td></tr>
<tr><td rowspan="3">DESCRIPCIÓN</td><td colspan="3">Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>Paso</td><td>Actor</td><td>Acción</td><td>Paso</td><td>Respuesta</td></tr>
<tr><td>1</td><td>AG / URG / UR</td><td>Solicitar al sistema comenzar el proceso de reproducción de audio / video.</td><td>2</td><td>El sistema retorna un flujo multimedia correspondiente al recurso solicitado.</td></tr>
<tr><td rowspan="5">ALTERNATIVAS</td><td>Paso</td><td colspan="2">Condición que provoca una respuesta alternativa</td><td colspan="2">Acción</td></tr>
<tr><td>3</td><td colspan="2">El AG / URG / UR cancela el proceso</td><td colspan="2">Fin de la funcionalidad</td></tr>
<tr><td>4</td><td colspan="2">El AG / URG / UR finaliza el proceso</td><td colspan="2">Fin de la funcionalidad</td></tr>
<tr><td>5</td><td colspan="2">El sistema detecta que los datos no son correctos</td><td colspan="2">Se informa al AG / URG / UR y finaliza la funcionalidad. También se puede valorar el recurso o servicio obtenido.</td></tr>
</table>

Esquema de interacción con el sistema para la reproducción de video / audio bajo demanda:

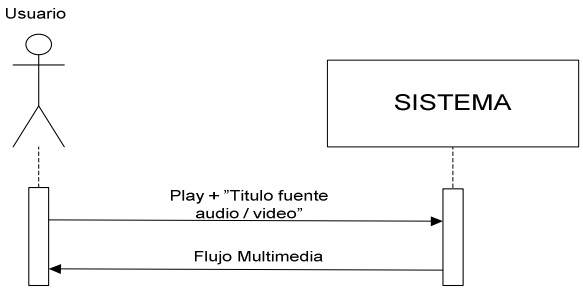



**Fig. 2.6** Reproducción de un recurso de audio/video.

### 2.2.5. Módulo de gestión

Este módulo especifica las acciones que puede llevar a cabo cada tipo de usuario por lo que respecta a la gestión de servicios, grupos o usuarios.

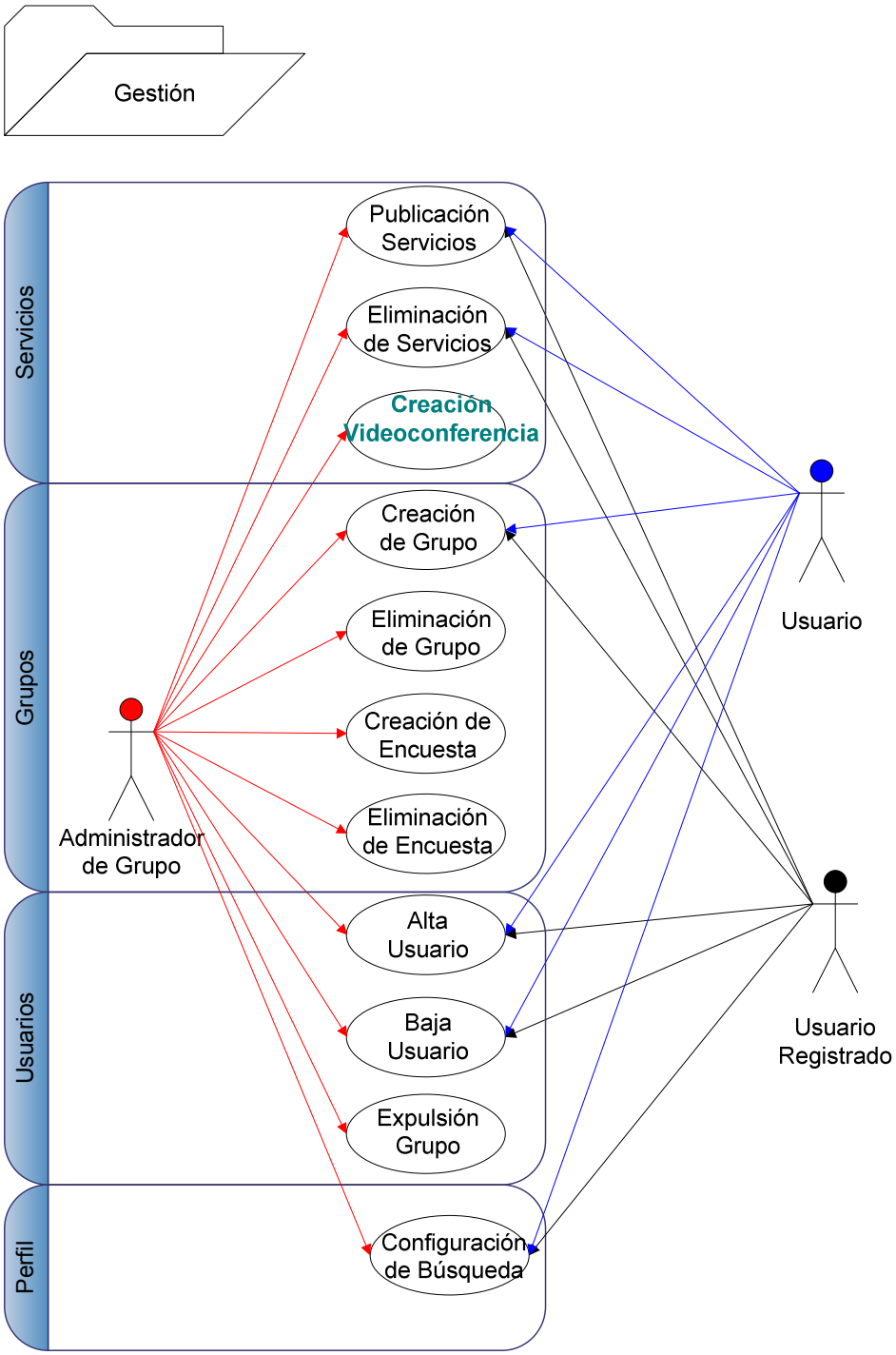



**Fig. 2.7** Acciones que puede llevar a cabo cada tipo de usuario.

Seguidamente se detalla el proceso a seguir para crear una videoconferencia desde el punto de vista de usuario.

**Tabla 2.3**

<table>
<tr><td>FUNCIONALIDAD</td><td colspan="5">Crear una videoconferencia</td></tr>
<tr><td>Propósito</td><td colspan="5">Permitir que un administrador de un grupo o usuario registrado al grupo pueda crear una videoconferencia</td></tr>
<tr><td>Precondiciones</td><td colspan="5">Estar identificado como administrador del grupo o estar registrado en el grupo, tener al menos a un participante en la videoconferencia.</td></tr>
<tr><td>Poscondiciones</td><td colspan="5">El grupo tendrá una nueva videoconferencia a la cual se podrá añadir usuarios registrados al grupo</td></tr>
<tr><td>Actores</td><td colspan="5">Administrador del Grupo (AG) o Usuario Registrado en el Grupo (URG)</td></tr>
<tr><td rowspan="5">DESCRIPCIÓN</td><td colspan="3">Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>Paso</td><td>Actor</td><td>Acción</td><td>Paso</td><td>Respuesta</td></tr>
<tr><td>1</td><td>AG o URG</td><td>Solicitud de una nueva videoconferencia</td><td>2</td><td>El sistema solicita información de la videoconferencia:<br>- Tema<br>- Fecha<br>- Hora<br>- Número de participantes<br>- Duración</td></tr>
<tr><td>3</td><td>AG o URG</td><td>Introducción de los datos y envío de los datos al sistema</td><td>4</td><td>Validación de los datos, se informa de que el proceso se ha realizado con éxito y se da la posibilidad e añadir una nueva videoconferencia.</td></tr>
<tr><td>5</td><td>AG o URG</td><td>Nueva videoconferencia</td><td>6</td><td>Volver al paso 2</td></tr>
<tr><td rowspan="5">ALTERNATIVAS</td><td>Paso</td><td colspan="2">Condición que provoca una respuesta alternativa</td><td colspan="2">Acción</td></tr>
<tr><td>3</td><td colspan="2">El AG o URG cancela el proceso</td><td colspan="2">Fin de la funcionalidad</td></tr>
<tr><td>4</td><td colspan="2">Datos erróneos</td><td colspan="2">Se avisa del acontecimiento y volver al paso 2</td></tr>
<tr><td>5</td><td colspan="2">No mas videoconferencia</td><td colspan="2">Fin de la Funcionalidad</td></tr>
</table>

Esquema de interacción con el sistema para la creación de una videoconferencia:

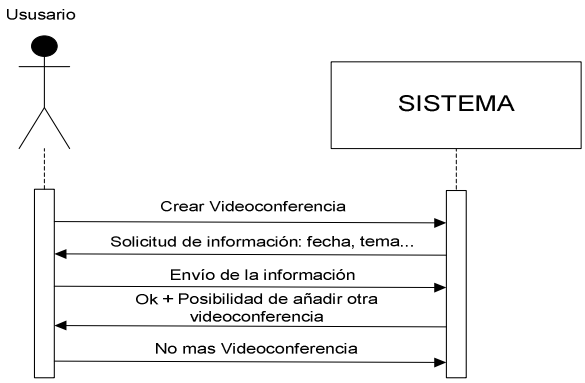



**Fig. 2.8** Creación de una videoconferencia.

### 2.2.6. Módulo de funcionalidades de grupo

Este módulo especifica las acciones que puede llevar a cabo cada tipo de usuario dentro de un grupo.

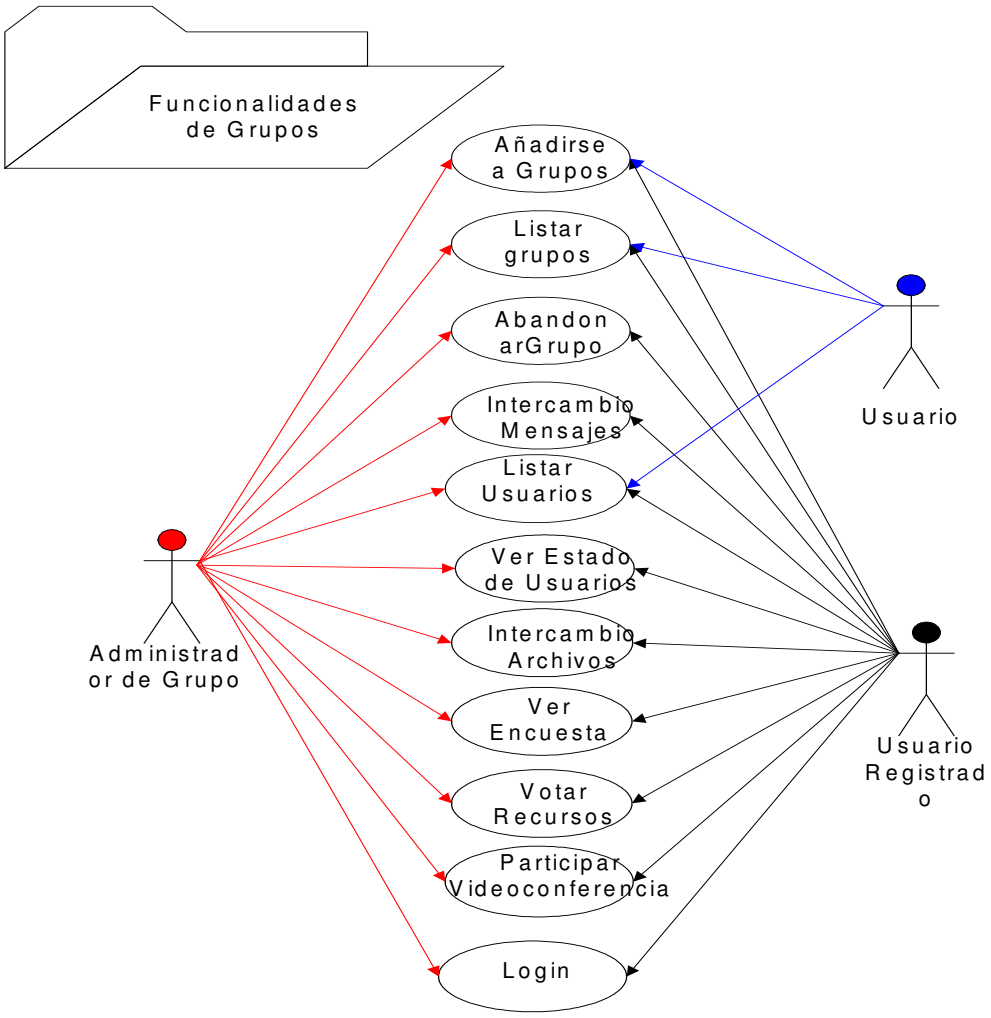



**Fig. 2.9** Funcionalidades de grupo.

### 2.2.7. Módulo de información del sistema

Este módulo especifica la información del sistema accesible para cada usuario en función del perfil que tengan.

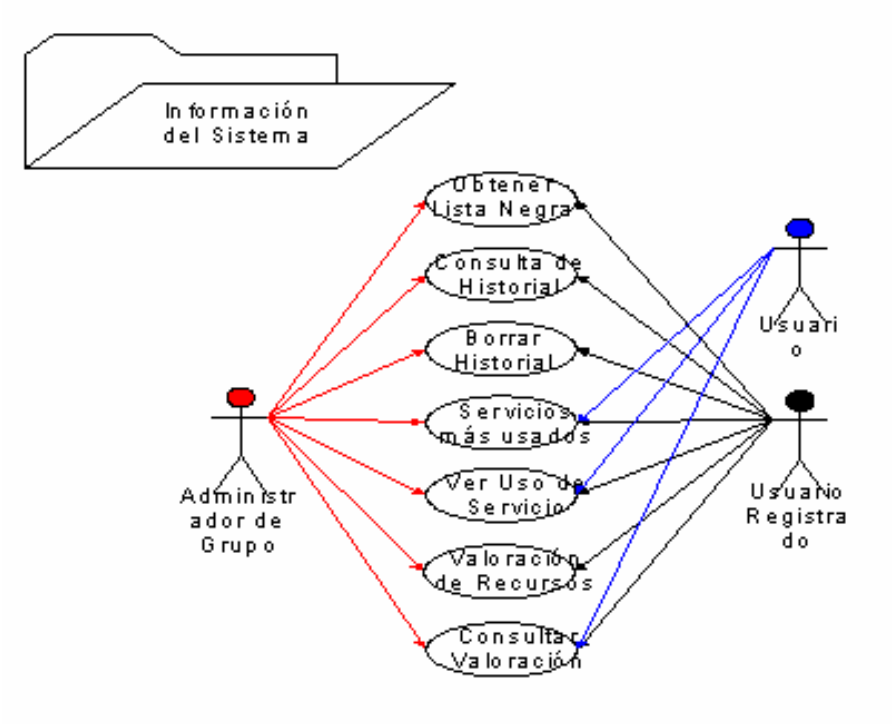



**Fig. 2.10** Diagrama de información del sistema.

# CAPÍTULO 3. ARQUITECTURA

A continuación se muestra un esquema diseñado para el Proyecto Integrado en el que se describen las interfaces a usar entre los diferentes módulos que se implementarán. En azul aparecen el módulo de servicio de localización y el módulo de provisión de servicios.

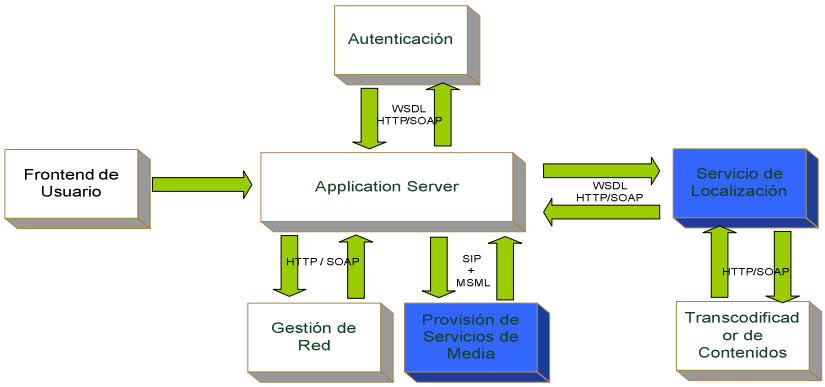



**Fig. 3.1** Módulos heterogéneos propuestos para el Proyecto Integrado.

Secuencia de interacción entre módulos heterogéneos en la localización de un recurso:

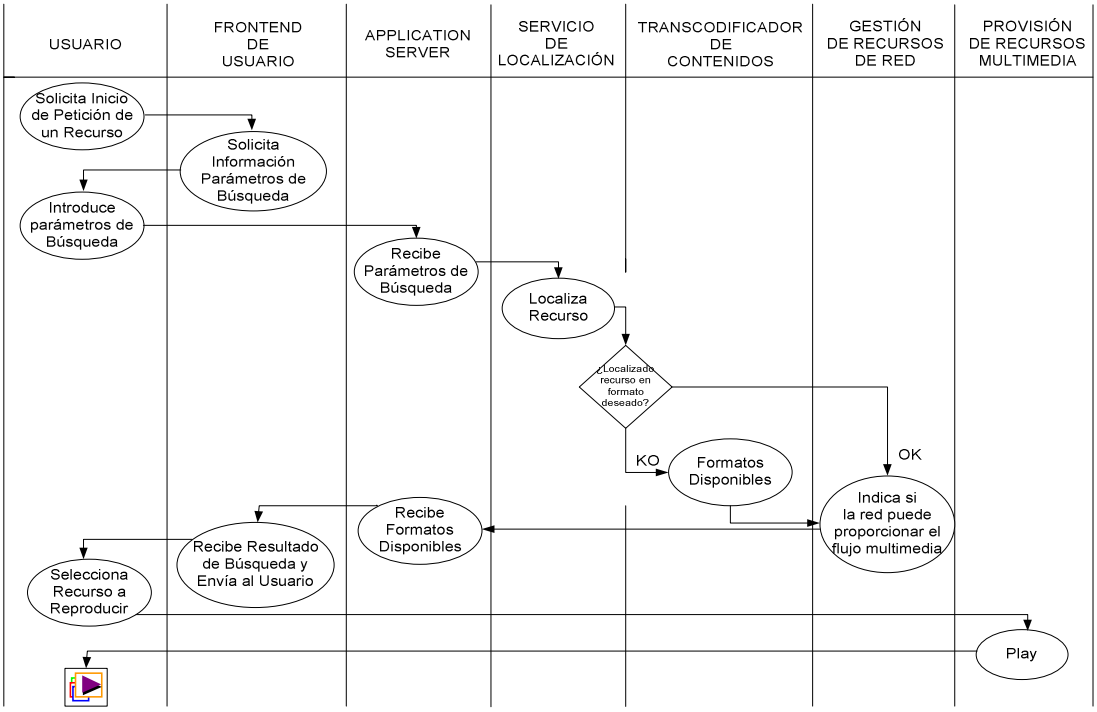
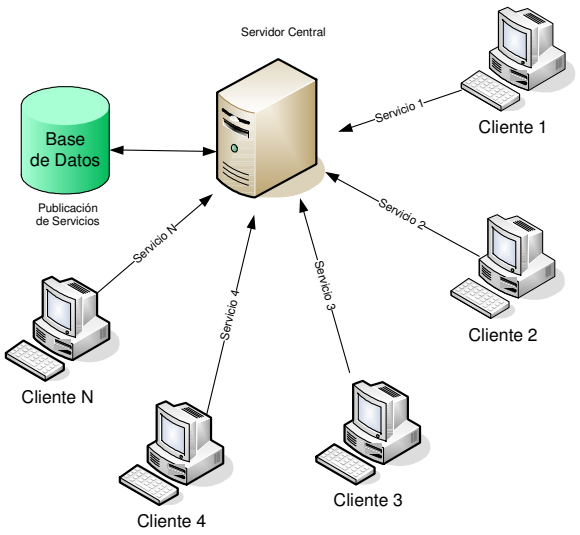



**Fig. 3.2** Secuencia de localización de un recurso.

EL servicio de localización tiene como rol fundamental la búsqueda y gestión de recursos multimedia. Se proponen dos alternativas para el sistema de localización: una arquitectura centralizada y una distribuida.

## 3.1. Arquitectura centralizada

La **estrategia centralizada** se basa en tener un conjunto indefinido de clientes que acceden a una única máquina para consultar y localizar dónde se encuentra el recurso deseado y en caso de estar disponible, obtenerlo.




Esta arquitectura presenta ventajas a la hora de simplificar la búsqueda de un recurso disponible pues la información de todos los recursos y servicios de la red se encuentra situada en una máquina conocida.

Sin embargo, sus desventajas son notables. En primer lugar, se debe disponer de un servidor de gran potencia para atender a todas las peticiones de los clientes. También cabe destacar la poca y costosa escalabilidad que presenta esta estructura, el problema de disponibilidad de información en caso de caída del servidor central y la escasa tolerancia a fallos (la cual se puede intentar solucionar a base de crear réplicas de la información / equipos lo que supone una gran inversión económica). Otro problema que aparece afecta a la publicación de los servicios y recursos multimedia. En este modelo, la publicación de servicios es manual.

Para implementar esta arquitectura primero se definen los componentes que la formarán, en base al estudio y descripción de las tecnologías estudiadas y que se pueden ver más ampliamente en los Anexos.

La arquitectura centralizada se basa en el uso de Servicios Web [Anexo 2] para acceder a una base de datos XML donde se almacenan todos los recursos de la red. En este caso se guardan recursos multimedia de la Fundación I2Cat.

A continuación se muestra un esquema de la situación.

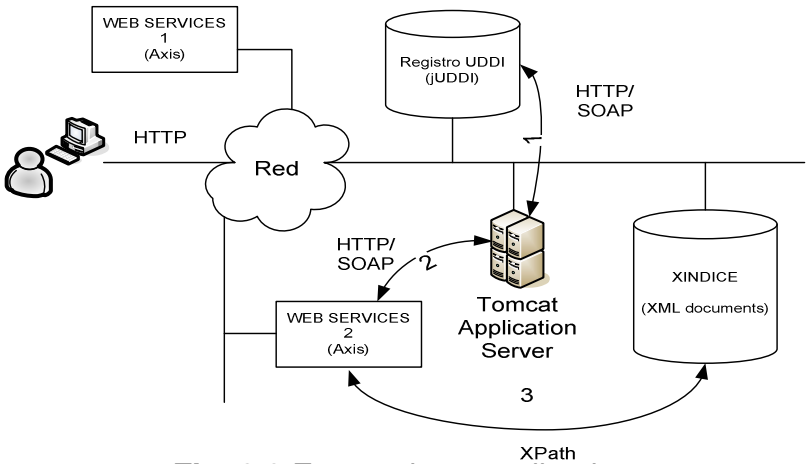



**Fig. 3.3** Escenario centralizado.

El usuario accede a una interfaz web para poder utilizar los Servicios Web publicados. Esta interfaz web la proporciona un servidor Tomcat que contiene un servlet, encargado de atender las peticiones HTTP hechas por el usuario a través de su navegador. El servlet implementa clientes de los Servicios Web.

El usuario podrá consultar los Servicios Web disponibles en el sistema gracias a su interacción con un registro UDDI centralizado [Anexo 2, apartado 2.8]. Se puede emplear el API de interacción con UDDI ofrecida por el proyecto jUDDI de Apache.

Una vez el usuario sabe cuáles son los servicios disponibles elige el que desee. En este caso nos centraremos en el de localización de recursos multimedia.

Una vez conocido el servicio, se le facilitará un entorno web para acceder a él. De forma transparente, se invocará un cliente del Servicio Web demandado. El cliente se escribirá en función de la descripción del servicio web que ofrece el documento WSDL generado [Anexo 2, apartado 2.7].

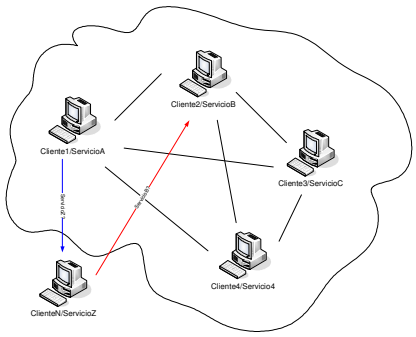
Acto seguido, el cliente enviará una petición HTTP/SOAP al Servicio Web, desplegado en un servidor Tomcat mediante el kit de desarrollo de Servicios Web de Apache Axis [Anexo 2, apartado 2.10].

El servicio web de localización de recursos se encargará de gestionar las comunicaciones con la base de datos XML, en este caso Xindice [Anexo 4], según las peticiones del usuario. Los Servicios Web se comunican con la base de datos XML mediante el uso de sintaxis Xpath ([30] y [Anexo 4]), ésta permite evaluar, extraer y encontrar los datos XML deseados. .

Todas las partes que integran esta arquitectura intercambiarán mensajes cuya información está en formato XML.

## 3.2. Arquitectura distribuida

La **estrategia distribuida**, P2P, se basa en un modelo en el que los usuarios del sistema pueden ser consumidores y productores (clientes y servidores) a la vez de un servicio. Los servicios no se encuentran localizados en un solo servidor sino que se encuentran repartidos por la red. La función de los usuarios es publicar sus servicios para que un usuario cualquiera los pueda localizar y usar en cualquier momento y en cualquier parte.




En este modelo, se puede plantear un escenario automático de publicación de servicios.

Las posibilidades que ofrece este modelo en cuanto a robustez, tolerancia a fallos y escalabilidad son enormes. Sin embargo, el problema se encuentra en la heterogeneidad que conforma una estrategia distribuida.

El entorno distribuido viene motivado por la posibilidad de desglosar la información, antes centralizada en una base de datos conocida, para crear peers (nodos de la red) especializados según la información que contengan. Los servicios disponibles en la red tampoco están publicados en un registro UDDI centralizado como el propuesto, sino que ahora se pueden publicar y consultar dinámicamente.


**Fig. 3.4** Paso hacia la localización distribuida. Fragmentación de la base de datos para crear nodos especializados.

Por lo tanto, la arquitectura distribuida propuesta es la siguiente:

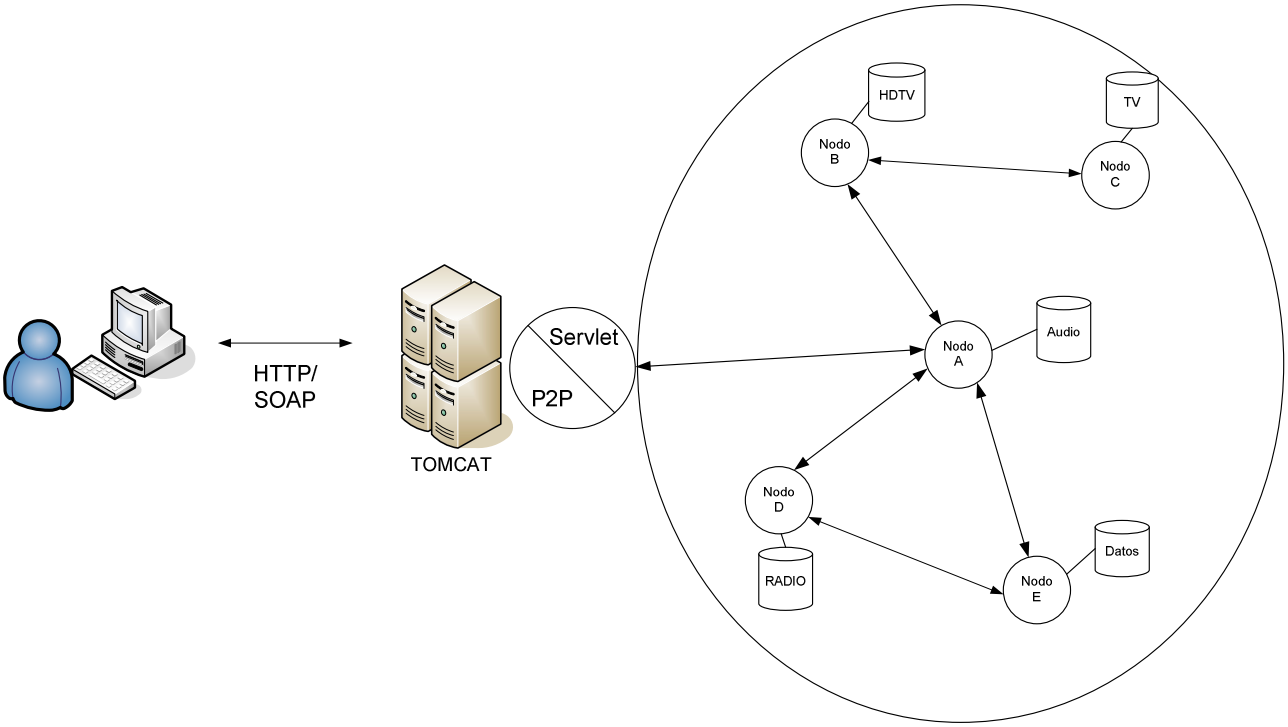



**Fig. 3.5** Escenario distribuido. Cada nodo se especializa en un tipo de recurso concreto.

En este escenario el usuario accede a una interfaz web como en el caso anterior. Sin embargo, esta vez se realiza la búsqueda de recursos multimedia en una red distribuida. Gracias al protocolo de descubrimiento (PDP) [Especificación de protocolos JXTA en Anexo 3, apartado 3.5] ofrecido por el framework JXTA [Anexo 3] el usuario podrá encontrar grupos de servicios, servicios, peers, recursos y pipes de manera automática.

