Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

# TRABAJO FINAL DE CARRERA

**TÍTULO DEL TFC: Arquitectura avanzada de adaptación de recursos multimedia**

**TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática**

**AUTOR: Alberto Carlos Toro Sánchez**

**DIRECTOR: Antoni Oller Arcas**

**FECHA: 24 de Julio de 2008**

**Título: Arquitectura avanzada de adaptación de recursos multimedia**

**Autor: Alberto Carlos Toro Sánchez**

**Director: Antoni Oller Arcas**

**Fecha: 24 de Julio de 2008**

## Resumen

Con el aumento de ancho de banda en la red y de las prestaciones de las máquinas se pueden crear nuevos y mejores servicios con los que lograr mejor experiencia del usuario, facilitando la vida de éste.

El objetivo del proyecto es desarrollar un elemento que recopile información sobre el usuario, la gestione, realice una clasificación basada en metadatos. Esta información se utilizaría posteriormente para ofrecer servicios a medida.

Inicialmente, se han hecho estudios previos de varias tecnologías que pueden ser de utilidad para llevar acabo el diseño y la implementación del elemento para ofrecer servicios a los usuarios. Se han realizado diferentes pruebas para comprobar sus funcionalidades de cara al trabajo final de carrera.

Teniendo claras las distintas tecnologías, se ha diseñado e implementado una arquitectura, basada en estándares para la gestión de *media* y perfiles: MPEG-7 y MPEG-21, para el sistema de manera iterativa, es decir, a partir de los desarrollos de prueba hechos para comprobar el funcionamiento de cada tecnología, se han ido añadiendo nuevas funcionalidades y comunicando varios módulos entre sí.

Finalmente se ha conseguido implementar una arquitectura donde el usuario puede interactuar con una aplicación *web* y desde la cual se le permite realizar registros, entradas al sistema, comprobar sus datos y preferencias almacenadas; y recibir recomendaciones desde el sistema de contenidos que le pueden llegar a interesar para posteriormente visionarlos.

**Title: Advanced architecture of adaptation for multimedia resources**

**Author: Alberto Carlos Toro Sánchez**

**Director: Antoni Oller Arcas**

**Date: 24th July 2008**

## Overview

Taking into account the increase in bandwidth and capacity of current networks and computers, new and better services can be created in order to achieve the best user experience.

The main objective of the project is to develop a software element that gathers user information, manages and stores it into metadata to offer personalized services.

First of all, previous studies has been made pointed to different techonologies that can be useful for engaging the design and implemetation of the element for offering services to users has been studied. Moreover, different tests have been made to check that the functionalities of these technologies met the requirements of this project.

Once the technologies were well known, an architecture, based on standards for media management and description: MPEG-21 and MPEG-7, has been designed and developed in an iterative manner. That is, starting from different proofs of concept, new operations have been added and the main components have been implemented individually. After that, all the components have been integrated to provide the full service.

Finally, it has been achieved the development of an architecture where the user can interact through a web interface which allows to register, login into the system, check user's stored personal data and preferences. The final application enables to receive recommendations from the system, which makes the system easier to use.

# AGRADECIMIENTOS

A lo largo de todo el tiempo dedicado a ese Trabajo Final de Carrera han sido muchas las personas que me han mostrado todo su apoyo.
En primer lugar dar las gracias a Toni y a Alberto por darme la oportunidad de realizar este trabajo, por todo lo que me han apoyado y orientado en momentos de desanimo. A Toni por confiar en mí ofreciéndome un puesto en las instalaciones de la Fundación i2CAT para realizar el TFC y a Alberto, por toda su ayuda prestada durante el desarrollo del trabajo.
Dar las gracias también a todos los compañeros de i2CAT que rápidamente me acogieron como uno más, y sobre todo por el tiempo que les he robado prestándome ayuda.
Como no, agradecer con un cariño especial a mis padres, hermanos, tíos, abuelos y el resto de familia que siempre han estado a mi lado, animándome y dándome lo mejor de ellos en toda mi época de estudiante.
De la misma forma, agradecer a Lara todos los ánimos, ayuda y cariño que siempre he tenido ahí, cuando más lo he necesitado, durante toda la carrera.
Gracias a todos mis amigos, ellos saben quien, por toda su amistad y apoyo desde bien pequeños y no tan pequeños.
Por último, no quiero olvidarme tampoco de todos los compañeros de clase que he tenido, a todos los que me han ayudado siempre que han podido.
A todos ellos, muchas gracias por hacer de estos años de estudiante algo que no se olvidará nunca.