Cada nodo de la red (peer especializado) dispone de su base de datos, Xindice, donde almacena la información relativa a los recursos multimedia. Estos nodos se agrupan en clusters con afinidades en el tipo de contenidos que desean publicar o encontrar. El grupo o clúster se gestiona por un super peer.

Un super peer puede:

- Almacenar datos de peers y recursos.
- Gestionar el grupo.
- Dispone de datos de otros super peers en la red (Super Peer ID, clusters de los que cada Super Peer administra, etc.).

Estos peers realizan la función de rendezvous si un peer no encuentra la información solicitada en el grupo.

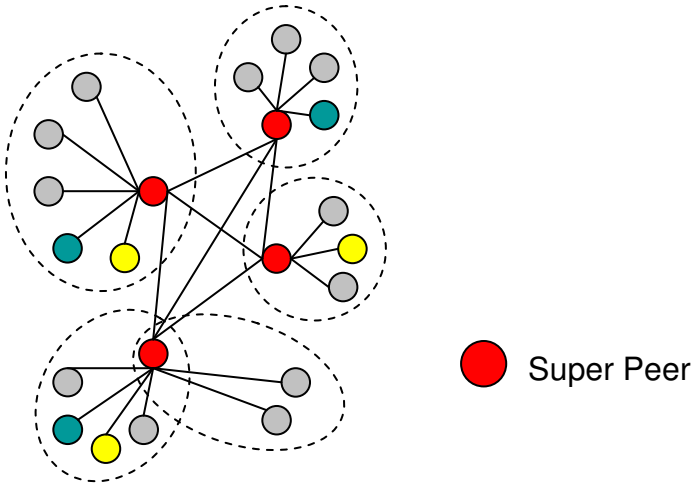



**Fig. 3.6** Red distribuida basada en Super Peers y nodos especializados.

En el Anexo 3 (apartados 3.3 y 3.4) se puede encontrar una descripción más detallada de los diferentes componentes y protocolos que constituyen la arquitectura P2P.

## 3.3. Representación de recursos multimedia (meta información)

Los datos, tanto en la arquitectura centralizada como en la arquitectura distribuida, se almacenan en una base de datos XML Xindice, la cual contiene información de los recursos (meta información) descrita según el framework MPEG 21 [Anexo 5].

Este framework especifica la representación de la meta información siguiendo una sintaxis XML. Se sustenta en la definición del Objeto Digital como objeto estructurado de información. También permite gestionar derechos de propiedad intelectual y de ejecución sobre un determinado recurso multimedia.

Por lo tanto, se localizarán y publicarán Objetos Digitales.

## 3.4. Comparativa P2P vs. Servicios Web

A continuación se muestra de manera resumida una comparativa entre Servicios Web y redes P2P.

**Tabla 3.1**

| Características | Webservices | P2P |
|---|---|---|
| Población | Grandes Servidores | Cualquier dispositivo o PC |
| Propiedad | Simple / múltiple | Múltiple |
| Búsqueda | Servicio búsqueda | Descentralizado |
| Gestión de usuarios | NA | Descentralizado |
| Gestión de recursos | Distribuida | Distribuida |
| Organización de trabajo | Descentralizado | Descentralizado |
| Interoperabilidad | Estándar | No estándar |
| Escalabilidad | ¿1000? | ¿Millones? |
| Capacidad | Garantizada | Varia |
| Prestaciones | Alta | Muy alta |

La arquitectura distribuida ofrece una escalabilidad enorme, infinitamente mayor a la ofrecida por la arquitectura centralizada.

A la arquitectura distribuida se pueden añadir nuevos nodos de manera automática y éstos se pueden comunicar entre sí de manera directa.

# CAPÍTULO 4. DISEÑO

En este proyecto se ha diseñado una interfaz web sencilla para acceder al servicio de localización. Esta interfaz proporcionará todas las funcionalidades descritas por el sistema, concretamente nos centraremos en las pertinentes al módulo de localización.

Esta interfaz web reflejará los resultados obtenidos de la búsqueda centralizada o distribuida.

En el Proyecto Integrado ya hay un grupo encargado de diseñar la interfaz de usuario, pero se ha diseñado una sencilla para realizar pruebas.

En este apartado también se comenta el diseño realizado para llevar a cabo la búsqueda de recursos multimedia para las arquitecturas centralizada y distribuida, estudiadas anteriormente.

## 4.1. Escenario Centralizado

En la visión centralizada se debe diseñar principalmente el servicio web a implementar y posteriormente hacerlo accesible para los usuarios.

En primer lugar, se definen una serie de clases Java que permitan interactuar con la base de datos Xindice que se instalará. Ésta base de datos centralizada, contendrá todos los recursos de la red.

Mediante estas clases Java se podrá:

- Conectar con la base de datos
- Añadir un recurso
- Eliminar un recurso
- Consultar un recurso dado un campo de búsqueda de la base de datos.
- Modificar el contenido de un recurso

Para ello se utilizará la API de Xindice. Estas clases definirán los métodos que se publicarán en el servicio web gracias a los documentos WSDL [Anexo 2, apartado 2.7], los cuales sirven para especificar formalmente el uso y acceso el servicio web.

Por lo que concierne a la base de datos, no debemos más que instalarla. No debemos preocuparnos de los campos y tablas que la conformarán, tan sólo de elaborar un documento XML estructurado según queramos, en nuestro caso siguiendo el framework MPEG 21 que especifica la estructuración de la meta información para la creación de Objetos Digitales. La información será almacenada según el documento XML generado. La única cosa a considerar es la colección en la que se almacenarán los recursos (documentos XML). Se

pueden almacenar todos en una misma colección [Anexo 4, apartado 4.2], o según colecciones especializadas para cada tipo de recurso.

Una vez se tengan las clases Java que facilitarán el servicio, se desplegará el servicio web gracias al kit de desarrollo Apache Axis.

Una vez el servicio web esté operativo, éste debe ser publicado en el registro UDDI privado, gracias al API para Java proporcionado por jUDDI, para permitir que los usuarios del sistema puedan encontrar el servicio de localización de recursos multimedia. La publicación de los servicios, su descripción, será almacenada en una base de datos relacional MySQL siguiendo la especificación de UDDI.

Con la finalidad de que el usuario pueda acceder al sistema de manera transparente, se implementará un servlet en un servidor de aplicaciones Tomcat. Éste facilitará al usuario un entorno web a través del cual poder encontrar el servicio de localización y usarlo adecuadamente. Para las páginas web se usará lenguaje JSP. Éste permite añadir funciones Java entre código HTML.

A la hora de diseñar la aplicación web, se seguirá un patrón de diseño Modelo – Vista – Controlador (MVC).

- Capa de Modelo: esta capa se encarga de manejar los datos y controlar sus transformaciones. La principal funcionalidad de esta capa será delegada a una serie de gestores que serán descritos más adelante en el apartado de diagrama de clases de la capa de modelo (4.4.3).
- Capa de Vista: esta capa se encarga de la representación visual fruto de una petición de un usuario. La función de la capa de vista será llevada a cabo por los JSP. Para ver el diseño de esta capa, ver el apartado Diseño de interfaces de usuario (4.4.1).
- Capa de Controlador: esta capa se encarga de recibir peticiones, en nuestro caso HTTP, procedentes de la capa de vistas y pasárselas a la capa de modelo. Una vez atendida la petición devuelve los datos a la capa de vista para que los presente al usuario. La función de la capa de controlador será llevada a cabo por el servlet, descrito en el apartado de diagrama de clases de la capa de control (4.4.2).

Gracias a esta separación, se pueden modificar las vistas sin tener que retocar el código implementado. Por lo tanto, supone una estructura que permite reutilizar el código desarrollado.

El usuario podrá, por tanto, usar los métodos del Servicio Web. Para usarlos, la interfaz web le facilitará una serie de formularios para realizar las funciones que desee (publicar, buscar, eliminar, etc.).

El usuario envía un formulario de búsqueda al servlet mediante HTTP. Estos datos serán gestionados por el servlet el cual se los pasará a los clientes de Servicios Web que tiene implementados para invocar sus métodos. Finalmente,

los métodos interactuarán con el servicio web mediante SOAP y éstos con la base de datos Xindice.

Los Servicios Web han permitido separar la capa de servicios de la aplicación web. Permitiendo que, si se desea, este servicio de localización sea rehusado por otras aplicaciones o servicios.

## 4.2. Escenario Distribuido

La arquitectura distribuida estará compuesta por una serie de nodos especializados según un tipo de recurso. Esto permite, con respecto a la visión centralizada, una separación del contenido.

Mediante la especificación del framework JXTA, se montarán una serie de nodos. Cada nodo dispondrá de su almacén de datos Xindice particular con la información de aquellos recursos que disponga.

Para la organización de los nodos especializados, se puede montar un grupo especializado en contenido multimedia (clústeres). Así todos los peers que pretendan aportar su servicio de publicación de recursos deben añadirse al grupo común dominado por un Super Peer.

Estos super peers son nodos que actúan como un servidor centralizado respecto a un conjunto de clientes. Se encargarán de gestionar el grupo o clúster y serán del tipo rendezvous.

El usuario accede al sistema de localización a través de una interfaz web, de la misma manera que anteriormente, sin embargo, esta vez detrás existe toda una arquitectura distribuida. La interfaz gráfica debe disponer de las mismas funcionalidades que las citadas anteriormente.

A continuación se muestra un diagrama de secuencia de la posible localización de un recurso.

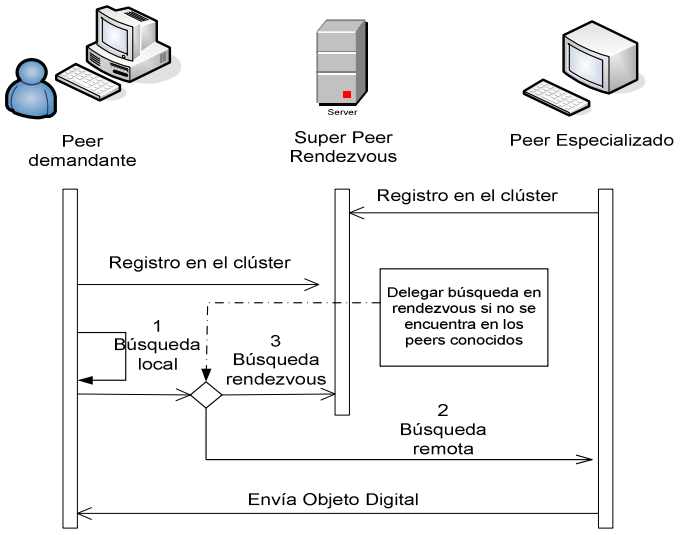



**Fig. 4.1** Localización de un recurso en el escenario distribuido.

Para llevar a cabo la búsqueda de un recurso se recurrirá al uso del servicio de descubrimiento [Anexo 3, apartado 3.5]. El peer que desea encontrar un recurso o servicio primero comprobará si tiene el anuncio almacenado en su caché local fruto de otro descubrimiento anterior. En caso contrario se realizará la búsqueda por la red. Se enviará una petición de búsqueda a los peers que conoce y en caso de no recibir respuesta se recurrirá al uso de los peers rendezvous para que éstos propaguen la petición a otros peers desconocidos por el peer demandante. Finalmente, se recibe el recurso (objeto digital) solicitado. Es posible que debido al constante cambio que caracteriza a este tipo de redes, no se obtenga el recurso.

En este caso el servlet en vez de implementar un cliente de servicio web, debe poder acceder a la red Peer to Peer como nodo. Este nodo deberá de ser capaz de:

- Encontrar el NetPeerGroup (Mundo).
- Encontrar un grupo determinado.
- Encontrar el servicio que desee.
- Acceder al servicio a partir de su anuncio.
- Publicar sus recursos.
- Comunicarse mediante pipes con otros peers.

Para ello se hará uso de los protocolos JXTA adecuados.

## 4.3. Interfaz del módulo de localización

La interfaz del módulo de sistema de localización será creada mediante Servicios Web (WSDL), HTTP y SOAP, mientras que la arquitectura del sistema de localización que se implemente detrás puede ser una de las 2 comentadas: centralizada o distribuida.

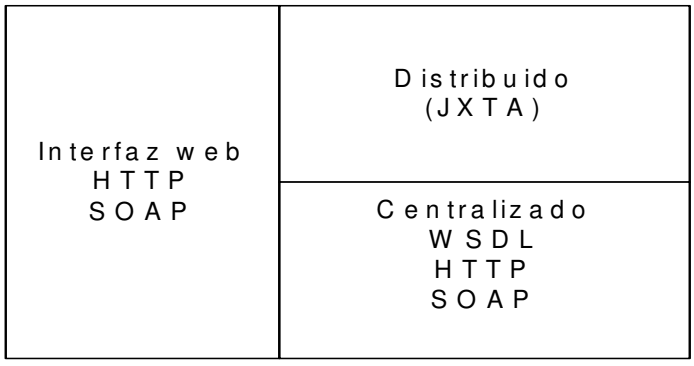



**Fig. 4.2** Interfaz módulo de localización.

El módulo de interfaz de usuario envía al módulo de servicio de localización el formulario de búsqueda (indicando qué se desea buscar: un servidor, un recurso multimedia, etc.) mediante HTTP. El módulo de localización de servicio le responde con el listado de resultado de búsqueda.

## 4.4. Diseño de la aplicación web

### 4.4.1. Diseño de interfaces de usuario (Capa de vistas)

A continuación se muestran diferentes vistas que representan el diseño de la interfaz de usuario. Estas vistas corresponden a la parte de presentación diseñada y programadas mediante lenguaje JSP.

Acceso al sistema

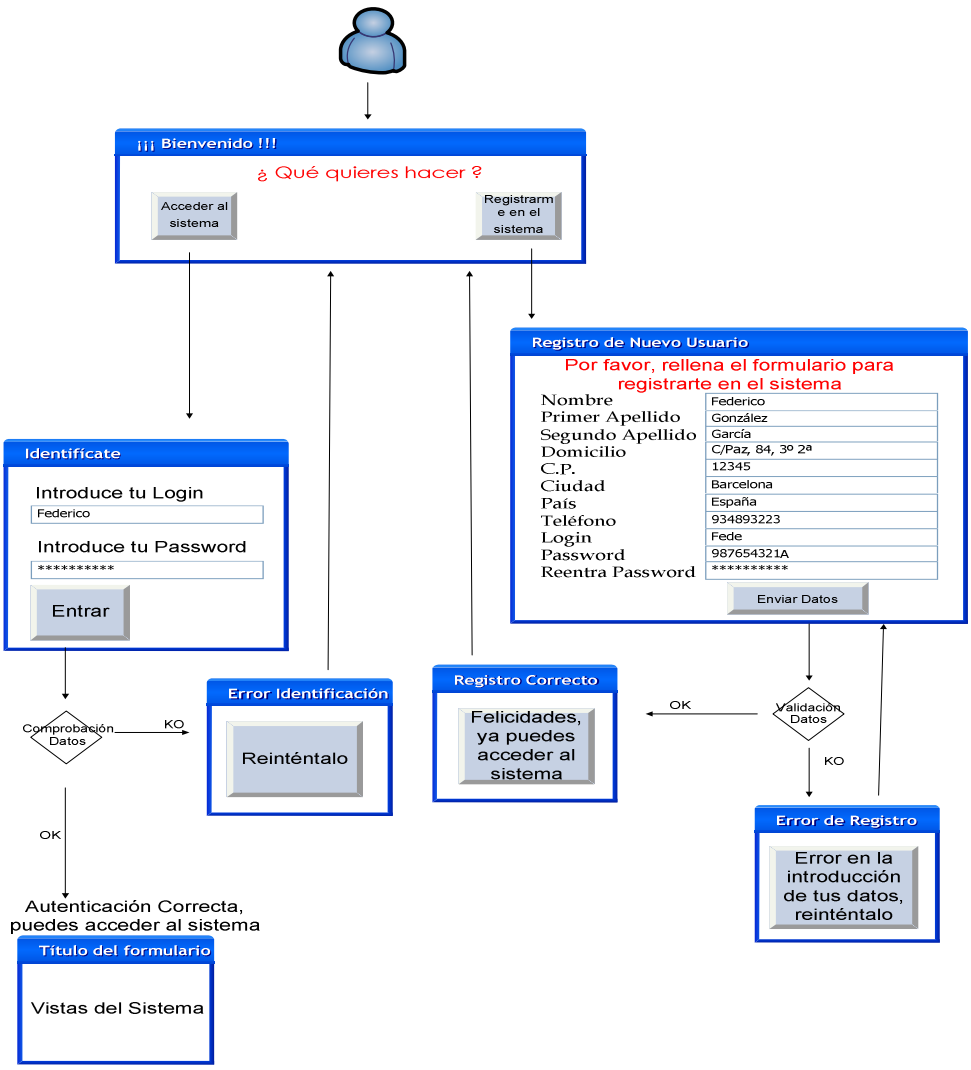



**Fig. 4.3** Vistas del proceso de acceso al sistema.

La validación del usuario será llevada acabo por un módulo especializado descrito en el Proyecto Integrado elaborado por I2Cat. La autenticación se realiza mediante login y password.

## Perfil de acceso

- Administrador de Grupo (ARG)

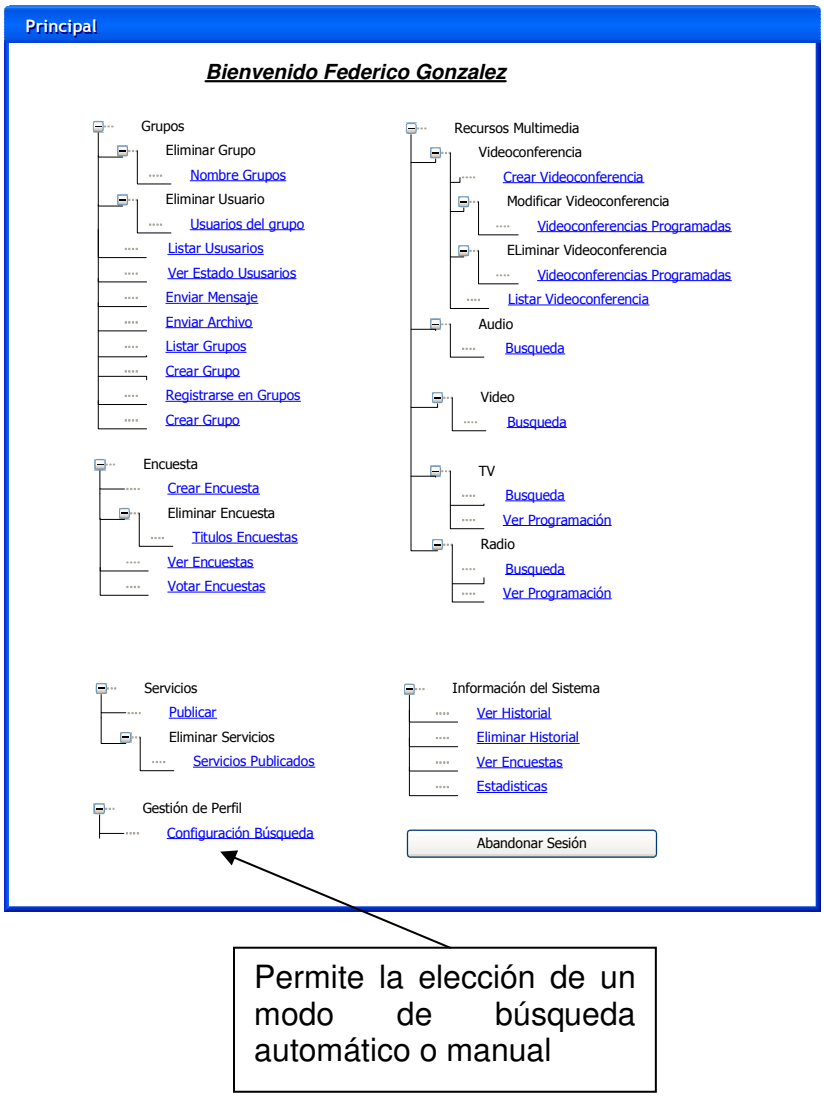



**Fig. 4.4** Pantalla principal, perfil de Administrador de grupo.

Una vez autenticado el usuario en el sistema, la aplicación muestra la página principal en función del perfil de usuario.

Esta vista corresponde a la pantalla principal que le aparece al administrador de un grupo, por ejemplo a un super peer en una arquitectura distribuida.

## Funcionalidades del sistema

- Localización, listado y reproducción de un recurso multimedia.

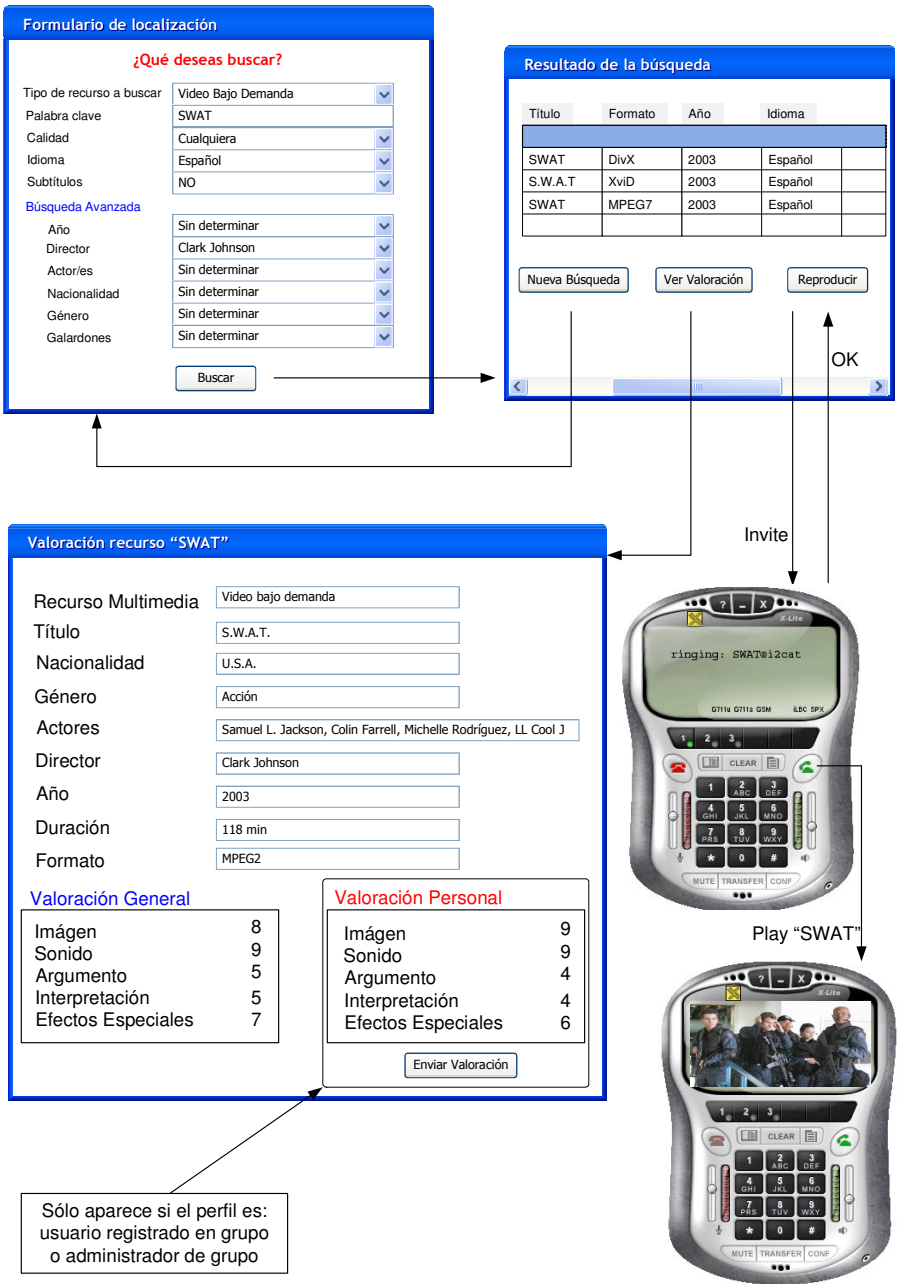



**Fig. 4.5** Vistas del proceso de localización y reproducción de un recurso multimedia.

### 4.4.2. Capa de Control

El usuario realiza peticiones al servlet (clase SuperServlet[2]) y éste, al recibir la petición, la dirige al gestor específico que la atenderá. Para crear la instancia del gestor que atenderá la petición, se recurre a una Factory (patrón de diseño[3]). La clase Factory permite instanciar clases al vuelo, es decir, crear instancias de los gestores que tiene implementados automáticamente.

La clase Factory sigue el patrón de diseño Singleton. Esto permite tener una única instancia de cada gestor. Esto supone que si se atiende una petición para un gestor que ya ha sido instanciado previamente, no se debe volver a instanciarla nuevamente pues se devuelve la instancia realizada anteriormente. En cambio, si no se usara una clase Singleton, para cada petición se debería instanciar un gestor cada vez.

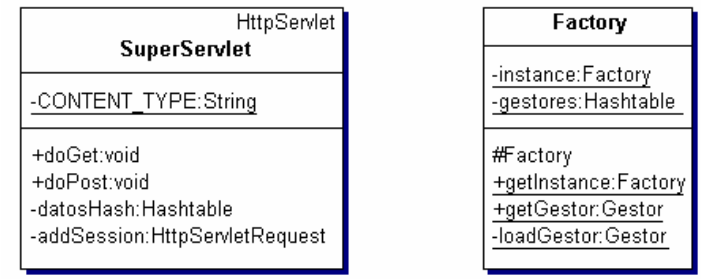



**Fig. 4.6** Clases capa de control.

### 4.4.3. Capa de Modelo

Esta capa está formada por un conjunto de gestores que atenderán las peticiones. La clase Gestor sigue un patrón de diseño Command. A continuación se explica el funcionamiento de éstos.

***GestorAutenticación***. Se encargará de comunicarse con el módulo de autenticación e indicará si el usuario está registrado o no. Para realizar esta comunicación se ha creado la clase AtuenticaClient.

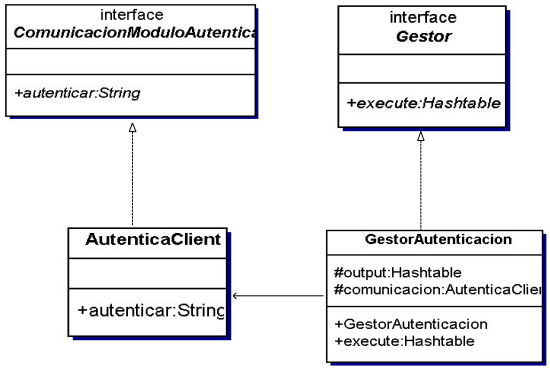



**Fig. 4.7** Diagrama de clases. GestorAutenticación.

[2] Sigue un patrón de diseño Front Controller
[3] Un patrón de diseño representa una solución estándar para un problema común de programación.

**GestorXmlDb.** Permite interaccionar con el módulo de localización para la búsqueda de un recurso. Para esta interacción se crea la clase *XmlDbClient.* Esta clase tiene implementados los métodos que permiten comunicarse con el servicio web (clientes de Servicios Web).

Cuando un método de un cliente se invoca, lo que se hace es una petición al Servicio Web. Se envía un mensaje SOAP que invocará uno de sus métodos remotos.

Ésta es la clase principal que accede al servicio de localización.

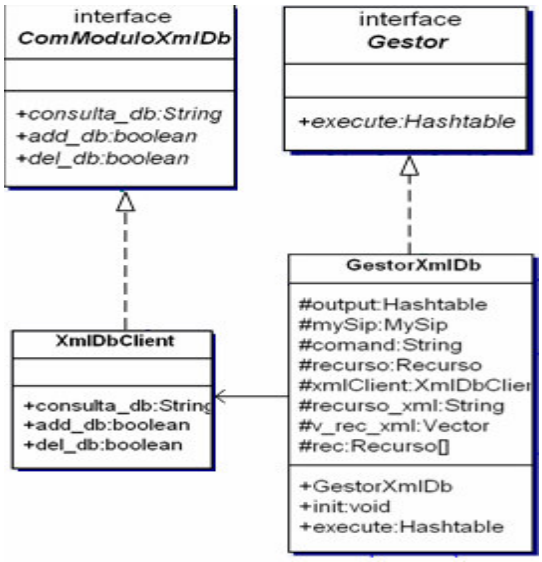



**Fig. 4.8** Diagrama de clases. GestorXmlDb.

La clase *GestorXmlDb* almacena los recursos obtenidos en un array de recursos.

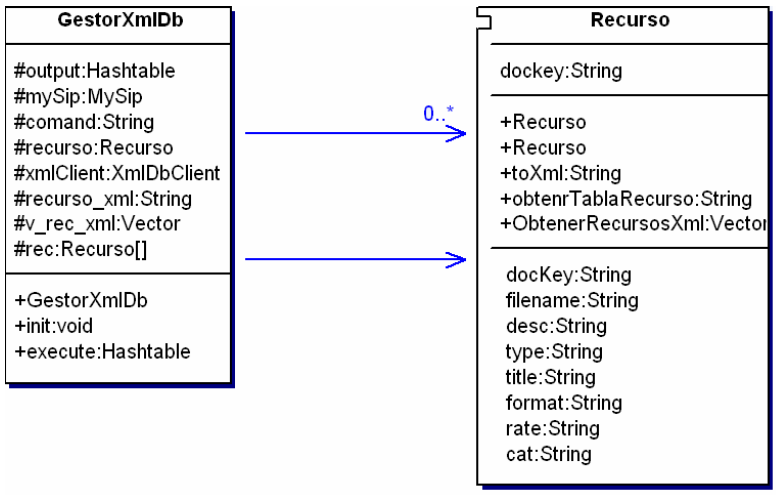



**Fig. 4.9** Diagrama de clases. Clase Recurso.

La clase *Recurso* contiene todos los campos necesarios para definir un recurso y para gestionarlo.

**GestorTranscodificador.** Será el encargado de obtener una lista de los recursos que se puedan transcodificar. Para obtener esta lista se comunicará con el módulo de trancodificación mediante la clase TrClient.

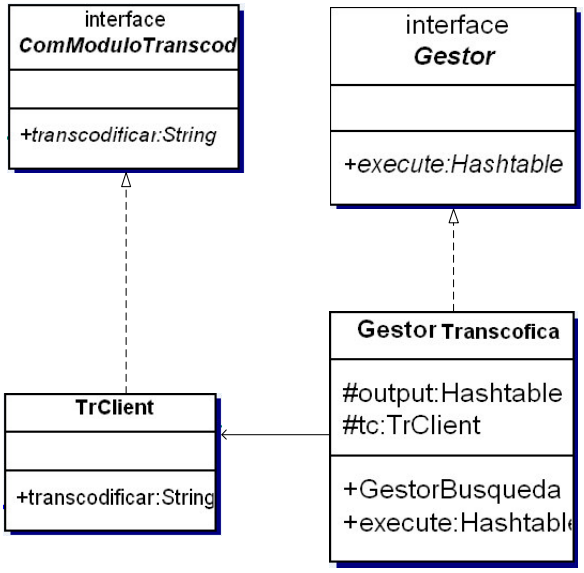



**Fig. 4.10** Diagrama de clases. GestorTranscodificador.

**GestorMedia.** Este gestor se encargará de atender las peticiones que requieran la reproducción y el control de un flujo de media. Para ello se comunicará con el módulo de provisión de servicios mediante el protocolo de señalización de inicio de sesión SIP.

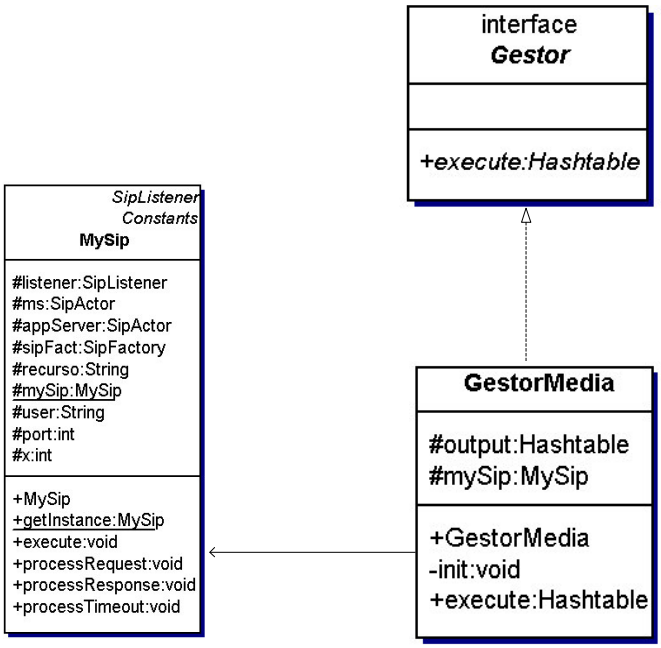



**Fig. 4.11** Diagrama de clases. GestorMedia.

## 4.5. Diagrama de clases servicio web

Para el servicio web de localización se definen las siguientes clases.

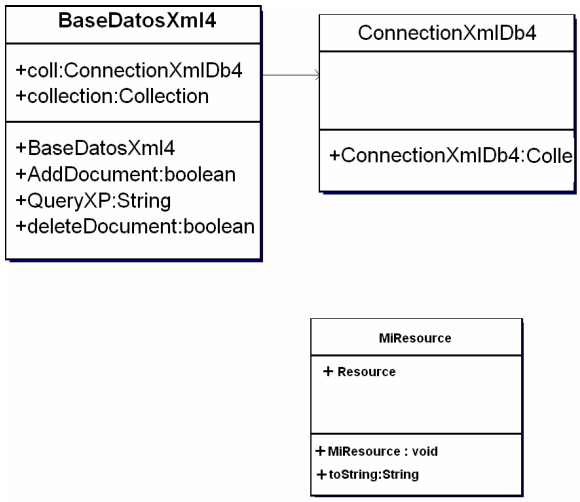



**Fig. 4.12** Diagrama de clases. Clases para crear el servicio web.

La clase *BaseDatosXml* se encarga de implementar los métodos que interaccionarán con la base de datos Xindice. Estos métodos son: consultar, añadir, eliminar, modificar, etc.

La clase *ConnectionXmlDb* se encarga de conectar con la base de datos Xindice.

# CAPÍTULO 5. IMPLEMENTACIÓN

En este capítulo se describen los principales aspectos relacionados con la implementación según el diseño realizado en el apartado anterior.

Se comienza definiendo el entorno de trabajo y las tecnologías y herramientas empleadas.

Acto seguido se comentan los pasos seguidos y las implementaciones realizadas

## 5.1. Entorno de trabajo

El desarrollo se ha llevado a cabo en las máquinas facilitadas por la universidad en el laboratorio 325.

Para la implementación de Servicios Web se han usado 2 máquinas: una máquina que contiene el servlet y la base de datos Xindice y otro equipo para publicar los Servicios Web.

Para el desarrollo de las aplicaciones P2P se ha empleado un hub y 2 PC's. En cada PC se ha desarrollado un peer que se comunica con el otro mediante pipes.

El sistema operativo usado ha sido Windows XP Profesional, aunque todo lo estudiado e implementado es compatible con otras plataformas como Linux pues son códigos abiertos, multiplataforma y desarrollados en Java.

De hecho, está previsto instalar todo lo desarrollado en una maqueta de I2Cat la cual dispone de una distribución Debian de Linux.

La elección se ha basado en los hábitos que tengo a la hora de trabajar. Estoy más familiarizado con Sistemas Operativos de Microsoft.

Para la gestión de la base de datos relacional MySQL se ha usado la interfaz gráfica MySQL Control Center. Esta interfaz facilita un entorno visual y flexible para la creación de tablas y campos, así como la introducción de los datos a almacenar.

Un aspecto interesante a comentar ha sido el uso de un servidor de ficheros CVS, en el que se puede almacenar todo lo desarrollado o escrito. Este servidor permite tener actualizada toda la información almacenada o incluso guardar versiones antiguas. También permite disponer de copia de toda la información que se albergue en él, en cualquier máquina con la que se trabaje.

El servidor de ficheros es una herramienta ideal para el trabajo en grupo pues todos los miembros de un grupo pueden consultar la misma información actualizada desde cualquier lugar.

La plataforma de programación usada para la implementación de las aplicaciones ha sido Java. En concreto la versión 1.4.2 del J2SDK de la plataforma Java 2 Standard Edition. Como entorno de programación se ha usado Eclipse Project. Esta aplicación ofrece una interfaz de usuario amigable y funcionalidades de debbug del código fuente.

Esta versión del JSDK ha sido elegida debido a que es una de las más actuales, estables y compatibles con el resto de versiones de otros componentes utilizados (Xindice, Axis, Tomcat, etc.).

## 5.2. Tecnologías y herramientas utilizadas

- **SDK 1.4.2:** versión del Java 2 SDK utilizada para desarrollar tanto los Servicios Web como la implementación de peers JXTA.
- **Apache Axis 1.1**: kit de desarrollo de Apache para el despliegue de Servicios Web en la arquitectura centralizada.
- **Apache Xindice 1.0**: base de datos XML de Apache. Se usa como almacén de datos para la arquitectura centralizada.
- **API de base de datos XML**: proporciona clases e interfaces para interactuar con la base de datos Apache Xindice. Package org.xml.sax.
- **Parseador SAX**: proporciona las clases e interfaces de la API de SAX XML para Java. Package org.xml.sax.
- **Transformaciones XSLT**: proporciona clases e interfaces para transformar documentos XML mediante XSLT. Package javax.xml.transform.
- **Proyecto JXTA**: especificación para implementar una red distribuida P2P en Java.
- **JXTA:** proporciona clases e interfaces para implementar una red distribuida según la especificación JXTA. Package net.jxta.
- **jxta-shell versión 2.3.1:** esta aplicación dispone de las librerías necesarias para realizar la implementación del proyecto. También permite probar el funcionamiento de una red P2P a partir de una consola que facilita una serie de comandos.
- **Apache Tomcat versión 4.1.30:** este servidor de aplicaciones permitirá por un lado tener un servlet atendiendo las peticiones que el usuario hace a través de su navegador y por otro lado tener los Servicios Web desplegados y por lo tanto accesibles. Se ha elegido esta versión por ser una de las más actuales, estable y compatible con la versión del kit de desarrollo de Servicios Web Apache Axis.
- **Java Server Pages (JSP)**: lenguaje de programación usado para crear las vistas de la interfaz web.
- **MySQL Server**: base de datos relacional para la implementación del registro UDDI privado.

## 5.3. Interfaz web y Servlet

La parte de la interfaz web ha sido desarrollada mediante JSP's. Estos JSP's son atendidos por un servlet. Se ha seguido el patrón de diseño de separación de modelo, vistas y controlador (MVC) definido en el diseño.

Hay una versión de la implementación de las vistas en las que se aplican hojas de estilo definidas por I2Cat pues se reutiliza el entorno para el Proyecto Integrado.

La interfaz web se ha implementado para poder probar la búsqueda centralizada. No se ha acoplado a las pruebas hechas de la implementación de la arquitectura distribuida mediante JXTA.

La interfaz web permite consultar (un recurso concreto o obtener un listado de todos), añadir un recurso y eliminar un recurso. No se han implementado todas las funcionalidades especificadas en el capítulo 2.

A continuación se muestran las vistas implementadas. Los resultados mostrados son fruto de una búsqueda centralizada. Después de realizar la búsqueda, el usuario puede reproducir el recurso seleccionado a partir de la interacción con el módulo de provisión de servicios.

Para añadir o eliminar un recurso se debe rellenar un formulario. Una vez enviado, se informa del éxito de la operación o si se ha producido algún error.

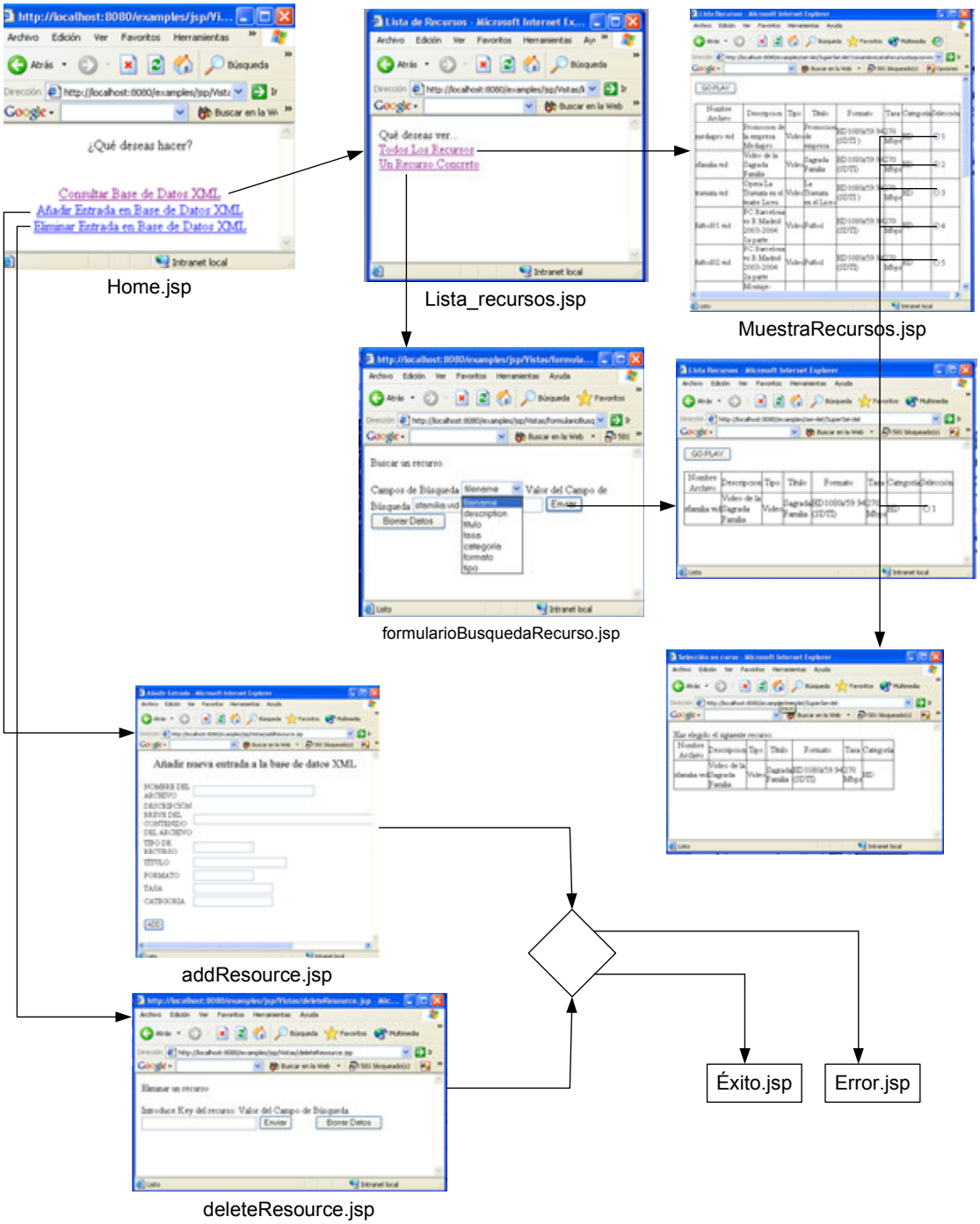



**Fig. 5.1** Interacción entre las páginas web. Aplicación web.

## 5.4. Escenario centralizado

La implementación realizada para este escenario ha sido la siguiente.

En primer lugar, se ha desarrollado un servicio web desplegado en un servidor Tomcat mediante Apache Axis que ofrece el servicio de localización. Éste básicamente permite:

- Consultar la base de datos
- Añadir un recurso a la base de datos
- Eliminar un recurso de la base de datos.

También se han implementado otros Servicios Web sencillos simulando a los módulos de autenticación, calidad de servicio y transcodificación acorde con los diseñados para el Proyecto Integrado.


**Fig. 5.2** Servicios Web implementados. Vista de los servicios desplegados mediante Apache Axis en Tomcat.

Se ha instalado una base de datos MySQL y se han creado tablas con sus respectivos campos a modo de almacén de servicios. La función de este repositorio es la de implementar el registro UDDI para la publicación de los servicios, mediante las API de jUDDI. No obstante, no se ha pasado del diseño de la base de datos. No se ha implementado el registro UDDI privado.

Acto seguido, se han adaptado clientes de Servicios Web al servlet diseñado. Éstos permiten invocar remotamente mediante mensajes SOAP los métodos implementados por el servicio web.

Para añadir un recurso en la base de datos XML, en el diseño se comentaba que se almacenaba mediante la descripción de objeto digital. No obstante, se ha implementado una manera más sencilla a modo de prueba. Simplemente se ha descrito un recurso mediante una semántica XML básica.

```
<resource>
        <filename>sfamilia.vid</filename>
        <desc>Video de la sagrada familia</desc>
        <type>Video</type>
        <title>Sagrada Familia</title>
        <format>HD1080i/59.94(SDTI)</format>
        <rate>270 Mbps</rate>
        <cat>HD</cat>
<resource>
```

**Fig. 5.3** Ejemplo de la estructura implementada para la descripción y almacenamiento de recursos multimedia en la base de datos Xindice.

Para eliminar un recurso de la base de datos la interfaz de usuario requiere la introducción del key (clave identificadora) que por defecto asigna la base de datos Xindice a un documento XML cuando se almacena.

Para realizar la búsqueda de un recurso en la base de datos Xindice, lo que se hace primeramente es solicitar al usuario mediante un formulario en la interfaz web que introduzca un campo de búsqueda y un valor para éste. Si lo desea el usuario también tiene la opción de solicitar el listado completo de recursos almacenados en la base de datos Xindice.

A la hora de retornar los resultados, se deben adaptar los formatos de manera adecuada para que los diferentes componentes de la cadena se entiendan.

Por este motivo se ha adaptado el resultado de la búsqueda de la base de datos XML. Ésta devuelve una serie de documentos XML de manera continua, especificando para cada recurso la cabecera <?xml version=”1.0”>. Se ha sustraído esta cabecera pues después, al recibir el resultado de la búsqueda en el servlet, con la finalidad de mostrarle los datos obtenidos al usuario, no se podían transformar adecuadamente pues la especificación determina que sólo puede haber una cabecera XML. Por lo tanto, lo que se ha hecho en el servicio web que devuelve el resultado, es englobar todos los recursos devueltos en un elemento RESULTSET.

Ejemplo de retorno de un recurso en XML.

```
<RESULTSET>
   <resource xmlnsrc="http://xml.apache.org/xindice/Query" src:col="/db/xml_db_re
      cursos2" src:key="1a8ead6d5b07f04f0000010089e78149">
      <filename>sfamilia.vid</filename>
      <desc>Video de la Sagrada Familia</desc>
      <type>Video</type>
      <title>Sagrada Familia</title>
      <format>HD1080i/59.94(SDTI)</format>
      <rate>270 Mbps</rate>
      <cat>HD</cat>
   </resource>
```

```
</RESULTSET>
```

**Fig. 5.4** Contenido del mensaje XML de respuesta a una petición de búsqueda de un recurso concreto, emitido por el servicio web.

Todas las operaciones efectuadas por los Servicios Web en la base de datos (añadir, extraer y consultar) se han llevado a cabo mediante el uso de expresiones XPath.

Una vez el resultado se recibe en el servlet, se parsea el contenido mediante SAX para almacenar los resultados en un array de recursos.

Acto seguido, para mostrar al usuario los resultados, surge el problema de la representación de los datos. Las páginas JSP deben mostrar los datos en formato HTML por ese motivo se recurre a las transformaciones XSLT [Anexo 2, apartado 2.4]. Mediante una plantilla XSLT se especifica cómo debe modificarse el archivo XML recibido del servicio web para que lo traduzca a HTML.

Al usuario se le muestran por pantalla los resultados en el formato correcto, permitiéndole escoger la opción de reproducir uno de los recursos obtenidos.

Mediante un botón de selección [Fig. 5.1], el usuario indica qué recurso va a reproducir. Acto seguido se le muestra a modo de confirmación el recurso que ha seleccionado. Si el usuario acepta, el vector de recursos almacenado se recorre con el fin de seleccionar el recurso demandado y pasarle los datos al módulo de provisión de servicio, el cual se encargará de reproducirlo adecuadamente.

A continuación se muestra un esquema del funcionamiento de la aplicación web desde el punto de vista funcional (SuperServlet).

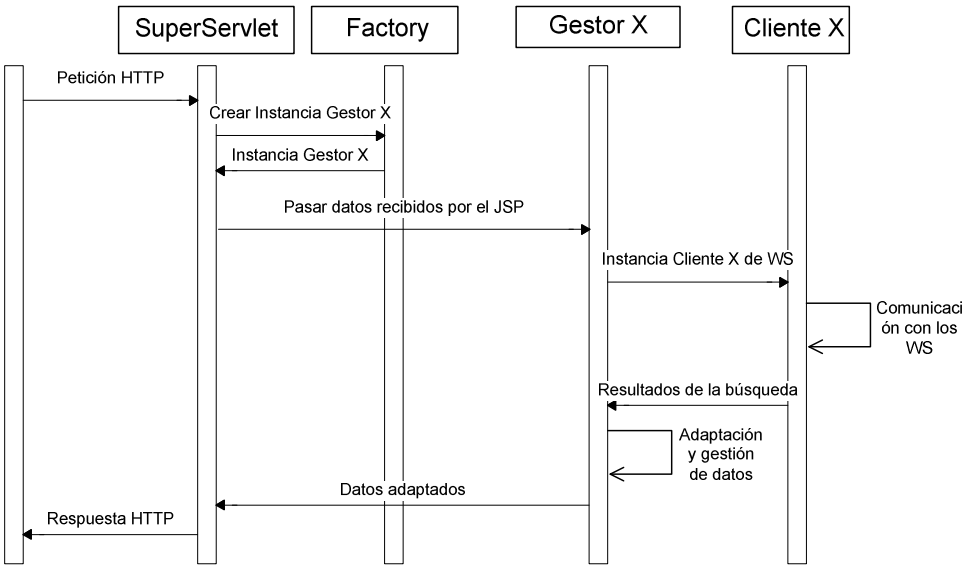



**Fig. 5.5** Funcionamiento de la aplicación web.

El esquema siguiente representa de la implementación realizada en la que se muestran los detalles principales.

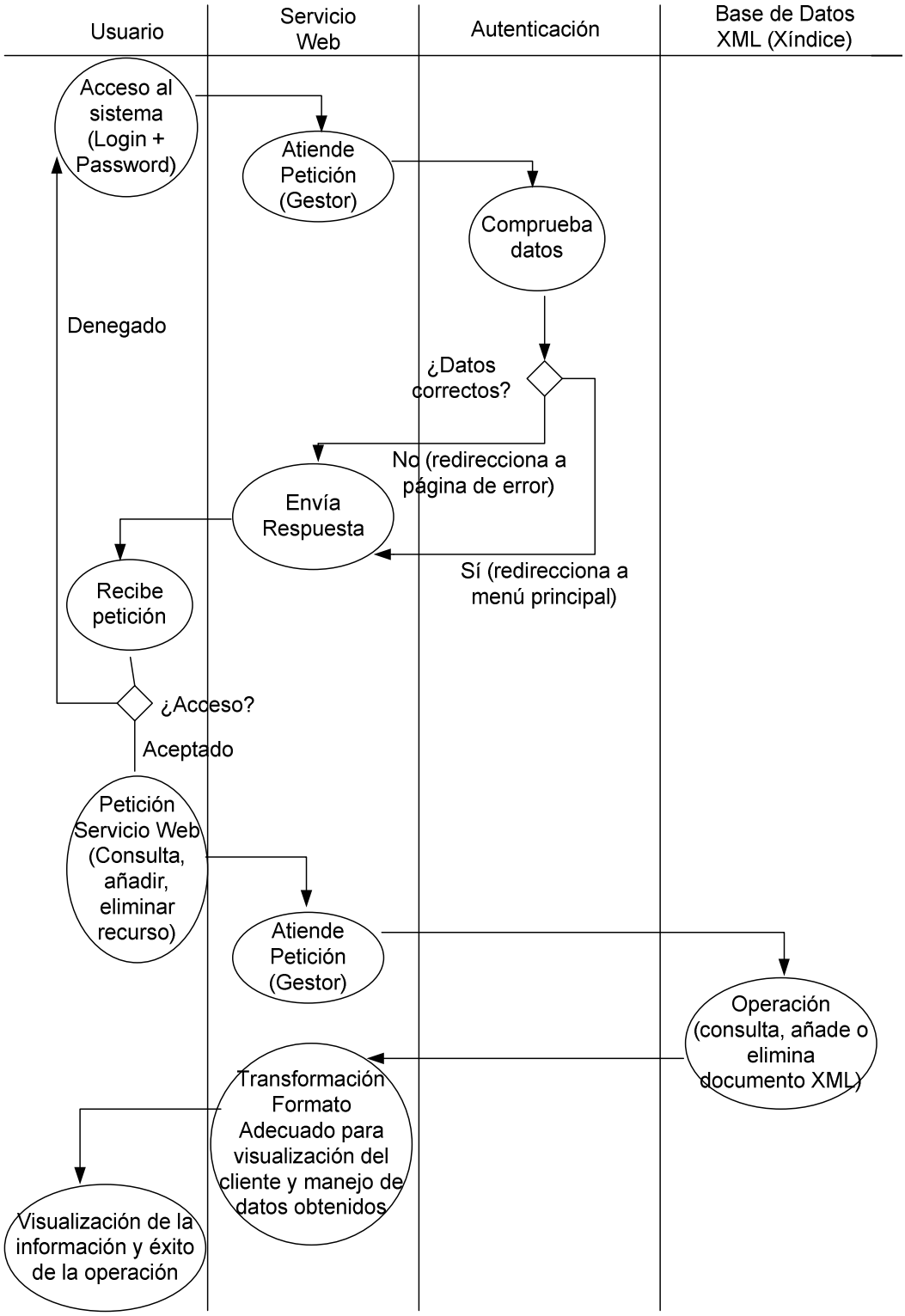



**Fig. 5.6** Esquema general de la implementación de la arquitectura centralizada.

## 5.5. Escenario distribuido

Para la implementación de este escenario se ha empezado realizando una aplicación sencilla a la que se le ha ido añadiendo funcionalidades.

La primera implementación que se ha realizado ha sido una aplicación tipo *HelloWorld*. Esta aplicación realiza las siguientes operaciones:

- Crear e iniciar un grupo global NetPeerGroup.
- Mostrar el anuncio del NetPeerGroup
- Mostrar información del grupo:
    - Identificador de grupo
    - Identificador de peer
    - Nombre del peer

También se han obtenido los servicios principales del núcleo:

- Descubrimiento (Discovery Service)
- Pipe (Pipe Service)
- Pertenencia a un grupo (Membership Service)
- Resolución (Resolver Service)

Sin embargo, no se han hecho funcionar.

A continuación se muestra lo que la aplicación muestra por pantalla.

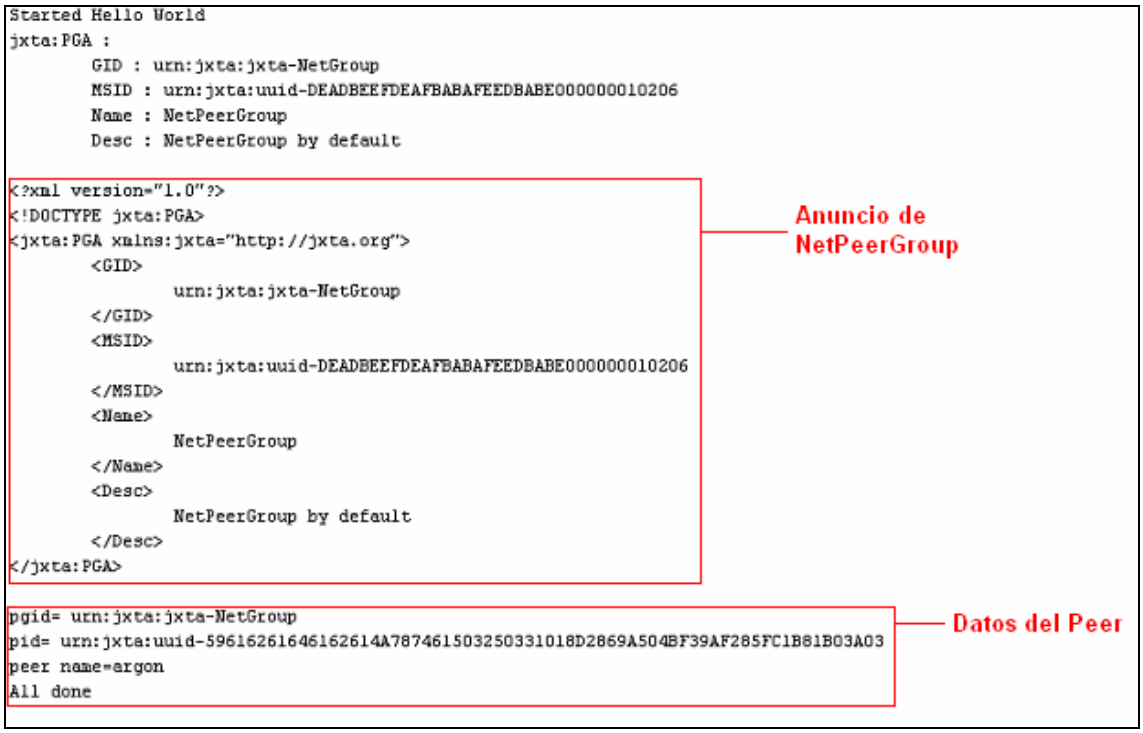



**Fig. 5.7** Aplicación HelloWorld.

Después de esta aplicación introductoria, se ha implementado una aplicación que permite probar el servicio de pipes, el cual permite a dos peers establecer un intercambio de mensajes a través de estas tuberías virtuales.