¡Gracias a todos!

# ÍNDICE

# ÍNDICE DE FIGURAS

# INTRODUCCIÓN

Vivimos en una época donde Internet es parte de nuestra vida cotidiana y tanto el incremento de ancho de banda para conexiones como el aumento de prestaciones de las máquinas (cpu, memoria, capacidad discos duros, etc.) y de las redes de comunicación nos plantea nuevos retos y caminos para el desarrollo de nuevos servicios de cara al usuario.

En este contexto se abre un abanico de posibilidades de cara a investigar, desarrollar y ofrecer nuevos servicios a usuarios. Los servicios que más interés despiertan entre la industria son los multimedia. La creación de servicios interactivos más inteligentes que, aprovechen el ancho de banda creciente en Internet y la versatilidad de los nuevos dispositivos, pretende mejorar en gran medida la experiencia del usuario y a la vez simplificar la intervención del usuario.

Para poder llevar a cabo toda esta mejora de experiencia y automatización para el usuario es necesario de una aplicación que reúna toda la información posible sobre éste, la trate y la organice; para posteriormente poder retomarla cuando haga falta y adecuar los nuevos servicios a las necesidades y/o posibilidades de cualquier usuario. Hablamos un servidor de perfiles de usuario, un *Profiling Server* que colabora para que el sistema pueda aprender del usuario.

Un elemento que en todo momento podría saber qué ofrecer, qué cualidades de los servicios destacar, cuándo realizar cambios, etc.; todo esto y muchas más funcionalidades se pueden destacar de la idea de un servidor que contenga un servicio de perfiles. Pero para poder llegar a todo esto, antes se necesita aprender del usuario: sus necesidades, qué hace, cómo lo hace, qué consume, dónde lo consume, etc. Sabiendo todo esto el servidor debería ser capaz de almacenar todo esto en perfiles y poder ofrecer multitud de posibilidades, dependiendo del servicio que se quisiera ofrecer, para el usuario. Todas estas posibilidades estarían acotadas a todas las preferencias que se hubiesen podido aprender o deducir de un usuario.

Este proyecto trata de desarrollar dicha idea y de buscar soluciones viables para el desarrollo del servicio.

El trabajo se organizará de la siguiente manera. En el primer capítulo, nos ubicaremos en el entorno de trabajo, una breve visión de lo que es la Fundación i2CAT [15], el proyecto I3Media [16], así como el contexto del proyecto y sus objetivos a cumplir.

En el segundo capítulo se realizará una prueba de concepto, comentando las posibles funciones de este servidor y un seguimiento sobre las funciones que se desarrollarán para éste.

En el tercer capítulo se realiza una breve explicación sobre la arquitectura del sistema I3Media y localizar dentro de éste al *Profiling Server*. Se explicarán

cada uno de los componentes más relevantes de este servidor y se mostrará la arquitectura utilizada para este Trabajo Final de Carrera (TFC). Se continúa con una explicación más exhaustiva sobre los componentes del servidor, cada diseño e interacción de estos.

En el cuarto capítulo se presenta la implementación de todo este trabajo: pasos seguidos para la realización, herramientas y tecnologías utilizadas, entorno de programación y una breve explicación de lo que nos podremos encontrar en el servidor.

En el quinto capítulo, se realizará una explicación sobre la planificación y los costes de desarrollo de todo el trabajo.

Finalmente se comentarán las conclusiones extraídas a lo largo de todo el desarrollo e implementación, una serie de mejoras para el sistema y se describirá el impacto medioambiental del TFC.

# CAPÍTULO 1. DESCRIPCIÓN DEL PROYECTO

El estudio para este trabajo nace a raíz de un proyecto iniciado en la Fundación i2CAT. La Fundación i2CAT es una entidad sin ánimo de lucro, que tiene como objetivo impulsar el desarrollo y la innovación en las tecnologías de Internet.