Para ello se ha implementado un peer que permanece a la escucha a través del anuncio que ha publicado de su pipe (input pipe).

Este peer lo primero que hace es unirse al NetPeerGroup. Acto seguido, se sirve del servicio de creación de pipes para crear su terminal (punto final de entrada o salida) por donde recibirá los mensajes de otros peers. Esta pipe se crea a partir del anuncio que confecciona el peer de su pipe [Fig. 5.8].

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement
xmlns:jxta="http://jxta.org">
       <Id>
              urn:jxta:uuid-
59616261646162614E50472050325033C0C1DE89719B456691A5
96B983BA0E1004
       </Id>
       <Type>
              JxtaUnicast
       </Type>
       <Name>
              PipeExample
       </Name>
</jxta:PipeAdvertisement>
```

**Fig. 5.8** Anuncio de pipe.

Para realizar esta operación se guarda el anuncio en un archivo local para asegurar que ambos extremos leen el mismo anuncio.

Finalmente, el peer permanece a la espera de la llegada asíncrona de mensajes. Esto lo realiza implementando una interfaz de escucha proporcionada por PipeMsgListener. Esta interfaz obliga a implementar un método (pipeMsgEvent) que permite recibir los mensajes que manden otros peers.

Por otro lado, se encuentra el peer que debe enviar mensajes al peer que permanece a la escucha.

Este peer lo primero que hace es unirse al grupo NetPeerGroup, donde se encuentra el peer de escucha. Acto seguido, lo que hace es invocar al servicio de descubrimiento que ofrece el grupo NetPeerGroup con la finalidad de encontrar el anuncio de la pipe. Para simplificar se ha especificado el anuncio que debe encontrar leyéndolo de un archivo (este anuncio es el mismo que tiene el peer de escucha).

Para encontrar la pipe y crear los bindings adecuados para establecer la comunicación, se recurre al servicio de descubrimiento local y remoto de anuncios. Una vez se haya encontrado el anuncio específico se crea la pipe.

Para crear la pipe, se ha usado un método basado en eventos. Una vez se crea el punto terminal de salida de mensajes (output pipe) lo que se hace es enviar un mensaje de texto al otro extremo.

El mensaje llega correctamente al otro extremo que permanece a la escucha y lo muestra por pantalla. Este extremo sigue a la escucha de nuevos mensajes.

En el peer de envío de mensajes, también se le ha dotado de la capacidad de encontrar peers rendezvous. Sin embargo, no se ha implementado esta opción a fondo.

A continuación se expone un esquema de lo descrito.

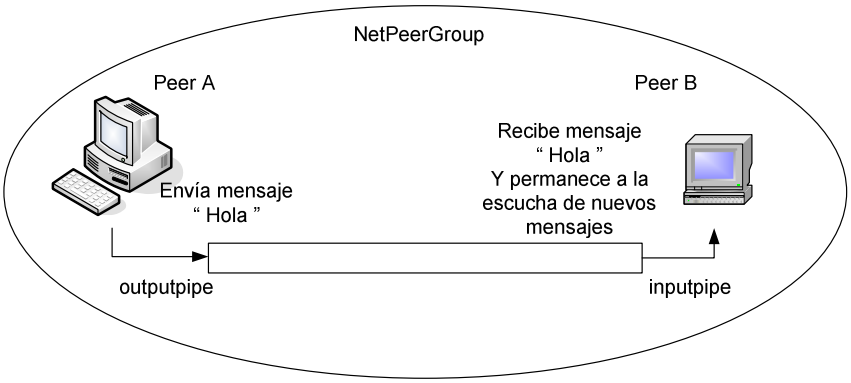



**Fig. 5.9** Comunicación mediante pipes.

Como paso siguiente a la implementación y desarrollo de peers, se ha ampliado la aplicación anterior. En este caso, el peer que permanecía a la escucha de mensajes dentro del grupo general NetPeerGroup ha creado un grupo propio (mediaNet) al que cualquiera puede unirse.

Como es rutinario, lo primero que hace este peer es unirse al NetPeerGroup. Acto seguido este peer, que a priori no sabe si está creado un grupo denominado de la misma manera que el que desea crear (mediaNet), intenta unirse a él. Para lograrlo se ayuda del servicio de descubrimiento y primero analiza su caché local a ver si encuentra el anuncio, ante la negativa lo busca remotamente. Si no lo encuentra es porque no existe el grupo, por lo tanto lo crea. Para crear el grupo y darlo a conocer se ha creado un anuncio y esta vez sí que ha sido propagado por la red.

La creación de la pipe sigue el mismo proceso que el descrito anteriormente, pero esta vez se realiza sobre el grupo creado. Esto implica que sólo se pueda comunicar con peers que se hayan unido al grupo al que se pertenece (mediaNet).

Posteriormente el peer ya está listo para permanecer a la escucha.

Por lo que respecta al peer que antes enviaba mensajes, se ha tenido que modificar para que se una al grupo nuevo creado. La implementación se ha llevado a cabo de la misma manera que el peer de escucha. El aspecto más

destacable es que éste busca y encuentra automáticamente el anuncio del grupo nuevo (mediaNet).

```
Attempting to discover the mediaNet Peergroup
Found the mediaNet PeerGroup Advertisement
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta="http://jxta.org">
       <GID>
              urn:jxta:uuid-
4D6172676572696E204272756E6F202002
       </GID>
       <MSID>
              urn:jxta:uuid-
DEADBEEFDEAFBABAFEEDBABE000000010306
       </MSID>
       <Name>
              mediaNet
       </Name>
       <Desc>
              mediaNet, lab 333 blue tower Prueba integrada
       </Desc>
</jxta:PGA>

The SearchPeer joined the mediaNet PeerGroup
Reading in pipexample.adv
Attempting to create a OutputPipe
Waiting for Rendezvous Connection
 Got an output pipe event
Sending message
message sent
```

**Fig. 5.10** Peer emisor de mensajes dentro del grupo mediaNet.

Se muestra a continuación un esquema de lo descrito.

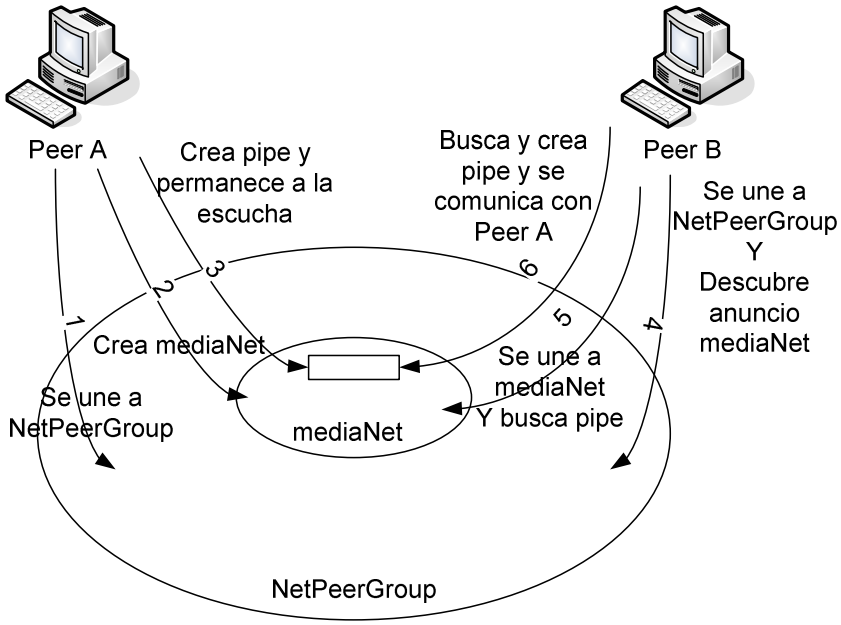



**Fig. 5.11** Creación y comunicación en el grupo mediaNet.

La última implementación que se ha realizado de esta arquitectura ha sido la creación de un servicio de calculadora sencillo que permite hacer sumas, restas, multiplicación y división de dos números. El resultado se devuelve al peer cliente.

Para ello se ha creado un servicio de peer (peer service), implementado y ofrecido por un solo peer.

Este peer confecciona un anuncio de servicio con la especificación del servicio que ofrece.

Para ello se crea un anuncio de especificación (módulo de especificación) asociado al servicio. El módulo de especificación contendrá toda la información necesaria para que el cliente contacte con el servicio. Por ejemplo, contendrá un anuncio de pipe (creado de la misma manera que en la aplicación anterior).

El cliente tendrá que obtener el mismo anuncio para poderse comunicar con el servidor del servicio. Cuando el cliente descubre el anuncio del módulo, extraerá el anuncio de la pipe.

Una vez realizado esto, se ha publicado el anuncio del servicio tanto en la caché como remotamente y se ha creado el punto terminal de entrada (input pipe) para atender las peticiones. Una vez llega una petición, se abre un punto de salida (output pipe) para enviar el resultado al cliente.

El cliente se dedica a buscar el servicio que desea e implementar la especificación detallada en el anuncio del servicio. A partir de ésta, crea una pipe de salida y una de entrada. La de *salida* para indicar al servidor la operación y operandos de la misma y la de *entrada* para recibir el resultado de la operación.

```
Start the Client
searching for the JXTASPEC:JXTA-EX1 Service advertisement
..We found the service advertisement:
jxta:MSA :
      MSID : urn:jxta:uuid-
BB2D6FDB5F4E4F54B685DA1865A73EF5CD88AED6CDFC4BC6ACB9C0E6F4819CB006
      Name : JXTASPEC:JXTA-EX1
      Crtr : sun.com
      SURI : http://www.jxta.org/Ex1
      Vers : Version 1.0
      Desc : Prueba JXTA Services Servicio . Usage: hola +
operacion[suma/resta/multiplicacion/division] + numero1 + numero2
      jxta:PipeAdvertisement :
            Id : urn:jxta:uuid-
9CCCDF5AD8154D3D87A391210404E59BE4B888209A2241A4A162A10916074A9504
            Type : JxtaUnicast
            Name : JXTA-EX1

Trying to bind to pipe...
name pipe adv:JXTA-EX1
message "hola" sent to the Server
Esperando mensaje de respuesta
El resultado es 8
Good Bye ....
```

**Fig. 5.12** Anuncio recibido por el cliente y recepción de resultado.

El servidor permanece a la espera de recibir nuevas peticiones.

A continuación se muestra un esquema de lo descrito.

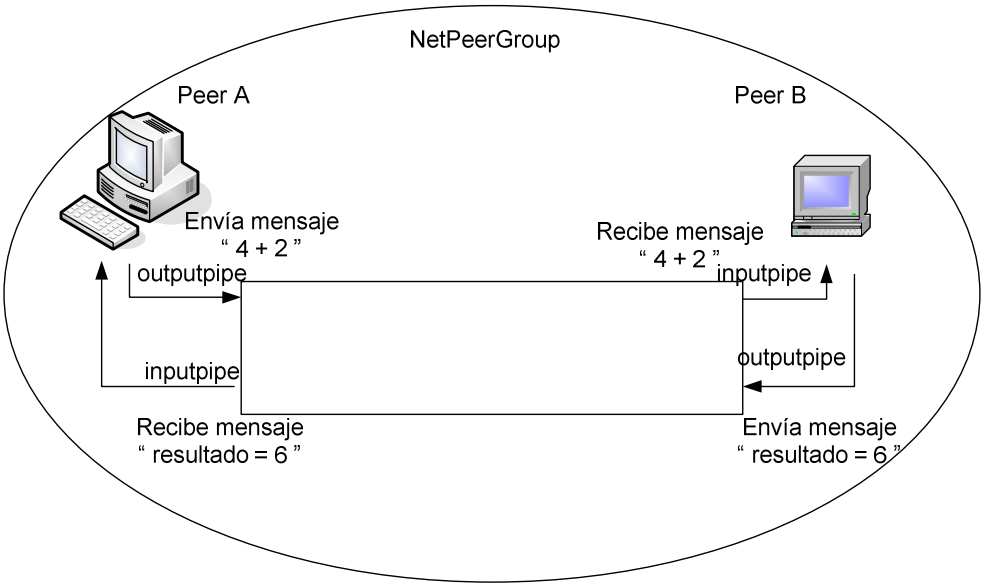



**Fig. 5.13** Servicio simple de calculadora.

# CAPÍTULO 6. PLANIFICACIÓN Y COSTES

En este capítulo se muestra la planificación realizada del proyecto y la estimación de costes que ha supuesto la realización de éste.

En el apartado de *planificación*, se muestra un diagrama de Gantt realizado con la herramienta Microsoft Project. Éste permite ver qué tareas se han desarrollado, en qué consisten, cuánto tiempo se le ha dedicado a cada una, cómo se relacionan y ordenan y qué herramientas se han utilizado para su realización.

En el apartado de *estimación de costes* se han tenido en cuenta varios aspectos para realizar la valoración. Se ha determinado el número medio de horas dedicadas a la realización del proyecto, se han tenido en cuenta los días totales consumidos para llevar a cabo el proyecto y se ha multiplicado por un valor que representa el precio de la hora estimado.

Finalmente, se ha valorado el coste del proyecto.

## 6.1. Planificación

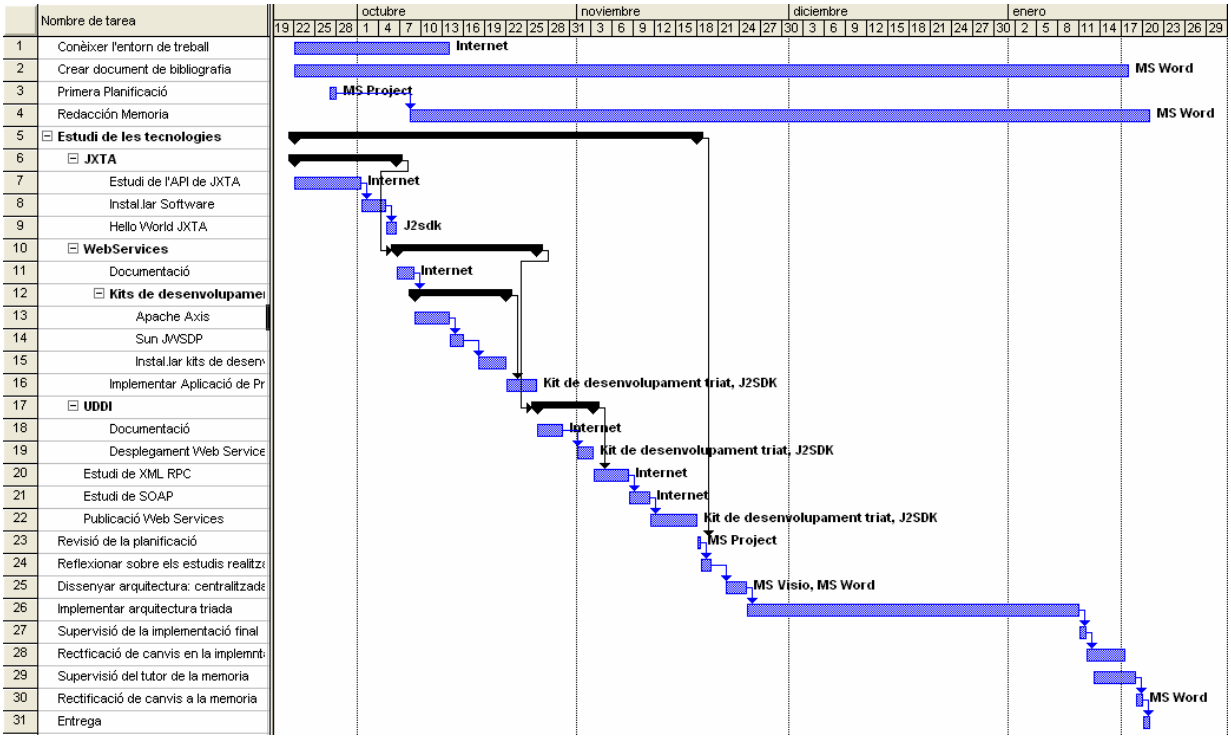



**Fig. 6.1** Diagrama de Gantt que muestra la planificación realizada para el desarrollo del proyecto.

Básicamente, la planificación se centra en:

- Conocer el entorno de trabajo.

  Familiarizarse con el entorno en el que se va a trabajar y las herramientas software que se utilizarán: servidor de ficheros CVS, Apache Ant, Apache Axis, Apache Tomcat, etc.

- Redacción de la memoria

  Se irá tomando nota de aspectos importantes a lo largo del estudio para incluir en la memoria.

- Estudio de las tecnologías

  Esta tarea se divide en varias subtareas detallando el tiempo estimado para cada una. El estudio de las diferentes tecnologías ayudará a crear un diseño de las arquitecturas que posteriormente se implementarán.

- Revisión de la planificación

- Reflexión sobre los estudios realizados

  Escoger las tecnologías estudiadas para proponer una arquitectura y diseño del sistema de localización.

- Diseño de la arquitectura: centralizada y distribuida

- Implementar arquitecturas

- Supervisión de la implementación final

- Rectificación de cambios en la implementación

- Supervisión del tutor de la memoria

- Rectificación de cambios en la memoria

En la revisión que se ha hecho de la planificación cabe destacar que se ha alterado el orden de estudio de las tecnologías. En la práctica se ha realizado primero el estudio de Servicios Web y posteriormente el de JXTA.

## 6.2. Estimación de costes

Para hacer una estimación de costes del proyecto se deben considerar varios aspectos.

Fechas de entrega: Detalladas en el apartado anterior (6.1. Planificación).

El tiempo total para realizar el proyecto (1 cuatrimestre) es reducido para implementar con profundidad ambas arquitecturas. Se ha realizado un estudio previo de tecnologías y diseño para definir los pasos a seguir en la implementación.

Hitos logrados: Los hitos logrados corresponden a los objetivos definidos.

Se ha desarrollado una arquitectura centralizada basada en Servicios Web.

Se ha desarrollado una serie de aplicaciones sencillas comprobando el funcionamiento de los protocolos básicos de las redes P2P basadas en JXTA.

Riesgos: Riesgo asociados al proyecto.

La complejidad inicial del proyecto se basaba en el desconocimiento de las tecnologías a usar.

El proyecto tiene un tamaño grande pues se pueden implementar gran cantidad de funcionalidades, existen muchas tecnologías y componentes interactuando entre sí.

Personal: Un estudiante de Ingeniería Técnica de Telecomunicaciones, especialidad en Telemática que realiza el proyecto dentro del Trabajo de Fin de Carrera y, por tanto, no cobra sus servicios, forma parte de su formación. El estudiante está bajo la supervisión de un tutor.

La dedicación media dedicada al desarrollo del proyecto ha sido de 6 horas diarias.

Herramientas y equipos: Éste es el apartado más destacable. El proyecto se ha realizado utilizando como sistema operativo Windows XP Profesional. Este sistema operativo de pago implica tener una licencia para su uso.

Sin embargo, cabe destacar que se podría haber usado un sistema operativo libre y gratuito como es Linux. Todas las tecnologías usadas son compatibles.

El entorno de programación ha sido Eclipse Project, gratuito.

Se ha necesitado un servidor Apache Tomcat, un kit de desarrollo de Servicios Web denominado Apache Axis, una base de datos XML Apache Xindice y una base de datos MySQL. Todo este software es gratuito, de libre distribución.

Los ordenadores empleados han sido dos. Se puede usar una sola máquina pero resulta más cómodo emplear dos equipos, sobretodo para la implementación de la arquitectura distribuida.

Se ha usado un hub de 8 puertos de 3Com para conectar las dos máquinas usadas.

Por lo que respecta al coste de los equipos, software empleado, electricidad y acceso a Internet la universidad es quien lo facilita. Realizar una estimación del coste de estos recursos es complejo pues la universidad ya lo asumió en la adquisición de los equipos. Estos equipos se rehúsan para otros proyectos y trabajos. Por lo tanto, no se puede especificar un valor económico de los recursos consumidos por el proyecto.

Coste: Una vez observados estos aspectos, no se puede estimar un coste específico y riguroso del proyecto. Sin embargo se puede decir que el coste es asumible y no muy elevado. Éste se centra principalmente en la formación que recibe el estudiante.

El coste del proyecto, si se atiende al mercado laboral, se puede aproximar a lo siguiente suponiendo que éste se hubiera becado:

- Horas diarias dedicadas: 6
- Días totales de realización del proyecto: 87
- Precio estimado de la hora: 4,0 €

    - Coste: 2088 €

# CAPÍTULO 7. CONCLUSIONES

## 7.1. Objetivos cumplidos

El objetivo principal del proyecto era solventar el problema de localización de recursos multimedia en una red centralizada y distribuida para seguidamente implementar los escenarios diseñados.

Se ha conseguido llevar un estudio de las tecnologías que se pueden emplear para el montaje de los escenarios centralizado y distribuido. Además, se ha llevado acabo la definición de cada una de las arquitecturas con detalle de sus componentes y posteriormente se ha realizado un diseño para saber cómo se pueden implementar.

El objetivo básico ha sido alcanzado satisfactoriamente.

No obstante, por lo que se refiere a la implementación de las arquitecturas que diseñadas no se ha completado totalmente. En la arquitectura centralizada ha faltado tiempo e información para implementar el registro privado UDDI basado en jUDDI. En parte, esto se ha debido a la poca información oficial que existe pues está en fase de elaboración según se puede comprobar en la página Web de Apache jUDDI[4].

Por lo que respecta al escenario distribuido se ha podido implementar una red virtual JXTA de peers básicos que se descubren, comunican y publican sus servicios. Sin embargo, no se ha completado el desarrollo de super peers.

## 7.2. Conclusiones de las arquitecturas implementadas

Vistas las diferentes arquitecturas, se puede decir que la que aporta funcionalidades más amplias y potentes es la arquitectura distribuida. Ésta permite publicar cualquier servicio o recurso de manera automática, siguiendo unos estándares bien definidos. La arquitectura centralizada requiere de un almacén único donde albergar la información de los recursos, aspecto que limita mucho la escalabilidad y flexibilidad del sistema en comparación con la visión distribuida.

Una de las ventajas más destacables de la red distribuida es que un peer especializado en un determinado contenido puede gestionar independientemente su información. También puede sacar el máximo provecho de la red, pues ésta le informa dinámicamente de lo que en ella puede encontrar.

Una desventaja de la arquitectura distribuida es que no se puede asegurar que la información obtenida de la red es completa debido a su

[4] http://ws.apache.org/juddi/

heterogeneidad y cambio constante e indeterminista. Cosa que en Servicios Web, al haber un elemento centralizador, dispone la misma información actualizada para todos los clientes.

Para solventar el problema de localización de recursos multimedia se propone la arquitectura distribuida, basándose en el potente estándar de Java para la creación de redes P2P denominado JXTA. Éste framework ofrece una gama de funcionalidades muy amplia y el hecho que aún esté en continuo desarrollo seguro que depara nuevas e interesantes utilidades.

## 7.3. Mejoras y ampliaciones futuras

A la hora de realizar este proyecto han existido ciertas limitaciones de tiempo y conocimientos principalmente. Estas limitaciones han hecho que el proyecto no haya llegado a ser tal y como se esperaba al inicio por lo que a las funcionalidades descritas respecta, aún a pesar de haber cumplido los objetivos. A continuación se citarán las posibles mejoras y ampliaciones futuras que se consideran más importantes

### 7.3.1. Posibles mejoras

Por lo que respecta al código implementado para ambas arquitecturas, se puede reorganizar el código de manera más lógica. Las clases implementadas se pueden reorganizar. Un caso claro es el de la definición de la clase recurso empleada en el servlet para determinar la estructura de datos que conforman un recurso almacenado en la base de datos. Esta clase actualmente incorpora métodos que se podrían implementar en otra clase dedicada a la gestión de recursos.

Otro aspecto que se debía haber probado a la hora de implementar los Servicios Web, ha sido el paso de objetos definidos por el usuario o tipos de datos complejos. Simplemente se ha experimentado con tipos de datos sencillos como cadenas de caracteres, números enteros, etc.

Mejorar la apariencia del entorno Web, adaptándolo adecuadamente al estilo de la página de I2Cat.

### 7.3.2. Ampliaciones futuras

Queda pendiente implementar un registro UDDI privado para el escenario centralizado.

En el escenario distribuido se debe implementar los nodos adecuadamente según el diseño de red basada en super peers realizado.

Otro aspecto común para ambas tecnologías es la adopción del framework MPEG 21. La implementación realizada ha adoptado el concepto general de

éste, pero no se basa rigurosamente en él pues no se ha seguido la definición DIDL para describir un recurso y su información (objeto digital).

Se pueden también emplear protocolos de seguridad compatibles para cada arquitectura, a modo de ofrecer servicios seguros.

Finalmente, si se observan las funcionalidades implementadas, son las más básicas posibles. Se pueden implementar muchísimas más, tal y como se detalla en la especificación (Capítulo 2, apartado 2.1).

## 7.4. El proyecto y el medio ambiente

Hoy en día, el impacto medioambiental de un proyecto es un aspecto muy importante a tener en cuenta y en muchas ocasiones es clave para definir la viabilidad de un proyecto y, por tanto, su aprobación o rechazo.

Es complejo encontrar el impacto medio ambiental que una aplicación software pueda suponer, sin embargo en la sociedad si que provoca un impacto relevante.

Como se puede observar, actualmente se intenta ofrecer un gran número de servicios fácilmente accesibles por individuos de todo el mundo a través de la red.

Uno de las aportaciones más notorias que ha supuesto esta aparición de servicios en red es que cualquier persona desde su oficina o vivienda puede acceder a ellos sin necesidad de desplazarse. Esto en muchas ocasiones supone un ahorro de tiempo y esfuerzo considerable a la hora de solicitar un servicio o contactar con una compañía o individuo. Estos servicios ofrecidos a través de la red permiten hacer con mayor eficiencia lo que antes ya se hacía sin el servicio en red.

Las comunicaciones electrónicas permiten transportar en millonésimas de segundo datos que sin ésta tecnología se moverían mucho más lentamente.

Gracias a servicios como la videoconferencia se puede establecer una reunión entre personas que se encuentran separadas físicamente y repartidos por el mundo y separados por grandes distancias físicas. Este servicio permite que las personas no se tengan que desplazar de su puesto de trabajo habitual para llevar a cabo la reunión, con la consecuencia de cuantiosos gastos de transporte y contaminación ambiental.

El servicio de localización de contenido multimedia distribuido en una red permite la compartición de esos recursos de manera dinámica con cualquier usuario autorizado.

El hecho de poder disponer de contenido multimedia ya supone un ahorro de soportes físicos importante. Por ejemplo, si se dispone de una foto ya no se genera un soporte en papel, un video ya no se almacena en una cinta de video,

etc. Actualmente existen soportes digitales que permiten almacenar numerosos recursos multimedia en un espacio reducido, especialmente si se emplean métodos de compresión.

Por lo que respecta a la localización automática de recursos a través de la red supone un ahorro de esfuerzos para la persona que solicita un recurso, facilitando y haciendo más eficiente su labor.

Otro aspecto a comentar en consonancia a la situación actual de las polémicas redes de compartición de contenido digital es que la difusión del contenido puede afectar especialmente al colectivo que tiene derechos de propiedad sobre un recurso. Esto supone un impacto directo en la responsabilidad social i exigencias éticas asociadas a estas tecnologías. Estos derechos deben ser gestionados adecuadamente para que no se haga un uso fraudulento de los recursos. Sin embargo, este tema se puede controlar mediante la implantación del framework MPEG 21, aunque actualmente sigue en fase de desarrollo y estandarización.

## 7.5. Conclusiones personales

La realización de este proyecto ha sido muy satisfactoria e interesante. El tema desarrollado encuentro que es muy atractivo, además supone el estudio de tecnologías muy punteras en el sector.

El aspecto más destacable ha sido la cantidad de conocimientos teóricos adquiridos sobre las tecnologías estudiadas y las herramientas usadas. El estudio ha sido denso pero muy enriquecedor.

No sólo he adquirido conocimientos teóricos sino que también he aprendido a adoptar ciertas actitudes a la hora de buscar y seleccionar la información útil, a planificar la realización del proyecto desde el principio y redactarlo siguiendo unas pautas de presentación.

Este cuatrimestre mi principal ocupación ha sido el proyecto. Durante su realización han sido muchas las penas y alegrías vividas. La frustración que surge cuando algo no funciona como deseas o no encuentras información que te oriente o ayude resulta a veces desesperante y eso se nota en la vida diaria. Sin embargo, he tenido la fortuna de tener un tutor dedicado y accesible que me ha orientado y ayudado en todo lo que le he preguntado, además de un compañero y amigo con el que se hacía más amena la estancia en el laboratorio y con el que solventaba dudas sobretodo en temas de programación. Este compañero era el encargado de desarrollar el módulo de provisión de servicios para el Proyecto Integrado.

Una de las cosas que más aprecio y que me ha enseñado el desarrollo del Trabajo de Fin de Carrera ha sido a superar dificultades y a afrontar los problemas. Problemas acentuados a menudo por la falta de conocimientos de programación o desconocimiento de las tecnologías. Sin embargo, la

motivación e ilusión por aprender cosas nuevas e intentar hacer las cosas bien y que funcionen ha sido el arma para salvar las desavenencias.

# CAPÍTULO 8. BIBLIOGRAFÍA

## 8.1. Libros consultados

[1] Scott Oaks, Bernard Traversat, Li Gong, *JXTA in a Nutshell a Desktop Quick Reference*, O'REILLY, 2002.

[2] Ethan Cerami, *Web Services Essentials*, O'REILLY, 2002.

[3] Barbara McKee, David Ehnebuske, *Providing A Taxonomy For Use In UDDI Version2 UDDI Working Draft Best Practices Document*, 2001.

[4] Joseph D. Gradecki , *Mastering JXTA: Building Java Peer-to-Peer Applications*, John Wiley & Sons, 2002.

[5] Eric M. Burke, "*Java and XSLT*", O'REILLY, 2001.

## 8.2. Artículos

[6]Graham Glass, "*Register and publish your Web Service*",ibm.com/developerWorks.

[7]"*Project JXTA: Getting Started*", Sun Microsystems, 2001.

[8] "*Project JXTA v2.0 Java Programmer's Guide"*, Sun Microsystems, 2003.

[9] Toni Oller, Jesús Alcober, Pedro Díaz, Alberto J. González, "*Projecte Integrat i2Cat, versión 0*", 2004

[10] Hideki Sakamoto, "*Introduction of MPEG-21Activities and DII&D (Digital Item Identification and Description=ISO W4532 )*", cIDF, 2002.

[11] Ian Burnett, Rik Van de Walle, Keith Hill, Jan Bormans, Fernando Pereira, "*MPEG-21:Goals and Achievements*", IEEE, 2003.

[12] Jane Thacker, "*ISO/IEC TR 18034-1 Information technology — Multimedia framework (MPEG-21)*", ISO/IEC, 2003.

[13]ISO/IEC TR 21000-1:2001(E), "*Information technology — Multimedia framework (MPEG-21), Part 1:Vision, Technologies and Strategy* ", ISO/IEC, 2001.

[14]Thomas DeMartini, Chris Barlas, Jean-Claude Dufourd, "*INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION*

*INTERNATIONALE NORMALISATION ISO/IEC JTC 1/SC 29/WG 11 CODING OF MOVING PICTURES AND AUDIO*", ISO/IIEC, 2004.

[15] Frederik De Keukelaere, Christian Timmerer, Gerrard Drury and Xin Wang, "*ISO/IEC 21000-8/FCD MPEG-21 reference software*", ISO/IEC, 2004.

[16] R. Van de Walle, I. Burnett, G. Drury, "*ISO/IEC 21000-2 DID 2nd Edition FCD*", ISO/IEC, 2004.

[17] Rik Van de Walle, Ian Burnett, Gerrard Drury, Munchurl Kim, and Viswanathan (Vishy) Swaminathan, "*ISO/IEC CD 21000-10 – Part 10: Digital Item Processing, ISO/IEC*", 2003.

[18] Ismael Caballero Muños-Reja, "*La norma IEEE1058.1: Plan para la Gestión de Proyectos de Software*", 1999.

[19] Ricardo, Borillo Domenech, "*Primeros pasos con XML y XSL*".

[20] Leitan Rong, Ian Burnett, "*Dynamic Resource Adaption in a Heterogeneous Peer to Peer Environment*".

## 8.3. Enlaces web

[21] UDDI.org, *UDDI 2.0 Specification*, URL <http://www.uddi.org/specification.html>

[22] MPEG 21 Validation chain. URL <http://www2002.org/CDROM/poster/175/>

[23] Sun Microsystems. *Java TM 2 Platform, Standard Edition, v1.4.2 API Specification* [En línea]. Página Web, URL <http://java.sun.com/j2se/1.4.2/docs/api/>

[24] Programación en castellano (2004) *Tutorial de java*, URL <http://www.programacion.com/java/tutorial/new2java1>

[25] James Clarck, "XSL Transformations XSLT version 1.1", W3C, URL <http://www.w3.org/TR/xslt11/>, 2001.

[26] XML Encryption Standard, URL < http://www.w3.org/Encryption/>

[27]SOAP Security Extension: Digital Signature, URL <http://www.w3.org/TR/SOAP-dsig/>

[28] Organitzation for the Advancement of Structured Information Standards (OASIS), URL < http://www.oasis-open.org/committees/security/>

[29] Especificación de protocolos JXTA "JXTA v2.0 Protocols Specification"

” URL < http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html >

[30] XML Path Language (Xpath) v 1.0, W3C, 1999, URL <http://www.w3.org/TR/xpath>

[31] Página web de proyectos para Servicios Web Apache, incluye jUDDI, Axis, SOAP, XML-RPC, etc. “*Web Services Project @ Apache*”, URL < http://ws.apache.org/>.

[32] Página oficial del proyecto JXTA, “*JXTA Project*”, URL <http://www.jxta.org/>.

[33] Página oficial del Servidor Apache Tomcat, “*Apache Jakarta Project*”, URL <http://jakarta.apache.org/tomcat/index.html>.

[34] Especificación WSDL, “*Web Services Description Language 1.1*”, URL <http://www.w3.org/TR/wsdl>.

[35]Especificación SOAP, “SOAP version 1.2”, URL <http://www.w3.org/TR/soap/>.

[36] Xalan Java Version 2.6.0, Xalan es un procesador XSLT basado en Java para documentos XML basado, URL <http://xml.apache.org/xalan-j/index.html>.

[37] On Java.com, “*Creating Web Services with Apache Axis*”, URL <http://www.onjava.com/pub/a/onjava/2002/06/05/axis.html?page=2>.

[38] Fundació I2Cat, URL <http://www.i2Cat.net>.

# ANEXO 1. Transcodificación de protocolos

La función de la transcodificación de protocolos se centrará en adaptar los diferentes protocolos propietarios de los media processors de contenido audiovisual al protocolo estándar.

El objetivo del módulo de transcodificación de protocolos es el de proporcionar señalización de servicios a usuarios mediante protocolo H323, SIP o incluso HTTP/SOAP, para permitir el acceso a los servicios de transporte de media processors.

El módulo de transcodificación de protocolos se compone de los siguientes elementos:

- Control de la señalización
- Transcodificación de protocolos de control
- Transcodificación de protocolos de control a protocolos de transporte y localización

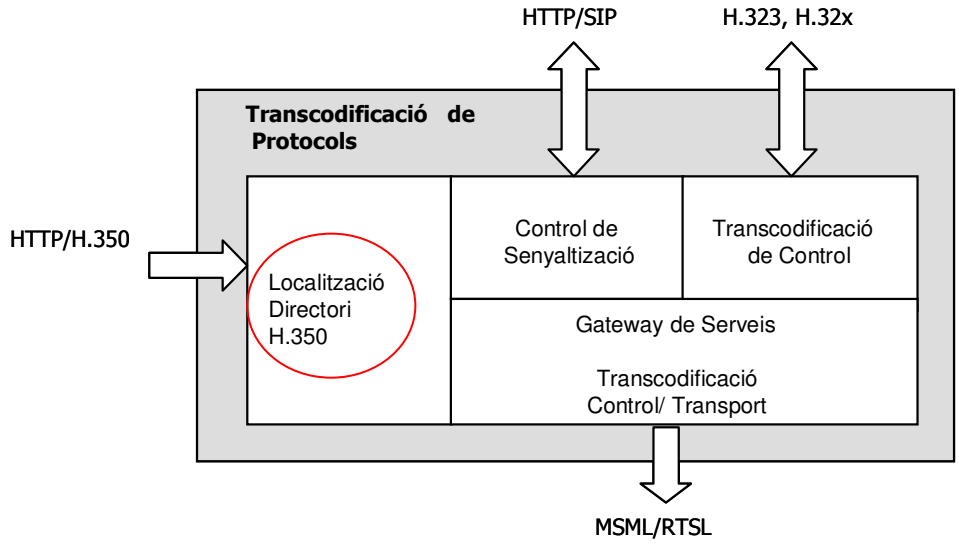



**Fig. 1.1** Transcodificador de protocolos.

El módulo de transcodificación de protocolos interactuará con el módulo de transcodificación de contenidos para ofrecer un servicio combinado de almacenaje, distribución en tiempo real e interactividad multimedia, de manera que un media processor puede, por ejemplo, ser invitado a participar en una videoconferencia.

La infraestructura multimedia de la plataforma integrada atiende la petición de servicios mediante SIP, el cliente interactúa con la plataforma a través de terminales hardware o Servicios Web. La interacción contempla la comunicación punto a punto en servicios de multiconferencia.

En una arquitectura centralizada, un directorio H350, LDAP o un registro UDDI o una plataforma basada en JXTA para una arquitectura distribuida permitirá la localización de estos servicios, así como la autenticación de los usuarios de manera inequívoca.

El gateway de protocolos entre el plano de transporte y control permite la integración de una gama variada de media processors en el Proyecto Integrado.

El gateway de servicios contempla 2 implementaciones posibles:

- Interfaz de control genérica para media processors.
- Interfaz de control para servicios orientados a media almacenada.

La interfaz de control genérica será implementada a partir del protocolo MSML, que proporciona el soporte para el control de capacidades de procesado de media. La implementación contempla el desarrollo de conectores específicos para cada tipo de control de transporte estándar como: media processors DV, media processors alta definición, media processors que soporten RTSP, etc.

# ANEXO 2. Servicios Web

## 2.1. Introducción a los Servicios Web

Un servicio web es cualquier servicio disponible y accesible a través de la red. Los mensajes que intercambia siguen una estandarización XML y son independientes de cualquier sistema operativo o lenguaje de programación.


**Fig. 2.1** Diagrama Intercambio de datos.

Hay diversas alternativas a los mensajes XML, por ejemplo: XML RPC (XML Remote Procedure Call), SOAP, incluso se podrían hacer peticiones GET/POST mediante HTTP y devolución de documentos XML.

Aparte de estas características, se pueden añadir 2 más que aunque no son obligatorias facilitan el uso de los Servicios Web. Éstas son las siguientes:

- Auto-describibles: es conveniente publicar una interfaz pública. El Servicio publicado debería incluir una documentación fácilmente entendible por un usuario (persona humana), a fin de que éste pueda

usar el servicio. Por ejemplo, si se crea un servicio descrito mediante SOAP, sería útil definir en una interfaz pública escrita en XML simple la especificación de los métodos, argumentos y retornos implementados.

- Localizables: si una persona crea un Servicio Web, debe haber un mecanismo sencillo para publicarlo y por tanto que otros usuarios lo puedan descubrir y usar. Este mecanismo puede ser descentralizado o centralizado.

Los Servicios Web están pensados para ser servicios ofrecidos por aplicaciones a otras aplicaciones (no a personas, usuarios) sobre la red. Esto no elimina la actuación del usuario, sólo especifica que la conversación se puede dar directamente entre aplicaciones, tan fácilmente como entre un navegador web y un servidor web.

Se pueden diferenciar 3 actores en Servicios Web: el proveedor, el cliente y el registro. Éstos serán explicados más adelante.

Una de las principales ventajas que ofrecen los Servicios Web es la separación de las capas de presentación de la de contenido. Por ejemplo, un sitio web puede estar hecho únicamente a base de contenedores web que pasan parámetros a los Servicios Web. Esto hace que sea fácil cambiar la presentación de un sitio web y la compartición entre usuarios de Servicios Web.

## 2.2. Panorama empresarial de Servicios Web

Actualmente, hay diversos frameworks y propuestas para el desarrollo de Servicios Web. Los principales impulsores son Microsoft's.NET, IBM Web Services y Sun Open Net Environment (ONE). Cada uno tiene sus específicos kits de desarrollo de Servicios Web, los cuales se basan en las tecnologías básicas que lo definen: SOAP (protocolo de comunicación), WSDL (descripción de los servicios exportados) y UDDI (registro para la localización de los servicios).

Muchos de los kits de desarrollo son propietarios, pero existen algunos de libre distribución[5].

[5] Como por ejemplo el kit de desarrollo de Apache denominado AXIS. Con este kit se desarrollarán los Servicios Web en este proyecto.

## 2.3. Arquitectura de Servicios Web

Se puede ver la arquitectura de los Servicios Web de 2 maneras: a partir del papel que desempeña cada actor o a partir de la pila de protocolos que los componen.

Actores en Servicios Web:

- Proveedor del servicio: es el actor que implementa el servicio y lo hace disponible cara a la red.
- Cliente: es el actor que consume el servicio habilitado. Éste utiliza un Servicio Web existente enviando una petición XML (XML request). Debe escribir un cliente que cumpla los requisitos especificados por el Servicio Web para poder usar los métodos implementados.
- Registro del servicio: se trata de un directorio centralizado de servicios donde se pueden publicar (por parte del proveedor), retirar o encontrar (por parte de cualquier actor) Servicios Web o servicios en general.

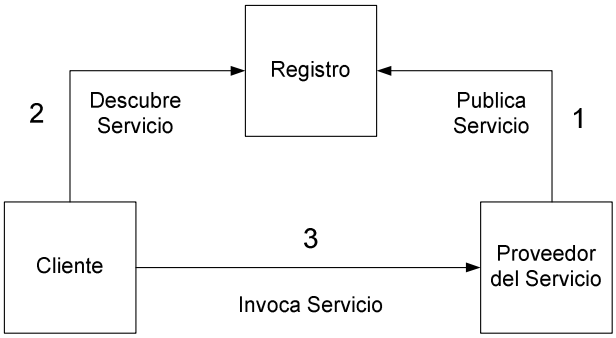



**Fig. 2.2** Interacción entre los diferentes actores.

Pila de protocolos:

Se pueden estructurar según cuatro capas principales.

- Transporte: esta capa es la encargada de transportar mensajes entre aplicaciones. Ésta admite principalmente los protocolos HTTP (Hyper Text Transfer Protocol), SMTP (Simple Mail Transfer Protocol) y FTP (File Transfer Protocol).
- Mensajería XML [Anexo 2, apartado 2.4]: esta capa es la responsable de codificar los mensajes intercambiados a XML para que todos los extremos puedan entenderlos. Esta capa incluye SOAP [Anexo 2, apartado 2.6] y XML-RPC [Anexo 2, apartado 2.5].
- Descripción: esta capa es la responsable de describir la interfaz pública de un Servicio Web, se basa en Web Service Description Language (WSDL) [Anexo 2, apartado 2.7].
- Descubrimiento: esta capa es la encargada de centralizar servicios en un registro común y de proporcionar un método simple para poder publicarlos y encontrarlos. Actualmente se usa el Universal Description, Discovery and Integration (UDDI) [Anexo 2, apartado 2.8].

No obstante, a medida que los Servicios Web evolucionan, más capas serán añadidas y nuevas tecnologías para cada una de ellas.

## 2.4. Mensajería XML (Extensible Markup Language)

XML[6] se ha introducido fuertemente en el mundo de la computación en los últimos años. Ha sido rápidamente aceptado pues permite a diversas computadoras compartir información de manera más sencilla, estructurar datos de manera simple y eficiente y es independientemente del sistema operativo o lenguaje de programación que se esté usando. Hay muchas utilidades para trabajar con XML, tales como parseadores o editores, disponibles para cualquier sistema operativo o lenguaje de programación.

XML es similar a HTML (Hypertext Markup Language), tiene elementos, atributos y valores. Los documentos XML pueden ser visualizados por navegadores web.

HTML contiene un número finito de elementos y atributos, mientras que XML permite cualquier número de ellos. Se muestra a continuación un ejemplo:

```
<libro>
      <autor>Aldous Houxley</autor>
      <titulo>Un Mundo Feliz</título>
      <precio>19,95</precio>
      <referencia>ah1234</referencia>
</libro>
```

**Fig. 2.3** Ejemplo de documento XML.

Después de representar la información en formato XML, se puede crear un schema para validar la información, para asegurar que tiene la correcta estructura y tipos de datos. Así, si se envía un documento XML al Servicio Web, se puede comprobar la corrección de la información incluida. A continuación se muestra un ejemplo de un esquema de validación del documento XML anterior.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="libro" type="libroType"/>
      <xsd:complexType name="libroType">
      <xsd:sequence>
      <xsd:element name="libroAutor"
      type="xsd:string"/>
      <xsd:element name="libroTitulo"
```

[6] Versión 1.0: Se encarga de definir elementos, atributos y tags englobado dentro de un elemento raíz y proporciona un modelo de estructurar la información y serialización de la misma.

```
        type="xsd:string"/>
        <xsd:element name="libroPrecio"
        type="xsd:float"/>
        <xsd:element name="libroReferencia"
        type="xsd:string"/>
        </xsd:sequence>
        </xsd:complexType>
</xsd:schema>
```

**Fig. 2.4** XML Schema de validación del ejemplo anterior.

Las dos APIs de programación más populares para parsear XML son:

- Document Object Model (DOM)
- Simple API for XML (SAX)

La principal diferencia entre ambos es que DOM proporciona un modelo de objeto genérico para representar la estructura (árbol) de los documentos y un conjunto de métodos estándares para manipularlos. Éste va leyendo elemento a elemento y da varias vueltas al documento para pasearlo, funciona como una base de datos o depósito que debe ser consultado varias veces. Sin embargo, SAX es más rápido y eficiente, usa menos memoria. En función del uso que se haga del documento XML uno es mejor que el otro, así por ejemplo si lo que se quiere es leer el documento una única vez para pasarle datos a una aplicación será más eficiente SAX, pero si por el contrario se desea usar el documento como una continua fuente de información o como objeto con el que interaccionar varias veces, DOM será más adecuado.

Otro aspecto importante y útil es la transformación de un documento XML. Se puede mapear y transformar un documento de un formato a otro. La especificación Extensible Stylesheet Language Transformation (XSLT) es muy usada para esta labor, aunque supone una mayor carga computacional al sistema.

XML define como insertar estructuradamente información en un documento, así pues, es tan importante insertarla como posteriormente extraerla y por lo tanto, poderla pasar de un formato a otro. XSLT permite la separación entre los datos y la información.

XSLT es una parte de XSL (Extensible Stylesheet Language) que fue desarrollado para transformar documentos XML en presentaciones para pantalla, papel o expresión oral. XSLT tiene una amplia aplicación en Servicios Web, especialmente para transformar XML en otro formato.

XSLT necesita de un parseador, SAX o DOM. XSLT aplica una hoja de estilo XSLT (una plantilla que contiene las instrucciones para transformar dicho documento a cualquier otro formato) a un documento XML para producir el documento resultante, transformación del entrante. Ésta es la manera por ejemplo usada para pasar XML a HTML.


**Fig. 2.5** Procesado XSLT. Transformación de XML a HTML.

Cuando un desarrollador decide construir un sistema de mensajería para Servicio Web, es muy habitual escoger XML.

Existen principalmente 2 contenedores de XML: XML-RPC y SOAP.

## 2.5. XML- Remote Procedure Call (XML-RPC)

XML-RPC es un protocolo que usa mensajes XML para invocar a procedimientos remotos. Las peticiones se codifican en XML y se envían mediante HTTP POST. La respuesta XML se encuentra dentro del cuerpo de la respuesta HTTP. El hecho de ser independiente de la plataforma permite la comunicación de diversas aplicaciones.

A continuación se expone un ejemplo de invocación remota. Se envía una petición con el código postal de una ciudad y devuelve el nombre de la ciudad (las cabeceras HTTP son omitidas). La petición consiste en una etiqueta *methodcall* que especifica el nombre del método y los parámetros a enviar. La respuesta consiste en un tag *methodResponse* que especifica el valor y tipo de retorno.

Petición:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
      <methodName>info.getNameCity</methodName>
      <params>
            <param><value>08100</value></param>
      </params>
</methodCall>
```

**Fig 2.6** Petición XML-RPC.

Respuesta:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
      <params>
            <param><string>Mollet del
      Valles</string></param>
      </params>
</methodResponse>
```

**Fig. 2.7** Respuesta XML-RPC.

XML-RPC es la manera más sencilla para empezar con Servicios Web. En muchos aspectos es más sencillo que SOAP.

## 2.6. Simple Object Access Protocol (SOAP)

SOAP fue desarrollado por UserLand, DevelopMentor y Microsoft.

SOAP es un protocolo de intercambio de información entre computadores basado en XML. Es una extensión de XML-RPC.

Aunque puede ser usado en múltiples sistemas de mensajería y puede ser transportado sobre una gran variedad de protocolos de transporte, lo usual es usar SOAP para invocación remota (RPC) y viajar sobre HTTP. Al igual que XML-RPC, SOAP es independiente de la plataforma y sirve para comunicar diferentes aplicaciones.

Otros frameworks como CORBA o Java RMI proporcionan funcionalidades similares, sin embargo los mensajes SOAP están escritos enteramente en XML y son independientes de plataforma y lenguaje.

Para entenderlo mejor, ver el ejemplo anterior representado mediante SOAP.

Petición:

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://www.w3.org/2001/09/soap-envelope/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <SOAP-ENV:Body>
      <ns1:getNameCity
            xmlns:ns1="urn:examples:infoservice"
            SOAP-
      ENV:encodingStyle="http://www.w3.org/2001/09/soap-
      encoding/">
      <zipcode xsi:type="xsd:int">08100</zipcode>
      </ns1:getNameCity>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Fig. 2.8** Petición SOAP.

Se puede observar como la petición SOAP es más compleja y detallada que la XML-RPC. Usa XML namespaces[7] y XML-Schemas[8]. En este caso también se especifica el método y valor.

Respuesta:

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
```

[7] Los identificadores únicos para los elementos y aplicaciones del documento XML. Son útiles para evitar errores en duplicación de nombres.

[8] Documentos XML que definen tipos de datos a usar, contenido, estructura y elementos permitidos. Sirve para validar mensajes XML.

```
        xmlns:SOAP-ENV="http://www.w3.org/2001/09/soap-envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <SOAP-ENV:Body>
        <ns1:getNameCityResponse
              xmlns:ns1="urn:examples:infoservice"
              SOAP-
        ENV:encodingStyle="http://www.w3.org/2001/09/soap-
        encoding/">
        <return xsi:type="xsd:string">Mollet del Valles</return>
        </ns1:getNameCityResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Fig. 2.9** Respuesta SOAP.

La respuesta explicita que devuelve una cadena de caracteres.

Las principales partes del mensaje SOAP son:

- Sobre (envelope): define el principio y final del mensaje.
- Cabecera (header): contiene cualquier atributo opcional del mensaje a usar en el procesado del mismo (tanto en un punto intermedio por donde tenga que pasar como en el punto final de recepción).
- Cuerpo (body): contiene la información en XML del mensaje enviado.
- Adjuntos (attachements): documentos adjuntos al mensaje principal.
- Interacción RPC: define como modelar las llamadas RPC a SOAP.
- Codificación (encoding): define como representar información simple o compleja transmitida en el mensaje.

Generalmente, los mensajes SOAP contienen sobre (obligado), cabecera (opcional) y cuerpo (obligado).

La especificación SOAP define 3 partes principales:

- Especificación del sobre SOAP (SOAP envelope). Define reglas específicas para encapsular la información que se intercambiará entre computadoras. Esto incluye el tipo de datos que necesita una aplicación, nombre del método a invocar, parámetros del método o tipos de datos a retornar. También incluye información a cerca de quién debe procesar el contenido del sobre o en el caso de error, cómo codificar mensajes de error.
- Reglas de codificación de información (Data encoding rules). Para intercambiar información las máquinas deben acordar qué reglas van a usar para codificar los datos. SOAP incluye una serie de especificaciones para determinar el tipo de datos a usar, se basan en la especificación XML Schema creada por el W3C.
- Invocación a procedimientos remotos (RPC): SOAP puede ser usado de muchas maneras para intercambiar mensajes, incluyendo mensajes simples que no precisan respuesta o mensajes de petición - respuesta. Para los mensajes de petición – respuesta, SOAP especifica una convención para hacer llamadas a procedimientos y respuestas. Esto permite a una aplicación de un cliente especificar el nombre del método

a demandar, incluir los parámetros necesarios para usarlo y recibir la respuesta del servidor.

SOAP no está ligado a ningún protocolo de transporte específico, de hecho se puede transportar vía SMTP, FTP, etc. Sin embargo, la especificación determina detalles para HTTP solamente, por ello HTTP se mantiene como el protocolo más popular a la hora de trasportar SOAP.

Las peticiones SOAP son enviadas en una petición HTTP y las respuestas SOAP son devueltas también en el contenido de la respuesta HTTP. Mientras que las peticiones SOAP se pueden enviar mediante HTTP GET, la especificación tan solo se centra en POST (se prefiere POST pues en la mayoría de servidores se limita el nombre de caracteres que puede tener una petición GET). Además, las peticiones y respuestas HTTP deben especificar su contenido como text/xml.

Como requisito adicional, se debe especificar una cabecera SOAPAction. Esta cabecera es una URI específica para un servidor, para indicar la intención de la petición. Esto hace posible determinar rápidamente la naturaleza de la petición SOAP sin tener que examinar la carga del mensaje SOAP. En la práctica, la cabecera SOAP es usada por cortafuegos para bloquear peticiones SOAP o para dirigir rápidamente los mensajes SOAP al servidor correspondiente.

El valor de SOAPAction depende de la implementación del servicio. Por ejemplo, para acceder a un servicio llamado InfoCity ubicado en broadband6 se debe especificar la `urn:broadbandInfoCity#InfoCity` como SOAPAction. Incluso si el servidor no requiere una SOAPAction completa, el cliente debe especificar una cadena de caracteres vacía (“”) o un valor nulo:

SOAPAction: “”
SOAPAction:

En el ejemplo siguiente muestra una cabecera HTTP usando el SOAPAction anterior:

```
POST /cgi/soaplite.cgi HTTP/1.0
Host: broadband.upc.es
Content-Type: text/xml; charset=utf-8
Content-Length: 538
SOAPAction: " urn:broadbandInfoCity#InfoCity "
//mensaje SOAP
```

**Fig. 2.10** Cabecera de petición HTTP POST transportando mensaje SOAP.

La cabecera de respuesta sería algo como lo siguiente:

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2001 15:01:55 GMT
Server: Apache/1.3.31 (Windows) tomcat/1.0 PHP/4.0.1pl2
SOAPServer: SOAP::broadband/cgi
Cache-Control: s-maxage=60, proxy-revalidate
Content-Length: 539
Content-Type: text/xml
```

```
            //mensaje SOAP
```

**Fig. 2.11** Cabecera de respuesta HTTP transportando mensaje SOAP.

El código 200 en la respuesta indica que la respuesta es correcta mientras que un código de valor 500 indicaría error (Internal Server Error) y en el mensaje SOAP aparecería un elemento *Fault* indicándolo.

## 2.7. Web Service Description Language (WSDL)

WSDL[9] fue desarrollado por Microsoft, IBM y Ariba y la versión 1.1 fue presentada al W3C el cual lo aceptó. Actualmente cuenta con amplia difusión y se puede afirmar que prácticamente con unanimidad la comunidad de Servicios Web lo acepta como el estándar de facto para la definición de Servicios Web.

WSDL es una gramática XML (XML Schema) para especificar la interfaz pública del Servicio Web. Ésta puede especificar los métodos implementados por el servicio, tipo de datos a intercambiar en cada mensaje XML, información de ubicación del servicio, protocolo de transporte a usar, etc. Ofrece una manera común de definición para la comunicación entre aplicaciones. WSDL no está necesariamente ligado a un sistema de mensajería XML específico, pero incorpora extensiones empotradas (embeded) para describir servicios SOAP. WSDL es el formato XML que describe en qué consiste un Servicio Web.

Los Servicios Web están comúnmente implementados usando lenguajes de programación diseñados para la interacción con el Web, como es Java o Java Server Pages (JSP).

A continuación se puede ver un ejemplo de WSDL siguiendo con el ejemplo mostrado. Nos fijaremos básicamente en 2 puntos, el primero es en el elemento *message* el cual especifica los diferentes mensajes XML que se transmiten entre computadoras y el segundo es el elemento *service* el cual especifica la disponibilidad del servicio vía SOAP en la ruta *http://localhost:8080/soap/servlet/service*.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="InfoService"
      targetNamespace="http://www.ecerami.com/wsdl/WeatherService.wsdl
"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:tns="http://www.ecerami.com/wsdl/InfoService.wsdl"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

[9] Schema WSDL: http://schemas.xmlsoap.org/wsdl/. Especificación WSDL: http://www.w3.org/TR/wsdl.

```
      <message name="getInfoServiceRequest">
      <part name="zipcode" type="xsd:int"/>
      </message>
      <message name="getInfoServiceResponse">
      <part name="city" type="xsd:string"/>
      </message>
      <portType name="InfoService_PortType">
      <operation name="getInfoService">
      <input message="tns:getInfoServiceRequest"/>
      <output message="tns:getInfoServiceResponse"/>
      </operation>
      </portType>
      <binding name="infoServiceSoapBinding"
type="tns:InfoService_PortType">
      <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="getInfoService">
      <soap:operation soapAction=""/>
      <input>
      <soap:body
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:examples:infoservice"
      use="encoded"/>
      </input>
      <output>
      <soap:body
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:examples:infoservice"
      use="encoded"/>
      </output>
      </operation>
      </binding>
      <service name="Info Service ">
      <documentation>WSDL File for Info Service</documentation>
      <port binding="impl:infoServiceSoapBinding"
name="InfoService_Port">
      <soap:address
      location="http://localhost:8080/soap/servlet/service"/>
      </port>
</service>
</definitions>
```

**Fig. 2.12** Archivo InfoService.wsdl.

Usando WSDL, un cliente puede localizar un Servicio Web e invocar cualquiera de sus métodos anunciados.

La especificación WSDL debe definir fundamentalmente:

- Especificación de todos los métodos hechos públicos en la interfaz.
- Tipo de datos para todos los mensajes de petición y de respuesta.
- Información de binding para saber el protocolo de transporte a usar.
- Dirección para localizar un servicio específico

Los elementos WSDL contienen una descripción de la información que se le debe pasar al Servicio Web con el fin que el emisor y el receptor entiendan los datos intercambiados. El archivo WSDL también contiene la descripción de las operaciones implementadas (métodos) con el fin de que el receptor sepa

procesar los datos recibidos. La especificación del tipo de protocolo a usar o transporte sirve para que el emisor de los datos sepa cómo enviarlos.

Dentro del documento .wsdl se encuentran estos elementos:

- Definiciones (definitions): éste es el elemento raíz principal de cualquier documento WSDL. Define el nombre del Servicio Web, declara múltiples namespaces y contiene todos los elementos necesarios para la descripción del servicio.
- Tipo de dato (data-type): tipo de dato, basado en XML Schema u otros (Ej. CORBA), a usar en los mensajes intercambiados.
- Mensaje (message): definición abstracta de un mensaje, descrito por campos con el fin de ser mapeados según una invocación de un método implementado. Especifica los parámetros a pasar a un método y los valores de retorno de éste.
- Operación (operation): indica el método especificado en un mensaje que atenderá el Servicio Web.
- Tipo de puerto (port-type): se indica para saber exactamente dónde será atendida la petición. Una petición se puede enviar mediante diferentes protocolos de transporte.
- Binding: el protocolo concreto de transporte y formato de información para los mensajes y operaciones de cada tipo de puerto.
- Puerto (port): especifica dónde está el servicio, por dónde se comunica.
- Servicio (servicio): especifica la dirección donde se encuentra el Servicio Web. Normalmente se especifica una URL para invocar al servicio SOAP.

Afortunadamente, un documento WSDL puede ser generado automáticamente usando herramientas para la creación de Servicios Web, kits de desarrollo. Éstos traducen el código de los métodos implementados a un XML schema determinado (archivo .wsdl).

## 2.8. Universal Description, Discovery and Integration (UDDI)

UDDI fue presentado por Microsoft, IBM y Ariba en septiembre de 2000. Estas empresas fundaron UDDI.org e invitaron a otras muchas a participar en este proyecto, actualmente integrado por más de 300 empresas, que definieron sus especificaciones iniciales.

UDDI representa una especificación para la publicación, encuentro e integración de Servicios Web. Para este proyecto se ha estudiado la API para java de Apache denominada jUDDI, pero no se ha llegado a implementar.

El registro UDDI puede ser público, es decir, se puede publicar un servicio en el registro UDDI público de las empresas operadoras que lo ofrecen para dar a conocer los servicios de una empresa o la empresa en sí misma. También puede ser privado, para publicar servicios en una red propia, en este caso se puede usar una base de datos privada (por ejemplo una base de datos relacional MySQL).

El registro UDDI público funciona de una manera similar a un Domain Name Server (DNS). Las empresas pueden registrarse en uno de los registros que facilitan IBM, HP, SAP o Microsoft y la información que faciliten será almacenada en una base de datos. Cada cierto tiempo la información se replica a las otras bases de datos de los otros registros, asegurando además que toda la información es la misma, está sincronizada. De esta manera un cliente puede obtener la misma información accediendo a cualquiera de los registros públicos. Si se desea actualizar información de un servicio publicado, la empresa debe acceder al registro original y ejecutar la función de actualización.

La seguridad es uno de los aspectos más importantes. Las empresas están autorizadas a actualizar información del registro por cualquiera de los operadores pero después deben acceder al operador original para borrar o eliminar información. Las empresas que desean registrarse en UDDI deben obtener primero una autorización y deben ser aprobadas por al menos uno de los operadores. Las aprobaciones son una clave que se envía a las empresas para que ellas accedan (Log) y depositen la información. La información de acceso a un operador no funcionará para otro. Una empresa debe escoger un operador y atarse a él.

La calidad y la validez de la información es otro aspecto muy importante. Se debe asegurar que las empresas registradas son reales y por lo tanto que su nombre, identificador (ID), teléfono, datos geográficos, categoría, etc. sean correctos.

Las dos partes principales de UDDI son el registro y el descubrimiento. El registro se refiere a que una empresa puede publicar información para que otras lo puedan localizar y descubrir.

Los usuarios del registro pueden interaccionar con UDDI usando unas API's específicas de SOAP o usando, si se dispone, de las interfaces de usuario que facilitan los operadores. Los operadores crean descripciones de sus Servicios Web mediante WSDL para su registro y descubrimiento.

La información almacenada en el registro se divide en 3 categorías principales:

- Páginas Blancas: información general que identifica a una compañía específica que publica un servicio web (nombre, descripción, dirección, etc.).
- Páginas Amarillas: clasificación de la compañía y del servicio mediante taxonomías[10].

[10] Nombres de etiquetas que sirven para definir y clasificar la información. Los sistemas taxonómicos compatibles con UDDI son, por el momento, los ofrecidos por North American Industry Classification System (NAICS), Universal Standard Products and Services Codes (UNSPSC), ISO 3166, Standard Industry Classification (SIC) y GeoWeb Geographic Classification. Por lo tanto, seleccionaremos las categorías que representan de forma más acertada a una empresa o un servicio ofrecida por ésta. NAICS se usa para el código de la empresa, UNSPSC se usa para catalogar el producto o servicio de una empresa e ISO 3166 para códigos geográficos.

- Páginas Verdes: información técnica del Servicio Web (descripción y manera de invocación del Servicio Web). Si existe una referencia a un documento WSDL se incluye en este apartado. Se debe especificar un identificador único para cada servicio.

La información de una empresa se organiza según los siguientes tipos de datos estructurados:

- businessEntity: es el elemento principal. Describe la empresa cuya información va a ser registrada.
  A continuación se muestra un ejemplo de definición XML para el elemento businessEntity:

```
<element name = "businessEntity">
      <complexType>
            <sequence>
                  <element ref = "discoveryURLs" minOccurs = "0"/>
                  <element ref = "name" maxOccurs = "unbounded"/>
                  <element ref = "description" minOccurs = "0"
                  maxOccurs = "unbounded"/>
                  <element ref = "contacts" minOccurs = "0"/>
                  <element ref = "businessServices" minOccurs = "0"/>
                  <element ref = "identifierBag" minOccurs = "0"/>
                  <element ref = "categoryBag" minOccurs = "0"/>
            </sequence>
            <attribute ref = "businessKey" use = "required"/>
            <attribute ref = "operator"/>
            <attribute ref = "authorizedName"/>
      </complexType>
</element>
```

**Fig. 2.13** Ejemplo de elemento BusinessEntitiy

El schema define un elemento complejo *sequence* constituido por nombres de atributos y calificativos que especifican si esos nombres pueden estar repetidos o son requeridos.

- business Service: descripción y nombre del servicio a publicar.
- bindingTemplate: información del servicio. Incluye una dirección para acceder a él. Aquí se incluyen datos tales como: mailto, HTTP, https, ftp, fax, teléfono y otros. La información debe estar en el formato adecuado para cada tipo de datos de cada campo, es decir, el campo mailto deberá incluir una dirección válida de correo electrónico, HTTP una URL correcta, etc. Se puede así especificar diferentes maneras de acceder a un servicio.
- tModel: una marca o colección de información que identifica inequívocamente a un servicio y explicita su especificación. Es el mecanismo de intercambio de meta información de un Servicio Web, como su descripción o lugar de ubicación del archivo WSDL.

  tModel sería el equivalente en UDDI al WSDL, pero a diferencia puede incluir referencias a servicios usando otras direcciones no necesariamente en formato URL y permite especificar transportes diferentes a SOAP.

Una de sus funciones principales es la de facilitar búsquedas en el registro para un servicio específico. La API de UDDI soporta búsquedas específicas de servicios.

- publisherAssertion: permite especificar relaciones entre diferentes businessEntity. Permite definir si son departamentos, contratas, etc.

Cuando la información se almacena por primera vez, el operador asigna un ID único[11] a cada estructura de datos.

No se especifica el formato en el que se debe almacenar la información en el registro, por lo que la manera en que se envíe la información al operador no presupone la manera en la que será almacenada.

El uso de un registro UDDI también puede ser privado. En una red interna se puede disponer de un almacén de datos en el que publicar los servicios, productos o aplicaciones que se desee. Mediante las API de UDDI también se puede acceder a este almacén para publicar, localizar y obtener los servicios deseados.

## 2.9. Seguridad

El aspecto de la seguridad en Servicios Web es crítico. Sin embargo, ni XML-RPC ni SOAP determinan ningún requerimiento de autenticación o seguridad.

No obstante, la comunidad de Servicios Web ha propuesto numerosos frameworks y protocolos de seguridad pero aún debe consensuar estos aspectos.

De manera general, la seguridad se centra en la confidencialidad, la autenticación y la seguridad en red.

Confidencialidad: su objetivo es asegurar la confidencialidad de la comunicación establecida entre cliente (envía petición XML) y servidor.

Afortunadamente, tanto SOAP como XML-RPC pueden viajar sobre HTTP y por lo tanto la comunicación puede ser encriptada mediante Secure Socket Layer (SSL), resultando HTTPS. SSL es una tecnología probada y ampliamente usada para la encriptación de mensajes.

Este aspecto, por otro lado, presenta desventajas. Uno de los principales beneficios de los Servicios Web es que éstos pueden ser usados por otros Servicios Web, creando una cadena de aplicaciones. En este caso, SSL no

[11] Tiene forma de UUID (Universal ID) el cual deriva de la Open Software Foundation Distributed Computing Environment; formalizada como ISO/IEC 11578: 1996 Information Technology—Open Systems Interconnection—Remote Procedure Call (www.iso.ch/cate/d2229.html).

resulta adecuado pues los mensajes necesitan ser encriptados en cada nodo de la cadena y esto supone un riesgo. Actualmente no existe ningún acuerdo para solucionar este aspecto pero se confía en que el estándar de la W3C, XML Encryption Standard[12], lo sea. Este estándar propone un framework para la encriptación y desencriptación de documentos XML enteros o tan sólo partes de ellos.

Autenticación: su objetivo es identificar correctamente a un usuario y saber si está autorizado para usar el servicio.

Una solución es integrar la autenticación en HTTP. HTTP incluye soporte para autenticar de manera básica. Sin embargo, no es una solución sólida pues los Servicios Web están accediendo constantemente a aplicaciones y almacenes de datos y , por tanto, cierta información importante puede ser pasada de un servicio web a otro sin riguroso control.

Al igual que sucede con la encriptación, no hay un consenso a cerca de cómo conseguir autenticación, pero existen varios frameworks que se están estudiando. El primero es SOAP Security Extensions: Digital Signature (SOAP-DSIG)[13]. Éste propone el uso de claves públicas para que el cliente o el servidor comprueben la identidad del otro extremo. En segundo lugar, está la propuesta de la Organization for the Advancement of Structured Information Standards (OASIS)[14] la cual está trabajando en Security Assertion Markage Language (SAML), diseñado para facilitar el intercambio de información de autenticación y autorización.

Seguridad en red: su objetivo es impedir la invocación ilícita de procedimientos de un computador.

SOAP es transportado sobre HTTP y permite a clientes invocar comandos y procedimientos remotos, aspecto que los cortafuegos pretenden impedir.

Si se intenta, por lo tanto, filtrar mensajes que contengan SOAP o XML-RPC una posibilidad es filtrar todas las peticiones HTTP POST que establecen su contenido (content type) como text/xml (requisito para las dos especificaciones). Otra opción es filtrar por la cabecera HTTP SOAPAction. Actualmente los distribuidores de cortafuegos están desarrollando herramientas para filtrar el tráfico de los Servicios Web.

## 2.10. Apache Axis: kit de desarrollo de Servicios Web

Axis es un framework abierto de Apache que permite crear clientes y servidores SOAP, para la creación de Servicios Web. En este proyecto se ha estudiado la implementación en Java pero también se está desarrollando una implementación en C++.

[12] http://www.w3.org/Encryption/
[13] http://www.w3.org/TR/SOAP-dsig/
[14] http://www.oasis-open.org/committees/security/

Se ha elegido este kit de desarrollo pues es gratuito, tiene una API para Java, es de fácil uso y está ampliamente usado en el mundo de los Servicios Web.

Axis incluye:

- Un servidor Stand Alone
- Un servidor que puede interactuar con otros servlets tipo Tomcat.
- Soporte para WSDL
- Generador de clases a partir de WSDL (WSDL2Java)
- Generador de WSDL a partir de clases (Java2WSDL)
- Desplegador de Servicios Web (deploy, undeploy)
- Ejemplos
- Una herramienta de monitorización de paquetes TCP/IP

Axis es la tercera generación de Apache SOAP, que empezó con IBM con el nombre de SOAP4J.

Axis tiene las siguientes características:

- Velocidad. Axis usa SAX.
- Flexibilidad. El usuario tiene total libertad para añadir extensiones al kit de desarrollo.
- Estabilidad.
- Independiente del transporte.
- Soporte para WSDL
- Soporte para todos los tipos de datos básicos
- Extensiones de seguridad
- Soporta HTTP como protocolo de transporte

Mediante este kit de desarrollo se pueden crear las descripciones WSDL de los servicios que a implementar de manera automática para después desplegarlo (publicarlo localmente) en el servidor de aplicaciones que esté instalado, en nuestro caso Tomcat, para que cualquiera pueda acceder a éste.

Una vez ya ha sido generado el archivo WSDL que contiene la descripción del servicio web a implementar, se debe generar los bindings para que se pueda acceder a éste antes de desplegarlo. Mediante la herramienta WSDL2Java se generan automáticamente los siguientes archivos:

- NombreClase_BindingImpl.java: Archivo Java que contiene la implementación por defecto del Servicio Web a implementar. Éste se debe editar una vez generado para añadir la implementación del servicio.
- NombreClase.java: Contiene los usos de java RMI.
- NombreClase_Service.java: Contiene la interfaz de la parte del cliente.
- NombreClase_ServiceLocator.java: Contiene la clase que se implementa por parte del cliente

- NombreClase_SoapBindingSkeleton.java: Skeleton de la parte del servidor.
- NombreClase_SoapBindingStub.java: Stub de la parte del cliente.
- Deploy.wsdd: Descriptor que sirve para desplegar el servicio web
- Undeploy.wsdd: Descriptor que sirve para retirar el servicio web.
- Se pueden generar archivos para otros tipos de datos necesarios para el servicio web.

Existe otra manera muy simple de crear Servicios Web sencillos. Mediante un archivo JWS. Simplemente se renombra el archivo .java a .jws. Los métodos de la clase serán pues los del servicio. Con Axis instalado se puede acceder vía web directamente al archivo, se muestra un archivo WSDL equivalente. Se instala e implementa el servicio automáticamente. Sin embargo, esta alternativa tiene numerosas desventajas tales como que sólo se pueden especificar tipos de datos sencillos o que no se pueden incluir packages.

# ANEXO 3. JXTA

## 3.1. Introducción a JXTA

JXTA[15] es un conjunto de protocolos Peer to Peer (P2P) abiertos que permiten a cualquier dispositivo conectado a la red (teléfono móvil, PDA, PC, etc.) colaborar y comunicarse como peers. Los protocolos JXTA son independientes de lenguaje de programación. Existen diversas implementaciones para diferentes entornos, conocidas como JXTA bindings, pero en este trabajo nos centraremos en la versión de JXTA Project para plataformas Java 2 Standard Edition (J2SE).

A medida que las redes se hacen más grandes tanto en contenido como en número de dispositivos, las aplicaciones P2P se están volviendo cada vez más populares. Éstas incluyen aplicaciones como intercambio de archivos, cálculo distribuido o servicios de mensajería instantánea. Aunque cada aplicación cumpla con funciones totalmente diferentes, todas ellas tienen aspectos comunes tales como el descubrimiento de los diferentes peers, búsqueda o transferencia de datos. JXTA pretende aportar una serie de protocolos para realizar estas funciones de manera estándar.

Las ventajas principales que ofrece JXTA son:

- Interoperabilidad. JXTA está diseñado para permitir a los diferentes peers que ofrecen diferentes servicios localizarse y comunicarse.

[15] Abreviatura de Juxtapose (yuxtapuesto), motivado por el hecho de romper con el modelo cliente-servidor tradicional.

- Es independiente de plataforma. JXTA es independiente del lenguaje de programación y protocolo de transporte.
- Acceso. Se puede acceder desde cualquier dispositivo conectado a la red.

JXTA pretende estandarizar la manera en que los peers se descubren entre si, se organizan en grupos, anuncian y descubren servicios en la red y se comunican entre ellos.

JXTA permite que en una red heterogénea se puedan descubrir diferentes peers mediante un protocolo de descubrimiento dinámico (incluso a través de cortafuegos), compartir archivos con cualquiera, crear grupos de peers que ofrecen un servicio, monitorizar las actividades de los peers remotamente y mantener comunicaciones seguras con otros peers.

## 3.2. Arquitectura JXTA

El proyecto JXTA se divide en 3 capas:

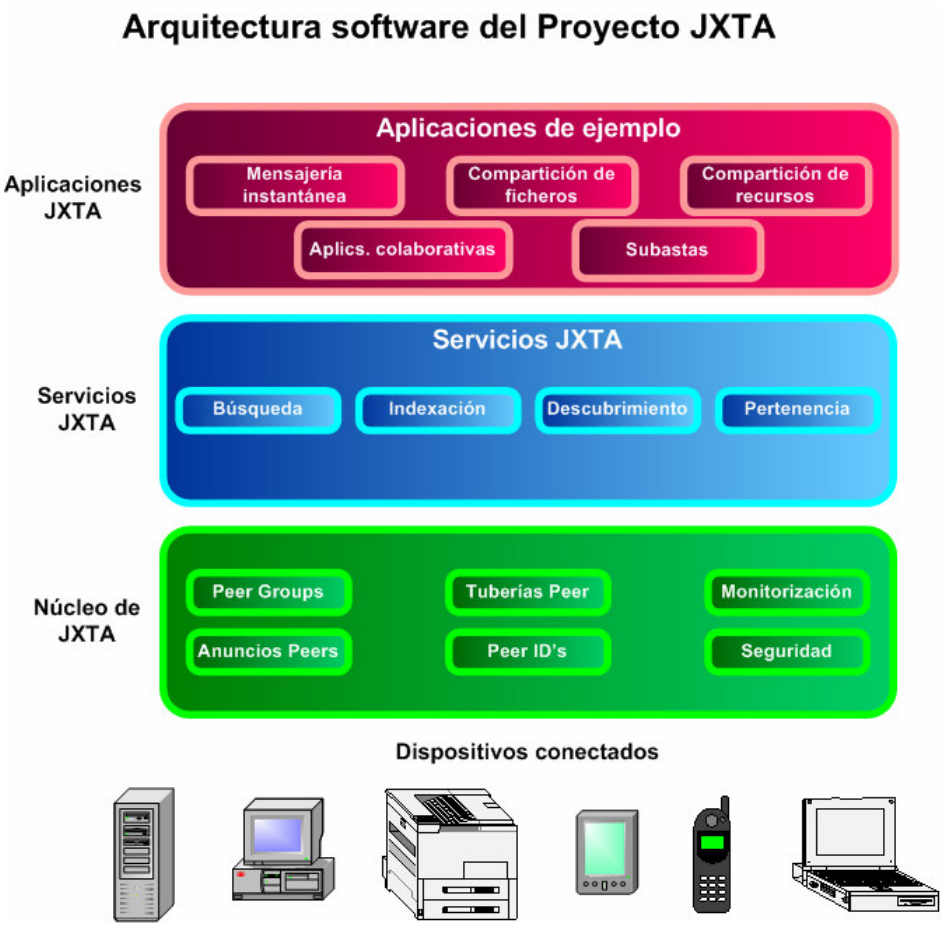



**Fig. 3.1** Capas de la arquitectura JXTA.

El núcleo de JXTA está compuesto por las rutinas esenciales que son comunes para la programación distribuida. Existen mecanismos para el descubrimiento,

transporte (incluido el manejo si hay cortafuegos), la creación de peers y grupos de peers y demás primitivas que implementan seguridad.

La capa de servicios incluye servicios que no tienen porqué ser imprescindibles en una red P2P, pero por norma común son muy útiles.

La capa de Aplicación incluye la implementación de las aplicaciones específicas.

### 3.2.1. Componentes JXTA

Una red JXTA está compuesta por una interconexión de nodos, o peers. Los peers se pueden agrupar en grupos de peers.

A continuación se muestra un esquema que representa la red virtual JXTA que se genera. Los peers se comunican directamente independientemente de la arquitectura de red física, ésta es transparente para el usuario. La conectividad en red no depende de la localización física.

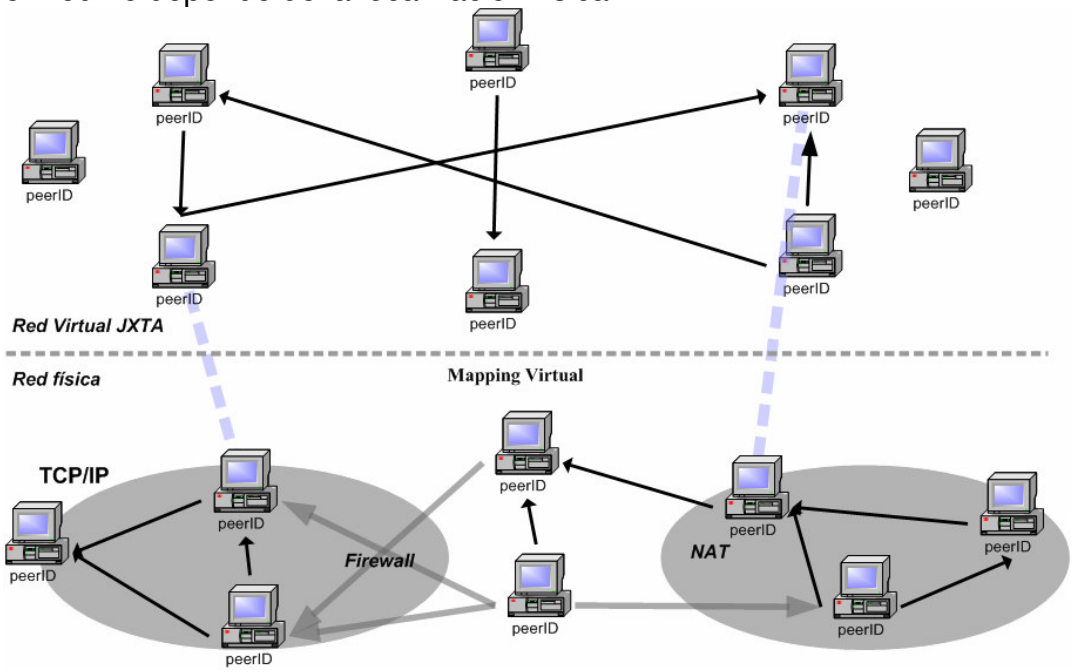



**Fig. 3.2** Red virtual JXTA.

Los peers JXTA anuncian sus servicios mediante documentos XML, denominados anuncios (advertisements). Éstos permiten a otros peers en la red aprender cómo conectarse e interactuar con los servicios publicados.

Los peers JXTA usan pipes para enviarse mensajes entre sí. Las pipes son un mecanismo de transmisión de mensajes asíncrono y unidireccional. Los mensajes son documentos XML sencillos cuyo sobre incluye información de enrutado y de identificación. Las pipes están asociadas a diferentes puntos terminales de comunicación como un puerto TCP asociado a una determinada dirección IP.

Los peers pueden hacer la función de cliente de un servicio o pueden ofrecer servicios, por lo tanto harían la función de servidor.

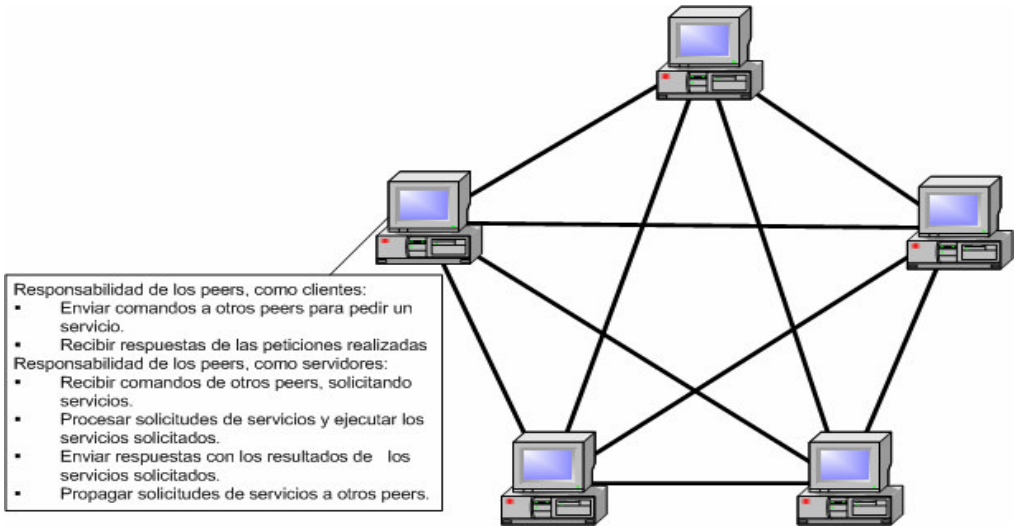



**Fig. 3.3** Funciones básicas de Peers y Servidores.

Cada peer y pipe dispone de un identificador único.

## 3.3. Conceptos básicos de JXTA

**Peer**: cualquier dispositivo en red que implementa uno o más protocolos JXTA. Cada peer funciona independientemente y asíncronamente de los demás y se identifica mediante un ID único.

Los peers identifican las interfaces de red como puntos terminales, los cuales son usados para establecer comunicaciones punto a punto entre 2 peers.

Los peers no necesitan tener una conexión directa punto a punto entre ellos. Los peers intermedios son usados para encaminar los mensajes que intercambian 2 peers separados tanto por conexiones físicas, como por configuración de la red (NAT, cortafuegos, proxies).

Los peers son configurados comúnmente para ser descubiertos espontáneamente entre sí y formar grupos de peers.

**Grupos**: son agrupaciones, colecciones, de peers que se unen para ofrecer un conjunto común de servicios. Cada grupo tiene su identificador único.

A continuación se muestra un esquema de representación de grupos. Éstos forman redes virtuales sobre la red física.

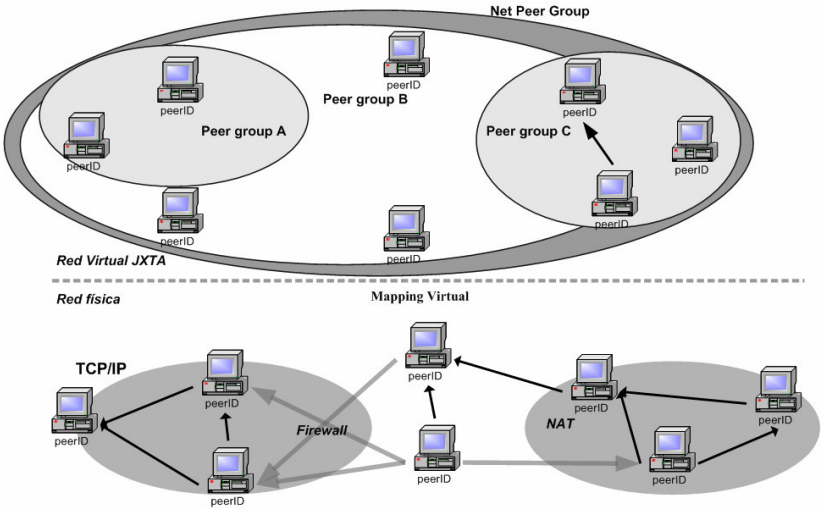



**Fig. 3.4** Grupos virtuales JXTA.

Cada grupo estipula su política de unión. Se puede adoptar la postura de permitir la adhesión de cualquier peer o una postura más segura y protegida requiriendo la introducción de una credencial (password por ejemplo).

Los peers pueden formar parte de más de un grupo a la vez. Por defecto, el primer grupo al que se une un peer es el denominado NetPeerGroup (representaría el mundo). Todos los peers pertenecen a este grupo y después tienen la libertad de juntarse a los otros grupos particulares que deseen.

Los protocolos JXTA describen cómo los peers deben publicar, descubrir o unirse a grupos.

La creación de un grupo viene motivada por el deseo de crear un entorno seguro, la limitación virtual de la red o delimitación de la misma, y la monitorización de peers.

Los grupos disponen de una serie de servicios propios implícitos definidos por JXTA, aunque adicionalmente se pueden crear los que uno desee.

Si 2 peers desean interactuar, deben pertenecer al mismo grupo.

Los servicios que se incluyen para los grupos son:

- Servicio de descubrimiento. Lo usan los peers para encontrar grupos, peers, recursos, pipes o servicios.

- Servicio de unión (membership). Lo usan los peers que son miembros de un grupo para elegir si aceptan o no a un candidato a unirse al grupo. Los peers que desean unirse a un grupo envían una petición a cualquier miembro. Ésta será aceptada o rechazada por el colectivo de miembros. El servicio fuerza una votación para impartir el dictamen.

- *Servicio de acceso.* Sirve para validar peticiones hechas por un peer y dirigidas a otro. No todas las acciones deben ser validadas, tan sólo aquellas limitadas para ciertos peers.

- Servicio de pipe. Sirve para crear y gestionar las comunicaciones de las pipes entre los peers de un grupo.

- Servicio de resolución. Se usa para enviar peticiones genéricas a otros peers. Los peers pueden definir e intercambiar mensajes para encontrar información como por ejemplo, el estado de un servicio o de un terminal de una pipe.

- Servicio de monitorización. Se permite a un peer monitorizar a otros miembros del grupo.

Cada grupo puede implementar los servicios que crea útiles.

**Módulos**: son una abstracción usada para representar piezas de código que definen comportamientos en el mundo de JXTA. Los servicios son un caso de comportamiento que los peers pueden instanciar. La especificación no determina qué es este código, puede ser una clase de Java, un .jar, un script, un DLL (dynamic library), un conjunto de documentos XML, etc. Los módulos pueden ser usados por ejemplo para representar diferentes implementaciones de un servicio web en diferentes plataformas.

Existen 3 tipos de módulos:

- Módulo de Clase. Anuncia la existencia de un comportamiento. Especifica que se requiere un comportamiento y también los bindings necesarios para soportar el módulo. Cada Módulo de clase tiene su ID propio.

- Módulo de Especificación. Se usa para acceder al módulo, contiene toda la información necesaria para invocarlo. En caso de un servicio por ejemplo, éste debe incluir un anuncio de una pipe. Cada Módulo de especificación tiene su propio ID.

- Módulo de Implementación. Representa la implementación de un módulo de especificación. Cada módulo de implementación contiene el ID del módulo de especificación que implementa.

Como ejemplo se puede considerar el Servicio de Descubrimiento. Este tiene un determinado ID de módulo de clase. Puede haber varias implementaciones del servicio, cada una incompatible con las otras. Éstas pueden basarse en la naturaleza de la red a la que van a ser aplicadas.

**Servicios**: los peers se comunican y cooperan para publicar, descubrir e invocar servicios. Los peers pueden publicar múltiples servicios y localizarlos gracias al Peer Discovery Protocol.

JXTA diferencia entre 2 tipos de servicios:

- Servicios ofrecidos por un peer (Peer Services). Es aquél que sólo lo suministra el peer que lo implementa, sólo lo instancia él. Por lo tanto, si el peer falla, el servicio también falla. Pueden publicarse diferentes instancias del servicio en diferentes peers (réplica) pero cada anuncio será diferente y se estará publicando un mismo servicio.

- Servicios ofrecidos por el grupo (Peer Group Services). Es aquel que está compuesto por una serie de instancias (que cooperan entre ellas) de un servicio corriendo en múltiples miembros del grupo. Si un peer falla, el colectivo de peers no resulta afectado (suponiendo que el servicio sigue activo en otro peer). Son anunciados como parte del anuncio del propio grupo.

**Pipes (tuberías)**: los peers usan pipes para enviarse mensajes. Soportan el transporte de cualquier objeto. Son asíncronas y unidireccionales.

Los terminales de una pipe se denominan input (punto de entrada) y output (punto de salida). Representan canales virtuales de comunicación.

Las pipes pueden ser:

- Punto a punto. Conecta dos terminales de peers, un input donde un peer recibe mensajes enviados por un output de otro peer. En la carga del mensaje aparece un campo ID para determinar la secuencia de mensajes.
- Propagadas. Conecta un output con múltiples input pipes.
- Unicast seguro. Es del tipo punto a punto pero ofreciendo un canal seguro.

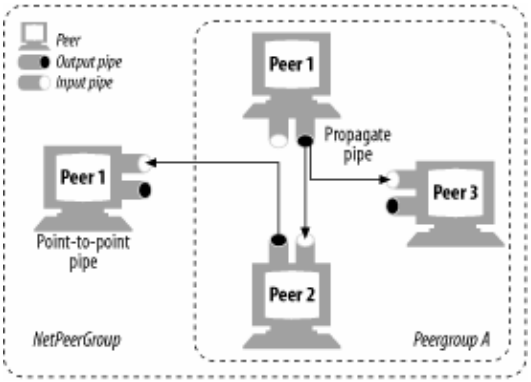



**Fig. 3.5** Modos de propagación.

Se pueden crear otros tipos de pipes, por ejemplo las bidireccionales.

A continuación se muestra un esquema de lo que simbolizaría una pipe. Permite comunicar a dos peers directamente a pesar de no estar conectados punto a punto el uno con el otro físicamente.

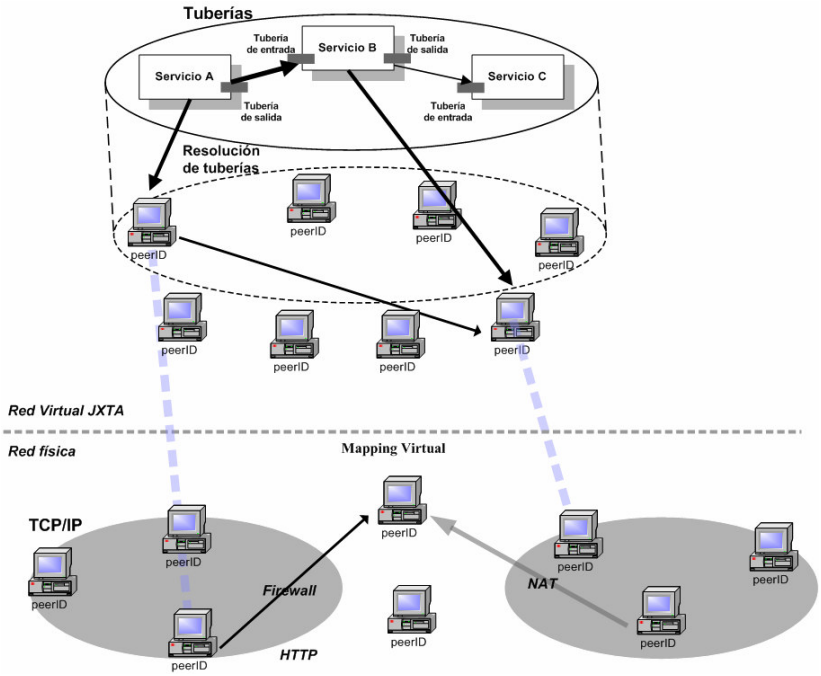



**Fig. 3.6** Comunicación entre peers mediante canales virtuales (pipes).

**Mensajes**: son los objetos intercambiados por las pipes entre peers. Se envían y se reciben gracias al Servicio de pipes y de terminal (end-point).

**Anuncios**: los peers, grupos, servicios y pipes son representados mediante anuncios. Son documentos escritos en XML.

Cada anuncio tiene un tiempo de vida limitado, esto permite que sean borrados aquellos anuncios obsoletos sin la intervención de un control centralizado. También pueden volverse a publicar.

Existen los siguientes tipos de anuncios.

- Anuncio de peer. Indica información del peer como el nombre, ID, terminales disponibles, etc.
- Anuncio de pipe. Describe el canal de comunicación de la pipe. Mediante el servicio de pipes, se crea el asociado input y output.
- Anuncio de Rendezvous. Describe a un peer que actúa de rendezvous para un determinado grupo.
- Anuncio de Información de Peer. Información varia del peer como recurso, por ejemplo: último mensaje recibido, hora de último mensaje enviado, estado del peer, etc.
- Anuncio de Módulo de Clase. Describe un módulo de clase y determina su existencia.
- Anuncio de Módulo de Especificación. Define un módulo de especificación. Su propósito es facilitar referencias a la documentación necesaria para crear implementaciones de las especificaciones.
- Anuncio de Módulo de Implementación. Define una implementación dado un módulo de especificación. Permite a un peer obtener la información necesaria para ejecutar una implementación.

Ejemplo de anuncio de pipe:

```
<?xml version="1.0"?>

<!DOCTYPE jxta:PipeAdvertisement>

<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
        <Id>
           urn:jxta:uuid-
59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504
        </Id>
        <Type>
           JxtaUnicast
        </Type>
        <Name>
           TestPipe.end1
        </Name>
</jxta:PipeAdvertisement>
```

**Fig. 3.7** Anuncio de pipe.

Los anuncios se componen de una serie de elementos jerarquizados. Cada elemento contiene su información o elementos adicionales. Un elemento puede tener atributos. Los atributos son las parejas nombre-valor. Un atributo se usa para almacenar meta información, la que ayuda a describir la información que contiene el elemento.

## 3.4. Arquitectura de red

Las redes P2P se caracterizan por ser muy heterogéneas. Las conexiones pueden ser muy transitorias y el tráfico de mensajes intercambiados entre peers no es determinista. Los peers pueden abandonar o unirse a la red en cualquier momento y las rutas pueden cambiar frecuentemente. La organización de la red no está definida por JXTA, pero en la práctica se pueden diferenciar 4 tipos de peers básicamente usados.

- Minimal edge peer. Puede enviar y recibir mensajes pero no puede cachear o encaminar anuncios para otro peer. Suelen ser peers ubicados en dispositivos de capacidades limitadas como una PDA o un teléfono móvil.

- Full-featured edge peer. Puede enviar y recibir mensajes y cachear anuncios. Responde a las peticiones de descubrimiento con la información de anuncios que tiene cacheada pero no reenvía ninguna petición de descubrimiento.

- Rendezvous peer. Es como cualquier otro peer pero mantiene una caché de anuncios. Puede reenviar peticiones de descubrimiento para ayudar a otros peers a descubrir recursos. Cuando un peer se une a un grupo automáticamente busca un peer rendezvous, y si no lo encuentra él se convierte en el rendezvous para el grupo. Cada rendezvous mantiene una lista de los demás rendezvous que conoce así como de

los peers que lo usan como rendezvous. En un grupo pueden haber tantos como sea necesario.

Los peers edge envían peticiones de búsqueda a los rendezvous, los que las reenvían a otros rendezvous en caso de no poderlas responder. El proceso continúa hasta que la petición es respondida o muere (se le agota el tiempo de vida).

- Relay peer. Mantiene información de rutas hacia los peers y encamina mensajes hacia los peers. Un peer primero busca en su caché local información de ruta para un peer y si no la encuentra le envía peticiones a un relay solicitándola. Éstos también reenvían mensajes de peers que no pueden enviar a otro peers pues están en diferentes redes físicas o lógicas, con la intermediación de NAT's o cortafuegos. Usa HTTP para transportar los mensajes.

### 3.4.1. NAT y cortafuegos

Un peer detrás de un cortafuegos puede enviar un mensaje directamente a un peer fuera del cortafuegos pero un peer fuera de un cortafuegos no puede establecer conexión directa con un peer de dentro del cortafuegos.

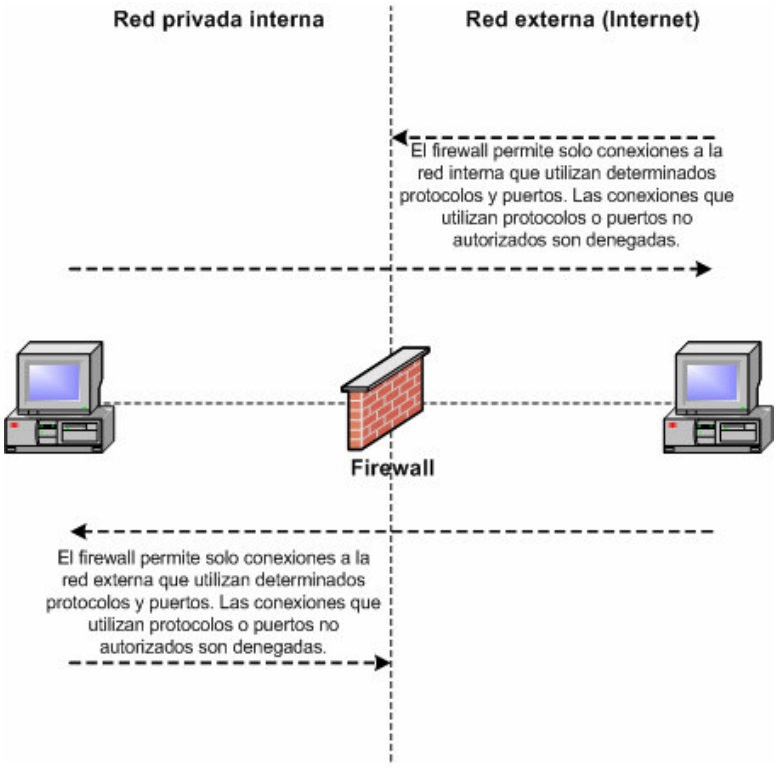



**Fig. 3.8** Escenario con cortafuegos.

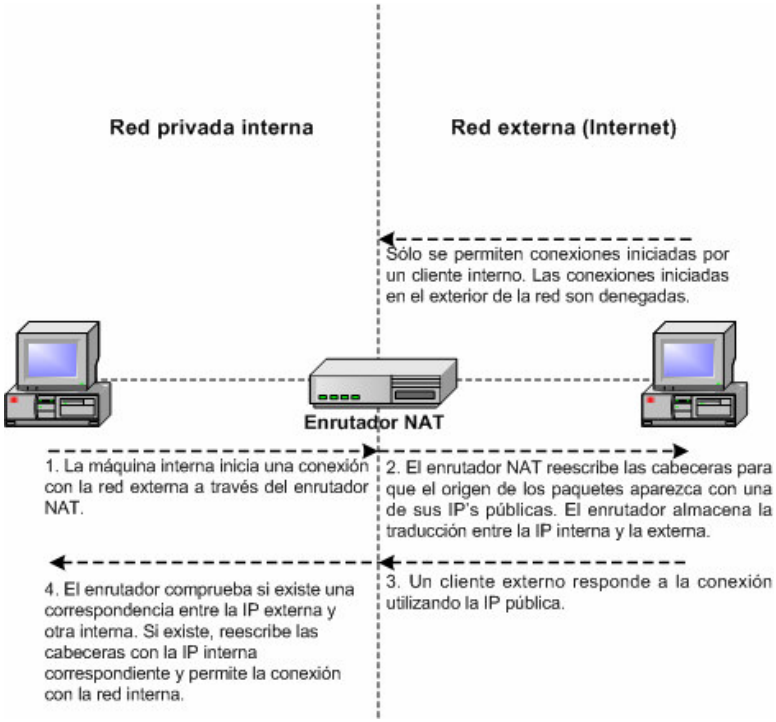



**Fig. 3.9** Escenario con NAT.

Para permitir la comunicación entre ellos a través de un cortafuegos deben darse las siguientes condiciones:

- Un peer dentro del cortafuegos debe conocer una peer situado fuera del cortafuegos.
- El peer dentro del cortafuegos y el de fuera deben soportar HTTP
- El cortafuegos debe permitir transferencias HTTP.

A continuación se expone un escenario en el que un peer A y un peer B se quieren intercambiar mensajes, pero existe un cortafuegos en medio. El peer A primero se conecta al peer C usando HTTP como protocolo que puede atravesar el cortafuegos. Luego, el peer C se conecta al peer B usando un protocolo como TCP/IP, dando lugar a una conexión virtual entre A y B.

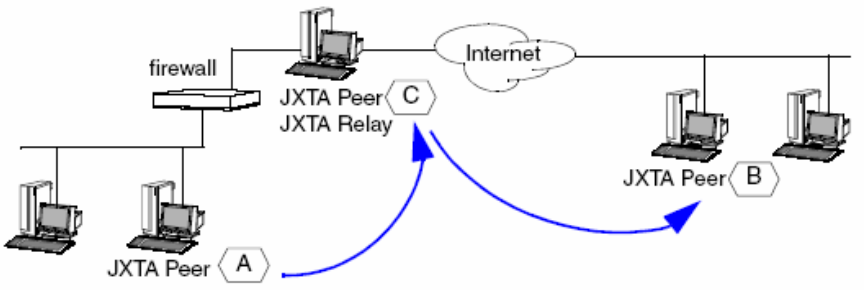



**Fig. 3.10** Encaminamiento de mensajes a través de un cortafuegos.

## 3.5. Protocolos JXTA

JXTA define una serie de formatos para mensajes XML o protocolos para establecer la comunicación entre peers. Los peers los usan para descubrirse, anunciar o descubrir recursos de red, y encaminar mensajes.

Existen 6 protocolos[16] JXTA:

- Protocolo de Descubrimiento de Peer (PDP). Lo usan los peers para anunciarse o descubrir otros servicios / recursos de otros peers. Cada recurso de peer se describe y publica mediante un anuncio. Un recurso puede ser cualquier cosa que se anuncie mediante un anuncio (peer, pipe, servicio, grupo, etc.).
- Protocolo de Información de Peer (PIP). Usado por peers para obtener información de estado de otros peers una vez éstos han sido localizados.
- Protocolo de Resolución de Peer (PRP). Permite a los peers enviar un mensaje genérico a uno o más peers y recibir una o múltiples respuestas. Los mensajes se pueden enviar a todos los peers en un grupo o bien a ciertos peers específicos en el grupo. A diferencia de PDP y PIP, que se usan para solicitar información específica, este protocolo permite a los peers intercambiar cualquier información.
- Protocolo de Binding de Pipe (PBP). Permite a los peers establecer un canal virtual o pipe entre uno o más peers. PBP conecta los terminales a las pipes de los peers.
- Protocolo de Encaminamiento de Terminal (ERP). Lo usan los peers para encontrar caminos a los puertos de otro peer. La información de ruta incluye una secuencia ordenada de peers relay que pueden ser usados para enviar un mensaje. Por ejemplo, un mensaje puede ser entregado si se le pasa al peer A, el cual se lo pasa (función relay) al B y éste al destino final.
- Protocolo de Rendezvous (RVP). Utilizado para propagar mensajes dentro de un grupo. En un grupo, los peers pueden ser peers rendezvous o peers que escuchan a peers rendezvous. RVP permite enviar un mensaje a todas las instancias del servicio. RVP es usado por el PRP y PBP para propagar mensajes.

El protocolo PDP estandariza el formato de mensaje que permite a un peer o elemento del sistema encontrar otros servicios, peers, etc. Este protocolo a su vez emplea el PRP el cual le permite enviar y recibir peticiones de búsqueda. A su vez, este protocolo emplea el RVP que se comunica con los peers Rendezvous que guardan información sobre los peer, servicios, etc. que conoce, de tal manera que puede proporcionar la información a otro peer que se comunique con él.

Existen varios tipos de mecanismos de descubrimiento:

- LAN-based: el descubrimiento se realiza mediante mensajes broadcast.

[16] Especificación de protocolos JXTA : http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html

- Invitation: un peer proporciona información del sistema sin solicitud previa de éste.
- Cascade: los peers se van proporcionando información sobre los otros peers que conoce, con autorización de éstos.
- Multicast: el descubrimiento se realiza mediante un mensaje multicast. Una vez conoce a peers de un grupo, sólo se comunica con ellos, no necesariamente con todos los nodos de la red.

Los protocolos JXTA no requieren que la red sobre la que funcionan tenga capacidades broadcast o multicast. La capa de aplicación se encarga.

Todos los protocolos JXTA son asíncronos y están basados en un modelo petición-respuesta. No es obligatorio que un peer implemente todos los protocolos, tan sólo aquellos necesarios. Sin embargo el PDP es de obligada implementación.

La primera cosa que un peer JXTA debe hacer es descubrir su entorno una vez ya está unido al Net Peer Group. Para ello, un peer que desea descubrir un recurso propaga una petición de descubrimiento dentro del ámbito de un grupo. Las respuestas se reciben de manera asíncrona. Debido a la heterogeneidad de la red, se pueden recibir todas, algunas, ninguna o redundantes respuestas, no existe ninguna garantía de recibir toda la información. Un peer puede estar en cualquier momento caído. Por eso, es responsabilidad de la aplicación P2P implementada el decidir por cuánto tiempo esperará respuestas a sus peticiones.

Si el peer desea descubrir un anuncio en particular, se debe explicitar el tipo de anuncio que se desea encontrar, el atributo y el valor exacto de lo que se desea encontrar y el número de anuncios máximos que se desea recibir (esto es especialmente útil para dispositivos con poca memoria).

Los anuncios descubiertos asíncronamente son almacenados automáticamente en una caché local del peer gracias a un thread que se ejecuta en modo background.

Todas las operaciones de descubrimiento son gestionadas por el servicio de descubrimiento. El descubrimiento se puede hacer de dos maneras, lo normal es implementar una detrás de la otra. La primera se basa en mirar primero los anuncios recientemente descubiertos y que son almacenados en la caché local (descubrimiento pasivo)

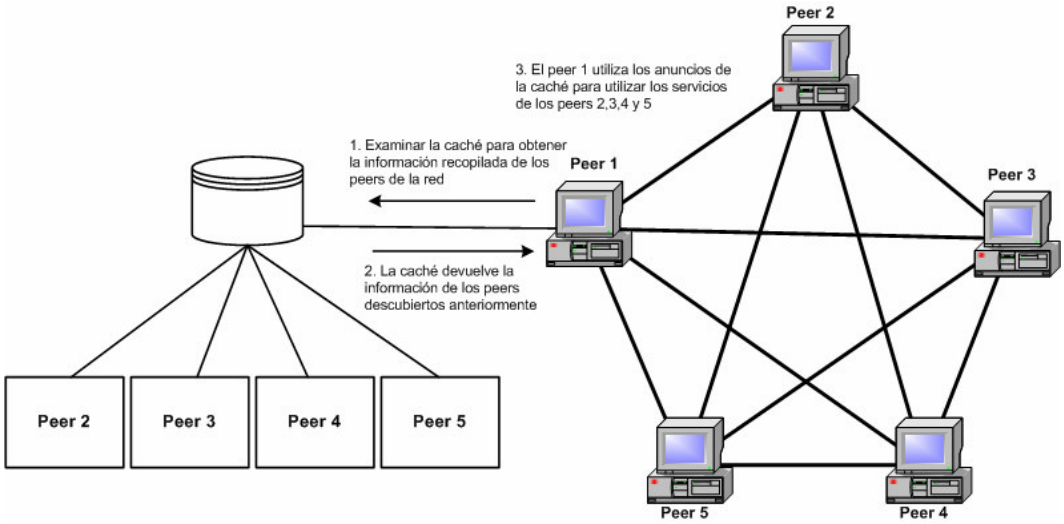



**Fig. 3.11** Descubrimiento Pasivo.

En segundo lugar, si no se encontrase en la caché, se procede al envío de una petición broadcast o multicast a los miembros del grupo al que pertenece el peer, en la misma LAN (descubrimiento directo).

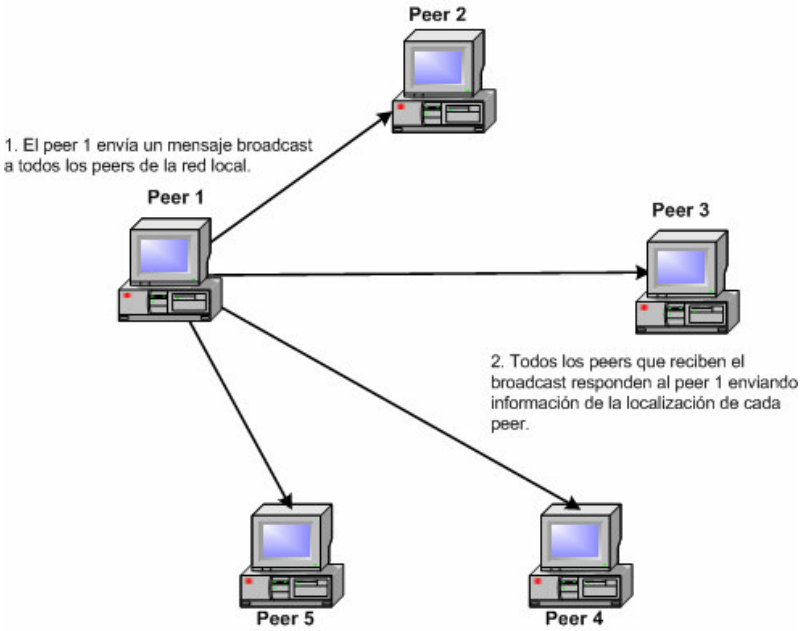



**Fig. 3.12** Descubrimiento Directo.

Si tampoco tuviera éxito el descubrimiento se procedería a pasarle la petición a un peer rendezvous (descubrimiento indirecto).

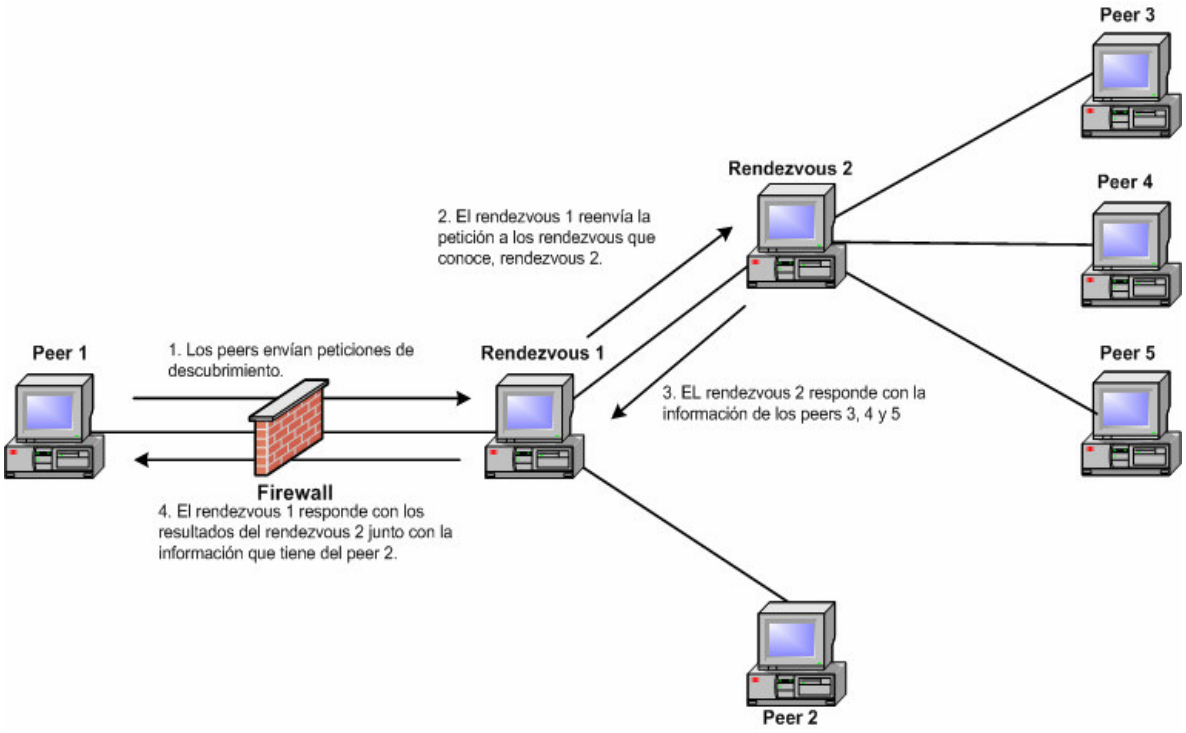



**Fig. 3.13** Descubrimiento Indirecto.

Como ejemplo pongamos que se desea implementar un servicio de restaurante. Para ello lo primero que se hace es buscar si existen anuncios correspondientes a un grupo llamado "restaurantes". Lo que se hace en primer lugar es mirar la caché local para ver si se encuentra el anuncio obtenido en una resolución anterior y en caso negativo se envía un petición de descubrimiento de anuncios de grupos según PDP en modo broadcast. En caso de negativa se contacta con un rendezvous conocido. Acto seguido, se procede a esperar las respuestas.

En caso de no encontrar respuesta pasado un tiempo, lo lógico es que el peer cree el grupo "restaurantes" pues se interpreta que no existe. Si se encontrase una respuesta, simplemente se uniría al grupo.

## 3.6. Identificadores (ID's)

Los peers, grupos, pipes y otros recursos JXTA deben estar identificados de manera única.

JXTA usa URN para expresar sus ID's. Los URN[17] son un tipo de URI que pretenden ser identificadores de recursos persistentes e independientes de la localización.

Los ID's son generados aleatoriamente. Existen 2 ID's reservados: el NULL y el del Net Peer Group.

[17] Ver IETF RFC 2141 para mayor detalle.

## 3.7. Seguridad

Los peers JXTA se basan en un modelo de confianza, en el que cada peer actúa bajo la autoridad otorgada por otro peer confiable.

Hay 5 requisitos que se deben cumplir.

- Confidencialidad. Garantiza que el contenido del mensaje no será visto por usuarios desautorizados.
- Autenticación. Garantiza que el emisor del mensaje es quien dice ser.
- Autorización. Garantiza que el emisor puede enviar el mensaje.
- Integridad de la información. Garantiza que el mensaje no ha sido alterado accidental o deliberadamente durante su transferencia.
- No repudio. Garantiza que el mensaje fue transmitido por un emisor identificable, y no es una réplica de un mensaje previamente transmitido.

Los mensajes XML pueden incorporar meta información referente a credenciales, certificados, claves públicas, etc. Los mensajes también pueden ser encriptados (mediante clave pública) y firmados (mediante certificados) para obtener confidencialidad y no repudio. Las credenciales pueden ser usadas para autorizar y autenticar.

JXTA pretende ser compatible con la mayoría de protocolos de transporte seguros como Secure Socket Layer (SSL) o Internet Protocol Security (IPSec). Sin embargo SSL e IPSec tan sólo ofrecen integridad y confidencialidad entre 2 peers comunicados. Para aportar seguridad en una red multisalto, una asociación de confianza debe ser establecida entre todos los peers intermediarios. La seguridad se compromete si alguien en la comunidad no está asegurado.

## 3.8. Ventajas y retos

Las ventajas de las redes P2P son:

- Escalabilidad, robustez, disponibilidad, prestaciones y recursos, libertad de usuarios y ausencia de mediadores

Los retos inmediatos son:

- La seguridad, confianza entre peers y su reputación, madurez de la tecnología, estandarización, definición de modelos de negocio, complejidad en su desarrollo y la administración y control de la red.

# ANEXO 4. Apache Xindice

Para la realización del proyecto se planteó estudiar bases de datos relacionales, en concreto MySQL, y bases de datos XML, en concreto Xindice de Apache. La elección de éstas últimas viene motivada por el hecho de que los recursos serán descritos mediante MPEG-21, basado en XML.

Xindice es una base de datos pensada para almacenar información en formato XML. Se basa en un servidor Xindice que es el encargado de gestionar todo lo que sería la base de datos. Todo lo que recibe o envía el servidor es XML.

Para hacer las peticiones que se hacen a la base de datos se basan en Xpath[18], y soporta DOM y SAX.

Tan sólo debemos preocuparnos de conformar la información como XML, que así será guardada.

Actualmente, las bases de datos XML representan un nuevo campo de desarrollo, y se puede considerar un proyecto pues aún está en constante desarrollo.

## 4.1. Características

Los documentos se engloban en colecciones. A la base de datos no le importa si se crea una única colección para almacenar todos los documentos XML que se tengan o si es para almacenar tan sólo documentos de un tipo determinado.

Las peticiones se basan en Xpath, definido por W3C.

Indexación XML. Se pueden indexar elementos y atributos de los documentos XML almacenados para acceder más rápidamente a la información deseada.

Se puede consultar información, añadir, borrar o modificar.

Xupdate permite actualizar información de un documento XML almacenado sin tener que extraer el documento entero. Permite hacer actualizaciones a documentos o a colecciones enteras.

Existe una API para entornos de programación Java.

Gestión por línea de comandos. Casi todo lo que se puede hacer mediante la API de programación de Xindice, también se puede hacer a través de la línea de comandos.

[18] Expresiones para evaluar, extraer y encontrar información en un documento XML. La especificación se encuentra en http://www.w3.org/TR/xpath.

Arquitectura modular. El servidor Xindice está construido de manera muy modular lo que facilita que se le puedan añadir o quitar componentes para adaptar el servidor a diferentes entornos o para empotrarlo en otras aplicaciones.

## 4.2. Estructura de la base de datos

Se almacenan colecciones de documentos XML. Éstas pueden ser ubicadas de manera jerárquica como si se tratara del sistema de ficheros típico de Unix o Windows.

En Xindice la información almacenada se encuentra en una colección principal, root. Esta instancia principal puede tener cualquier número de colecciones hijas. Por defecto el elemento raíz que aparece cuando se instala Xindice se denomina db. Por lo tanto la ruta seria /db.

Se podrían añadir tantas colecciones hijas como se desease, como por ejemplo:

/db/hijo_1/hijo_2/..../hijo_n

# ANEXO 5. MPEG 21

## 5.1. Introducción

El grupo MPEG (Moving Picture Expert Group) ha realizado varias estandarizaciones de formatos de compresión de imagen en movimiento, audio y vídeo. Los conjuntos de estándares MPEG 1, MPEG 2 y MPEG 4 tratan el tema de la compresión de vídeo a diferentes calidades cada uno. Posteriormente, el grupo comenzó a desarrollar MPEG 7 y MPEG 21. Sin embargo, esta vez no están orientados a la compresión de contenido multimedia. MPEG 7 se encarga de etiquetar contenido multimedia mediante meta datos de manera detallada. Con estos meta datos se puede realizar la búsqueda en bases de datos de registros correctamente etiquetados según MPEG 7.

Por otro lado, MPEG 21 se encarga de definir una red de intercambio de contenido multimedia. Su objetivo principal es definir los participantes que intervienen en una transacción digital, en el que los bienes a intercambiar son datos. MPEG 21 se sustenta en la definición de un Objeto Digital (DI), el cuál será el bien a intercambiar. También se especifican derechos de propiedad intelectual y de utilización que tiene cada usuario sobre los objetos digitales disponibles en la red.

El Objeto digital (Digital Item) es un documento XML estructurado que representa la meta información de un recurso multimedia (música, fotos, video,

canciones, etc.) de manera estándar. Esta definición sirve para describir los recursos multimedia que se vayan a almacenar. Para describirlos se basa en la agrupación de unos sub-items o componentes que están ligados a descriptores según se define en la Definición de Objeto Digital (DID).

El propósito de MPEG 21 es definir un entorno abierto para la compartición de contenido multimedia. Su proceso de estandarización sigue en curso.

## 5.2. Partes de la especificación MPEG 21

MPEG 21 se estructura en las siguientes partes principalmente:

**Parte 1. Visión, tecnología y estrategia.**

Describe el marco funcional junto a los elementos que lo integran y los requisitos funcionales.

**Parte 2. Declaración de Objeto Digital (DID).**

Esta especificación pretende describir un conjunto de términos y conceptos para formar un modelo útil de definición de objetos digitales.

La DID se describe en 3 secciones:

- Modelo: describe una serie de conceptos abstractos para conformar el modelo a partir del cual se define el objeto digital.
- Representación: descripción sintáctica y semántica de cada elemento del DI.
- Schema: Schema XML normativo del DID.

**Parte 4. Gestión de propiedad intelectual y protección de componentes.**

Describe el marco de trabajo y las herramientas para la gestión de derechos.

**Parte 5. Lenguaje de expresión de derechos (REL).**
Lenguaje usado para declarar derechos y permisos usando los términos definidos por el Diccionario de Derechos de la Información.

REL ofrece mecanismos para un amplio y transparente uso de los contenidos digitales en su publicación, distribución y consumo.

**Parte 6: Diccionario de Derechos de la Información.**

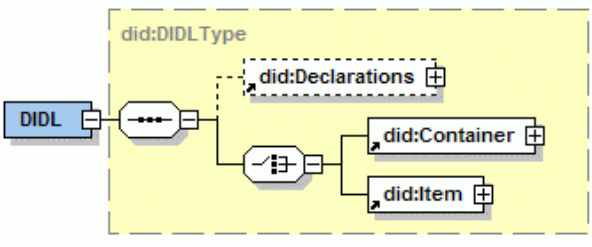
Comprende un conjunto claro, consistente, estructurado e integrado de términos para el uso de REL.

**Parte 7: Adaptación de Objeto Digital**

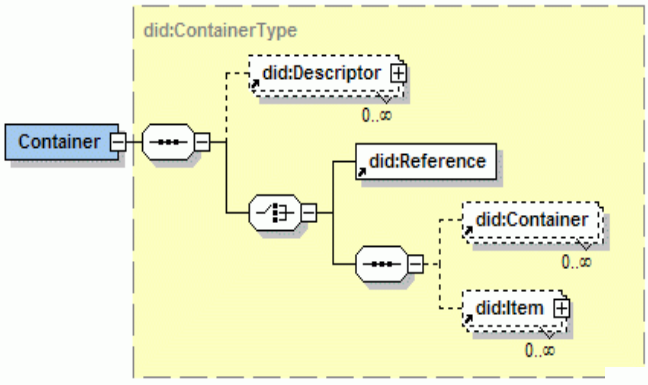
Describe una serie de herramientas que permiten construir un entorno UMA (Universal Multimedia Acces).

## 5.3. Declaración de Objeto Digital (DID)

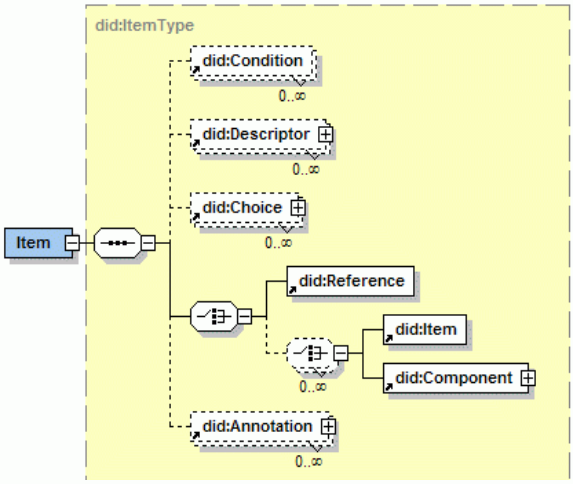
**DIDL**: raíz del documento DID.




**Container**: estructura que permite agrupar otros *containers* o *items*.

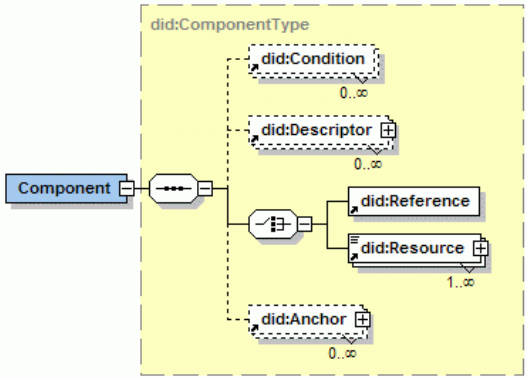



**Item**: un *item* agrupa sub*items* y/o *components* ligados a *descriptors*. *Descriptors* contienen información sobre *item*. *Items* pueden contener *choices* que permiten configurar/personalizar el *item*. También se puede realizar *annotations*.



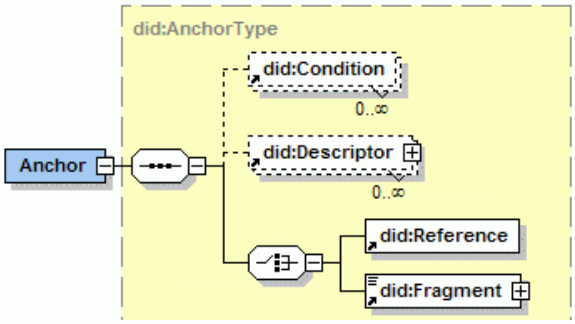

**Component**:

Es el conjunto de un *resource* con todos sus *descriptors*.



**Anchor**:

Conjunto de un *fragment* y sus *descriptors*.

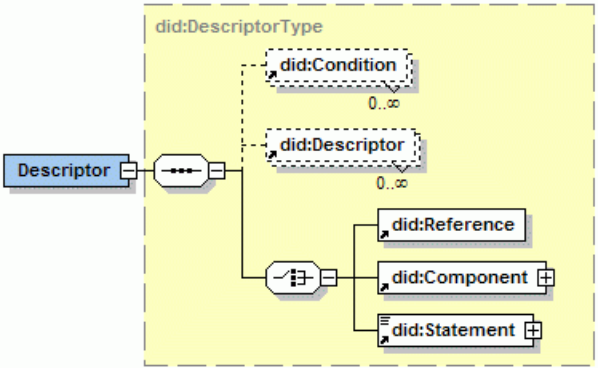



**Fragment**:

Puntero a una parte de un *resource.*

**Descriptor**:

Proporciona información asociada a al elemento "padre". Puede ser una descripción textual (*statement*) o un *component* (ex: thumbnail).

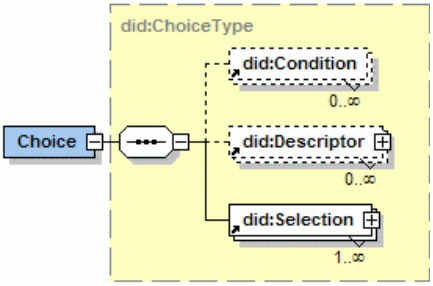



**Condition**:

El elemento que cierra es opcional, se asocia a unas determinadas *selections*.

**Choice**:

Describe un conjunto de *selections* que pueden afectar a la configuración del *item*.

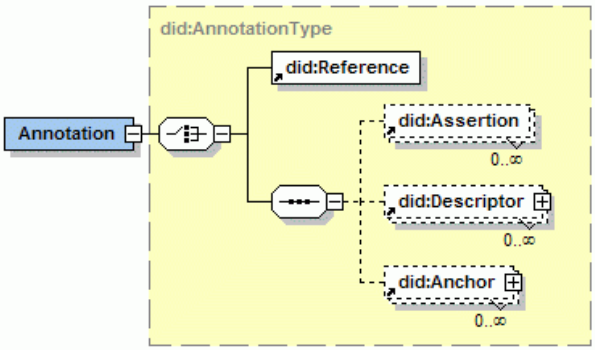



**Selection**:

Describe una decisión específica que puede afectar a una o más *conditions* dentro de un *item*.

**Annotation**:

Añade descripciones adicionales a un *item*.




**Assertion**:

Permite resolver una *choice*, fijando sus *selections* como ciertas o falsas.

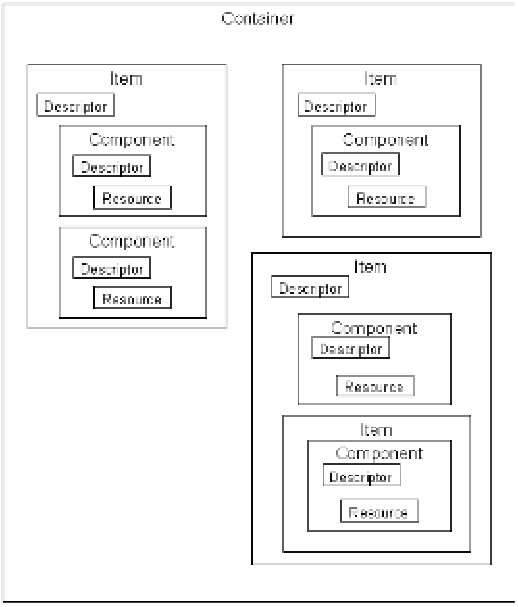
**Resource**:

Es el contenido multimedia (ex: clip de audio, video, una imagen...).

**Statement:**

Información en forma de texto.

La figura siguiente muestra las relaciones que se establecen entre los principales elementos de la DID.




# ANEXO 6. Instalación de JXTA

Para poder trabajar con JXTA se necesitan una serie de librerías. Estas librerías se pueden obtener de www.jxta.org las aplicaciones de prueba MyJXTA o JXTA Shell, ambas soportadas por Microsoft Windows, Linux, Apple Macintosh y Solaris.

Cuando se desarrolla un programa basado en el API de Java y se quiere arrancar la aplicación creada, aparece una serie de ventanas para configurar el entorno JXTA.

El configurador JXTA muestra 4 paneles.

**Básico**. Se debe introducir un nombre de peer. Este nombre será usado en los anuncios que se difundirán por la red (no tiene porqué ser único).

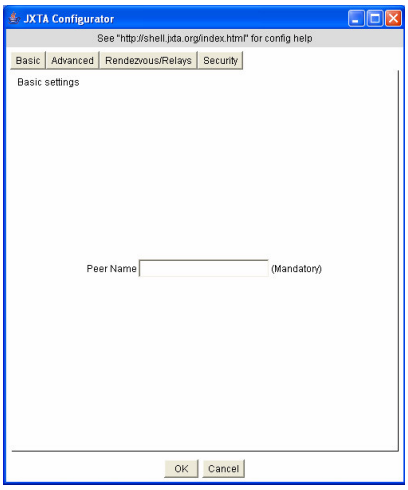



**Fig. 6.1** Opciones Básico.

**Avanzado**. Permite configurar información de la red donde se encuentra el peer. Soporta TCP/IP y HTTP. Por defecto viene configurado TCP/IP y HTTP.

Se debe activar las propiedades TCP si hay otros peers en la red local que se desean encontrar mediante la versión broadcast del PDP. Aparece también una lista de todas las redes que el mecanismo conoce (sólo se puede seleccionar una donde escuchar mensajes).

Si están corriendo múltiples instancias de la plataforma JXTA en un ordenador se debe cambiar el puerto (por defecto 9701). Cada instancia debe tener su puerto.

Se activan las propiedades HTTP si se quiere conectar a los rendezvous.

También se activan si hay un cortafuegos o NAT de por medio.

Si se está detrás de un cortafuegos se debe activar la opción de proxy.

Si no se está conectado a Internet, las opciones HTTP deben estar desactivadas.

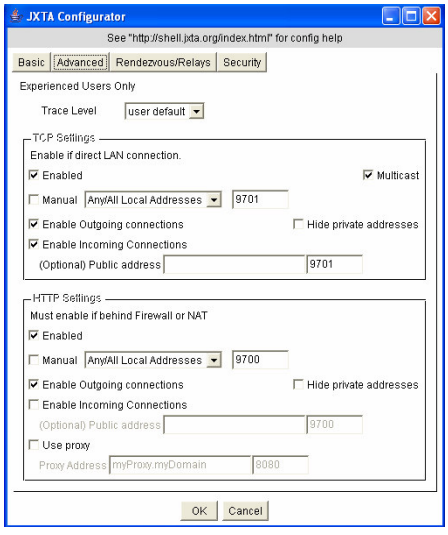


**Fig. 6.2** Opciones Avanzadas.

**Rendezvous /realays.** Permite especificar que un peer será usado como rendezvous o relay. Por defecto se puede descargar una lista de peers relay/rendezvous.

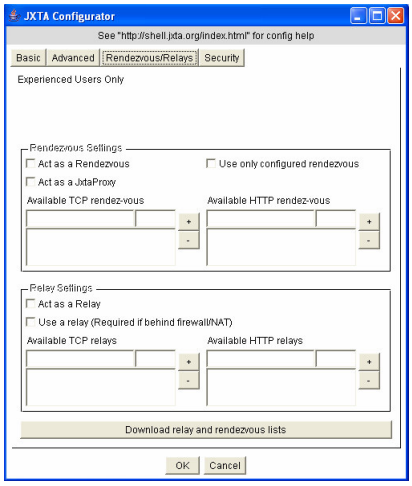



**Fig. 6.3** Opciones Relay / Rendezvous.

**Seguridad**. Solicita un nombre y password para poder acceder a una determinada configuración de peer.

Los anuncios descubiertos (caché local) y propiedades de la configuración son almacenados en una carpeta denominada .jxta.

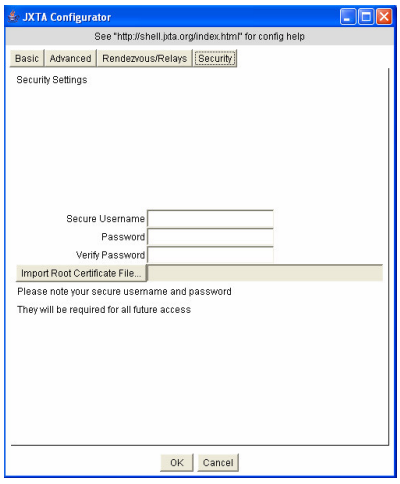



**Fig. 6.4** Opciones de Seguridad.

# ANEXO 7. Funcionalidades

A continuación se detallan las funcionalidades especificadas en el Proyecto Integrado de I2Cat. Estas funcionalidades se basan en:

- Servicios: el servicio principal tratado es la localización de recursos multimedia, sin embargo el sistema de localización, dentro de lo que es el Proyecto Integrado también se encargaría de localizar cualquier otro servicio disponible en la red. Otro servicio sería el de localización de usuarios o grupos de usuarios, videoconferencias, mensajería entre usuarios, etc.
- Recursos de media: el objeto de la búsqueda, es el elemento devuelto por el servicio de localización de recursos de media.

En la arquitectura distribuida cabe destacar que se potencia el concepto de usuario, facilitando por ejemplo la agrupación de los mismos en grupos definidos según ciertas afinidades, propósitos o intereses. Por lo tanto, el sistema de localización también puede ser el encargado de encontrar tanto a usuarios individuales como grupos de usuarios. Esto permite que cualquiera pueda crear grupos o añadirse a grupos y por lo tanto éstos deben ser publicados en la red y localizados, siempre en el caso de disponer de los derechos correspondientes según un perfil de usuario.

También se especifica las operaciones de gestión que se pueden efectuar sobre los servicios, recursos, grupos y usuarios.

## 7.1. Localización de recursos de media

- Obtener listado de recursos multimedia. Se puede especificar unas condiciones de búsqueda en función del servicio:
  - Audio bajo demanda:
    - Grupo
    - Año
    - Título
    - Género (por ejemplo: Pop, Rock, Heavy, Jazz, Clásica, Electrónica,).
    - Formato (por ejemplo: Mp3, Ogg, Wma, Wav)
    - Calidad (frecuencia muestreo)
    - Tamaño
  - Video bajo demanda:
    - Director
    - Actor
    - Título
    - Género (por ejemplo: Comedia, Drama, Terror, Suspense, Acción)
    - Formato (por ejemplo: Mpeg2/7/21, Avi, Xvid, Divx)

- Calidad (Resolución)
- Idioma
  - Versión Original.
  - Doblada (Idiomas, por ejemplo: inglés, español)
  - Subtitulada (Idiomas, por ejemplo: inglés, español)
- Años
- Galardones

▪ Videoconferencia:
- Tema.
- Fecha
- Participantes y/o organizador.

▪ Radio y Televisión.
- Canales
- Programación.
- Temática (por ejemplo: documentales, series, reality-shows)

***Nota:*** *Los parámetros de búsqueda de 'género', 'formato' e 'idiomas' mostraran los posibles valores registrados en el sistema de manera automática.*

- Información de disponibilidad del recurso multimedia. El usuario puede consultar si un recurso está disponible en un determinado instante.

- Valorar un recurso utilizado. Estos datos servirán como información estadística para el sistema.

▪ Valorar del 1 al 10 dependiendo del recurso solicitado.
- Imagen (Video bajo demanda/Videoconferencia/Televisión)
- Sonido(Video bajo demanda/audio/ Videoconferencia/Televisión/Radio)
- Argumento(Video bajo demanda)
- Interpretación(Video bajo demanda)
- Efectos especiales(Video bajo demanda)
- Valoración personal (Video bajo demanda/audio/ Videoconferencia/Televisión/Radio)

## 7.2. Localización de procesadores de media

Permite al usuario obtener el listado de Procesadores de Media que ofrecen el recurso requerido y si el Procesador de Media está disponible.

## 7.3. Ejecución de recursos de media

- Audio bajo demanda:

▪ Play

- Stop
- Pause
- Fast Forward
- Rewind

- Video bajo demanda:

  - Play
  - Stop
  - Pause
  - Fast Forward.
  - Rewind
  - Selección de escena.

- Videoconferencia:

  - Participar
  - Abandonar

- Radio/Televisión:

  - Ver Programación
  - Selección del canal
  - Visualización de varios canales simultáneamente.
  - Finalizar

## 7.4. Gestión

- Recursos. El usuario, según los derechos de los que disponga al acceder al sistema, podrá:
  - Publicar y añadir recursos (mediante registro UDDI usando Servicios Web o anuncios en redes P2P JXTA)
  - Eliminar recursos
  - Modificar la información relativa a un recurso.

- Servicios.

  - Publicar servicios (mediante registro UDDI usando Servicios Web o anuncios en redes P2P JXTA)
  - Eliminar servicios.
  - Crear una videoconferencia.

- Usuarios.

  - Dar de alta a un usuario (módulo de autenticación).
  - Dar de baja a un usuario (módulo de autenticación).
  - Denegar acceso, expulsar a un usuario.
  - Listar usuarios.
  - Consultar perfil de usuario.
  - Actualizar perfil de usuario.

- Grupos.

  - Crear un grupo.
  - Eliminar un grupo.
  - Crear Encuesta.
  - Eliminar Encuesta

- Perfiles de usuario. Permite seleccionar un modo de búsqueda automático o manual.

  - Automático: el sistema proporciona al usuario el recurso que más se adapte a sus características.
  - Manual: El usuario selecciona el recurso entre la lista de recursos disponibles facilitada por el servicio de localización de recursos de media.

## 7.5. Funcionalidades de grupos

- Agregar usuarios.
- Listar usuarios.
- Listar grupos.
- Abandonar un grupo
- Intercambiar mensajes entre usuarios pertenecientes a un mismo grupo.
- Visualizar el estado de los usuarios del grupo al que se pertenece.
- Intercambiar archivos entre los usuarios miembros de un grupo.
- Ver Encuesta.
- Votar Encuesta.
- Participar en videoconferencia.
- Login (Identificación hacia el grupo).

## 7.6. Información del sistema

- Obtener “lista negra” de servidores (por ejemplo aquellos que envían archivos infectados con virus, archivos multimedia engañosos, etc.). Estas listas se pueden obtener automáticamente a través de la información del usuario. El usuario puede introducir su valoración sobre el servicio o recurso recibido.
- Ver historial de los servicios visitados durante el último mes, última semana, últimos 2 días o ayer.
- Borrar historial citado en el punto anterior.
- Estadísticas de los servicios más utilizados.
- Estadísticas de cuantos usuarios acceden a la vez un determinado servicio en un momento preciso.
- Estadísticas de utilización de un servidor como son: hora punta del día, días de mayor actividad durante la semana, periodos del año más utilizado.
- Valoración del recurso.

## 7.7. Perfiles de usuario

El sistema define una serie de perfiles de usuario los cuales determinan las funcionalidades que un usuario puede llevar a cabo.

- Usuario
  - Localizar recursos multimedia
  - Ver el estado de los servidores.
  - Ver el estado de los recursos.
  - Publicar Servicios.
  - Eliminar Servicios.
  - Listar Usuarios.
  - Localizar Servidores
  - Ejecutar recursos multimedia.
  - Ver listado de grupos
  - Solicitar la incorporación a grupos
  - Darse de baja del sistema.
  - Crear Grupos.
  - Visualizar todas las estadísticas.
  - Consultar valoración de servicios / recursos obtenido por parte de los usuarios registrados y/o administrador, los cuales son los que han valorado.

- Usuario Registrado en un Grupo: Este tipo de usuario puede realizar todas las acciones de Usuario y además las citadas a continuación.
  - Listar miembros de un grupo, así como su estado (Disponible, no disponible)
  - Intercambio de mensajes con los miembros del grupo.
  - Ver el historial de los recursos utilizados.
  - Poder borrar el historial propio.
  - Participar en una videoconferencia (cada usuario registrado dispone de un identificador propio como por ejemplo un SIP URI).
  - Intercambio de archivos.
  - Proponer encuesta.
  - Votar encuesta.
  - Agregación a una videoconferencia.

- Administrador de un Grupo / Servicio: Este tipo de usuario puede realizar todas las acciones de Usuario y además las citadas a continuación.
  - Acceso a los historiales relacionados con el grupo de todos los usuarios registrados.
  - Eliminación de participantes de un grupo.
  - Eliminación del grupo.
  - Eliminar encuesta.
  - Crear videoconferencia.
  - Eliminar videoconferencia programada
  - Modificar programación videoconferencia.