El proyecto en el que se basa este TFC, denominado I3Media, está siendo desarrollado por las empresas más importantes del sector audiovisual, como por ejemplo, Media Pro, Telefónica, Alcatel-Lucent, TV3, junto con i2CAT, las cuales promueven un consorcio de investigación y desarrollo para la creación de contenidos audiovisuales inteligentes, personalizados y automatizados, cubriendo toda la cadena de valor del audiovisual.

I3Media, es uno de los 15 grandes proyectos del Centro para el Desarrollo Tecnológico Industrial (CENIT), supone un gran avance en nuevas fórmulas de producción y consumo de contenidos audiovisuales. El objetivo de I3Media es el análisis de nuevas tecnologías para la producción, gestión y explotación de contenidos. A través de estas nuevas tecnologías se tendrá la posibilidad de crear contenidos inteligentes, automatizados y adaptados a gusto del usuario. Toda la información estará estructurada en diferentes capas de información por las que el consumidor podrá navegar ofreciendo una total personalización, lo cual permitirá interactividad entre el contenido y el usuario.

El objetivo del proyecto es desarrollar un elemento capaz de mejorar la experiencia de usuario, automatizando de manera transparente la gestión de los contenidos audiovisuales teniendo en cuenta los gustos del usuario, conformado por varios módulos, como pueden ser, por ejemplo, el módulo de adaptación y el módulo *Profiling Server.* Este TFC se centrará en este último.

## 1.1. Servidor de perfiles

El sistema debe ser capaz de:

- Almacenar y gestionar:
  - Perfil de usuario.
  - Recursos multimedia disponibles.

- Ofrecer al usuario varios modos de funcionamiento:

  - Activo: El usuario solicita una recomendación y el sistema le propone contenidos

  - Pasivo: Es el sistema que ofrece directamente al usuario recomendaciones, por ejemplo, cuando éste último entre en el sistema.

Un ejemplo de la funcionalidad del sistema sería que, por ejemplo, un usuario, sin conocimiento técnico, al salir del trabajo, podría conocer las noticias más destacadas del día. Estas noticias habrían sido seleccionadas por el sistema, teniendo en cuenta todos los parámetros referentes al contexto del usuario (gustos, dispositivo, características del contenido, etc.). De esta manera, el sistema ofrece al usuario lo que le puede interesar.

Al llegar a casa, el usuario mediante el mando a distancia y pulsando un simple botón podría seleccionar de entre varias recomendaciones ofrecidas directamente por el sistema, según sus gustos y preferencias, una película adaptada a sus posibles necesidades y al tipo de televisor, aprovechando las cualidades de éste, y ofreciendo una experiencia confortable y sencilla.

## 1.2. Conceptos básicos

El contexto del usuario se organizará en dos estándares, uno para los recursos *media* y otro para las preferencias de usuario. Estos estándares son los siguientes:

- **MPEG-7 [1]:** Estándar basado en *Extensible Markup Language* (XML). Principalmente describe elementos de bajo nivel del audio o del video como, por ejemplo, color, texturas, volumen del audio, etc. También nos puede informar sobre elementos audiovisuales como tiempo, localización y calidad. Este estándar puede ser expresado de dos maneras: como documento textual e inteligible para poder ser editado, realizar búsquedas o filtrados. O como binario y solo comprensible por máquina, y de esta manera poder transmitirlo o almacenarlo si hiciese falta.

- **MPEG-21 [2]:** Es una recopilación de tecnologías o herramientas. Este estándar intenta solucionar muchos problemas existentes hoy en día con la distribución de los contenidos digitales, principalmente ilegales. Su propósito principal es el de establecer, de una manera clara, quiénes son los participantes de la transacción dentro de un mercado digital. Al igual que MPEG-7, este estándar también se basa en el lenguaje XML, donde encontramos dos conceptos:

    - *Digital Item (DI)*: La descripción del elemento multimedia que se distribuye o el “que.

    - *User*: Quien interactúa con el elemento multimedia o el “quien”.

  Dentro de los metadatos de un MPEG-21 podemos encontrar llamadas a metadatos MPEG-7, de esta manera el contenido del *DI* se puede enlazar con la descripción del recurso multimedia al que hace referencia.

Una vez el usuario a seleccionado el recurso a consumir, utilizando las propias preferencias tomadas desde el servidor de perfiles, el tipo y modelo del dispositivo, se le mandará el flujo de datos adaptados [7] a las características y posibilidades del dispositivo que esté utilizando el usuario.

Dicho esto, con un simple *login* del usuario a la máquina servidora podría obtener una selección de recursos al gusto y sin necesidad de tener que introducir preferencias en el instante de realizar la petición, otorgando de esta manera una mayor sencillez de cara al usuario doméstico.

Para poder realizar el estudio del sistema propuesto se montarán varios Servicios Web, uno para el *Profiling Server* y otro para el módulo de decisión.

Los Servicios Web son una herramienta útil para conseguir integración entre aplicaciones. Mediante un fichero de configuración se describe el servicio, detalladamente, indicando que debe recibir para funcionar correctamente y que retorna; con lo cual se pueden crear clientes mediante dicho fichero de configuración para poder realizar la comunicación con el servicio.

Dichos servicios proporcionarán la información necesaria a la interfaz Web que manejará el usuario.

En cuanto al servidor de perfiles se utilizarán bases de datos (BBDD) para poder almacenar los datos y gestionarlos de una manera rápida y sencilla.

## 1.3. Objetivos

Como ya se ha comentado en el apartado anterior, se implementará un servidor de perfiles que proporcione los datos del usuario de manera transparente.

Para poder implementarlo, se hará un estudio de los distintas tecnologías de gestión de datos, cómo los perfiles o metadatos, cómo recuperarlos y de qué manera tratarlos.

Una vez elegidas las tecnologías que se utilizarán, se empezará montando el servicio *Profiling*, desde el cual podremos tanto introducir datos, como extraerlos. El siguiente paso será desarrollar el módulo de decisión, la función del cual será explicada en capítulos posteriores.

Una vez estén listos el servidor de perfiles y el módulo de decisión se procederá al desarrollo de la interfaz de comunicación del usuario con los servicios.

Finalizado el montaje de toda la arquitectura, ésta deberá ofrecer al usuario la posibilidad de crear su perfil, poder verlo y modificarlo cuando lo necesite y recibir recomendaciones en base a los parámetros de su perfil.

# CAPÍTULO 2. ESPECIFICACIÓN

En este capítulo se realiza una prueba de concepto y se explica las funcionalidades y la especificación formal de todo el sistema.

## 2.1. Funcionalidades

La siguiente figura identifica que la interfaz realizará peticiones a los servicios *Profiling Server* y al módulo de adaptación.

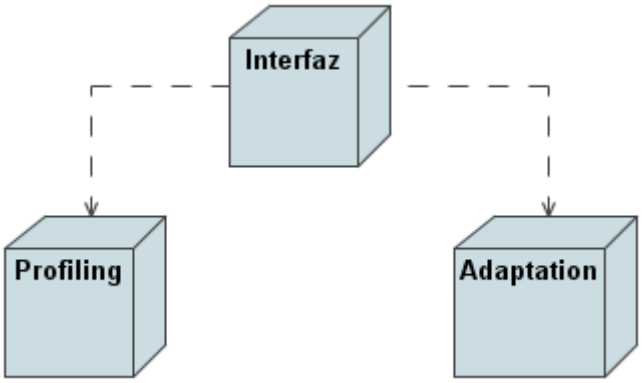



**Fig. 2.1** Esquema de interacción entre módulos

- **Interfaz de usuario:** Interfaz *web* que interactuará con los clientes de los servicios del módulo de adaptación y de la interfaz de gestión del servidor de perfiles.
- **Profiling Server:** El gestor de usuario y el gestor de *media* (parte dedicada a la gestión de metadatos de los recursos audiovisuales) forman parte de este servicio.
- **Adaptación:** Servicio de adaptación de contenidos, a partir de la información extraída por el servidor de perfiles.

A continuación se presenta un resumen sobre las funcionalidades del proyecto.

### 2.1.1. Gestión de recursos

En este apartado trata de la gestión realizada al usuario y la gestión realizada a los contenidos *media*.

En la *Figura 2.2* se muestra la diferencia entre un usuario básico y un usuario con permisos de administrador del sistema.

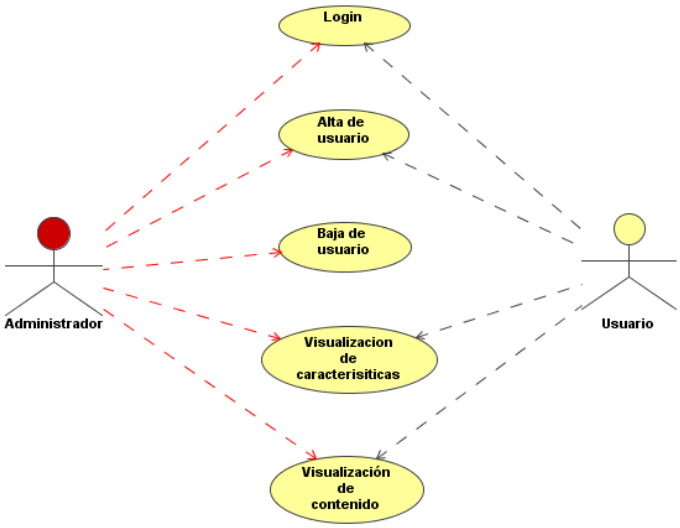



**Fig. 2.2** Diagrama de casos de uso para acceso al sistema

Como se puede observar, el usuario con permisos de administración es el único que puede efectuar la baja de un usuario en el servidor, por lo demás, se pueden realizar el resto de funciones, que serán explicadas a continuación, como usuario normal.

### 2.1.1.1. Registro del usuario

El registro de usuario se centra tanto en la gestión de datos personales del usuario como de sus preferencias almacenadas en la base de datos XML.

La gestión de los datos personales se realiza mediante un identificador de usuario o *userId*, único para cada uno, generado a partir del nombre y los apellidos. Con dicho identificador se podrán extraer los datos personales:

- Nombre
- Apellidos
- Nombre usuario
- Password
- Identificador de usuario
- Lista dispositivos registrados

Y también las preferencias o características:

- Características de usuario
- Características de red
- Características de entorno
- Características del dispositivo

Para poder almacenar las características de usuario será necesario el *userId*, el cual se asociará a los metadatos de estas características que se inserten en las bases de datos.

Hay que puntualizar que para poder extraer las preferencias de los dispositivos se tomará la lista de estos extraída de los datos personales y se realizará la búsqueda de éstos a partir del identificador de usuario y el identificador del terminal, en el caso de que sea necesario.

#### 2.1.1.2. Baja de usuario

Es necesario un Administrador para poder dar de baja a los usuarios, el cual, mediante el *nick* de usuario *y* el *password*, para mayor seguridad, pudiese borrar cualquier dato relacionado con el identificador de usuario proporcionado.

#### 2.1.1.3. Login

A través del *login* el usuario puede autenticarse y recibir el servicio teniendo en cuenta sus preferencias y características, de las cuales algunas fueron introducías por el usuario en su formulario de registro y otras se han ido almacenando a medida que el servidor recopilaba información sobre éste.

#### 2.1.1.4. Visualización de características de usuario

El usuario podrá realizar una petición para que se muestren sus características dentro del sistema, de esta manera podrá modificarlas y ajustarlas aún más a sus gustos y preferencias.

### 2.1.2. Gestor de recursos de media

Solamente los usuarios Administrador podrán incorporar nuevos recursos *media* al servicio.

Para la gestión de los recursos de *media* simplemente hay que conocer el identificador del recurso que se quiere consumir, ya que el XML de metadatos asociados del recurso tiene la misma identificación que éste.

#### 2.1.2.1. Feedback

Mediante el feedback el usuario podrá devolver al servidor una puntuación sobre la recomendación que se le ha ofrecido, de esta manera se podrá saber si las recomendaciones de contenido que realiza el servidor se ajustan a las preferencias del usuario, y poder de esta manera mejorar futuras recomendaciones.


**Fig. 2.3** Ejemplo de puntuación

### 2.1.3. Módulo de adaptación

La función de este módulo es poder tomar decisiones sobre los contenidos que se le han de mostrar al usuario que lo ha solicitado, por ejemplo, que codificación y tamaño de imagen utilizar dependiendo del dispositivo que el usuario esté utilizando en ese preciso momento.

Se ofrecerá la mejor adaptación según el contexto, en base a unos parámetros de entrada de usuario que son:

- Estado de anímico
- Preferencias

## 2.2. Especificación formal

El sistema se compone de varios módulos: interfaz de usuario, *profiling*, adaptación.

A continuación se explicarán los casos de uso para cada una de las funcionalidades descritas anteriormente.

### 2.2.1. Acceso al sistema

Un usuario puede acceder al sistema mediante el *login,* en el caso de que esté registrado. En caso contrario puede optar a registrarse, para posteriormente realizar un *login*.

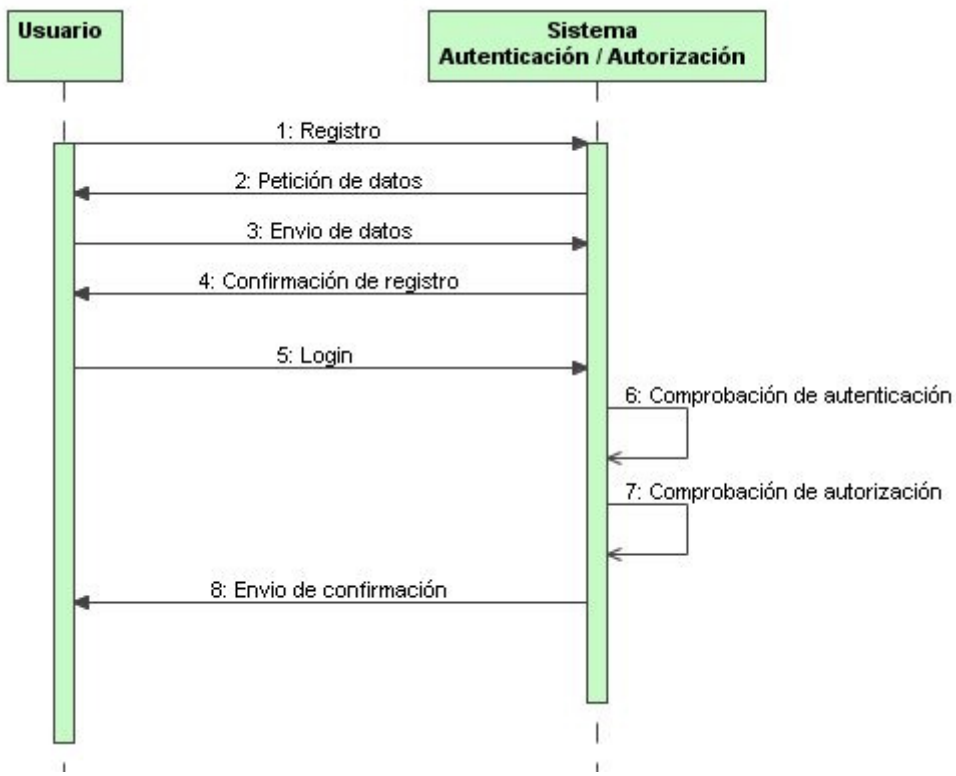



**Fig. 2.4** Esquema de entrada al sistema, con registro

### 2.2.2. Visualización de características

Tomando el ejemplo de la *Figura 2.4,* una vez hecho el *login*, el usuario podrá realizar una petición de comprobación de sus características. Se le mostrarán los datos más relevantes como pueden ser su nombre y apellidos, terminales asociados, localización, etc.

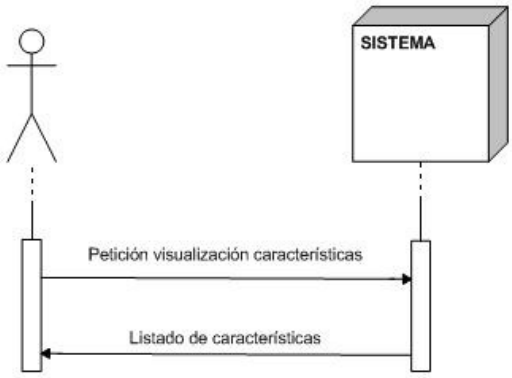



**Fig. 2.5** Ejemplo de visualización de características

### 2.2.3. Recomendación de contenidos

De la misma manera que en la función de visualización de características, una vez *logueado* en el servidor, se podrá hacer una petición de recomendación de contenido audiovisual.

Esta función preguntará al usuario el estado de ánimo en el que se encuentra y mediante este parámetro y la lista de gustos o preferencias del usuario el servidor realizará una selección de contenidos, en base a estas preferencias, y se le recomendará al usuario.

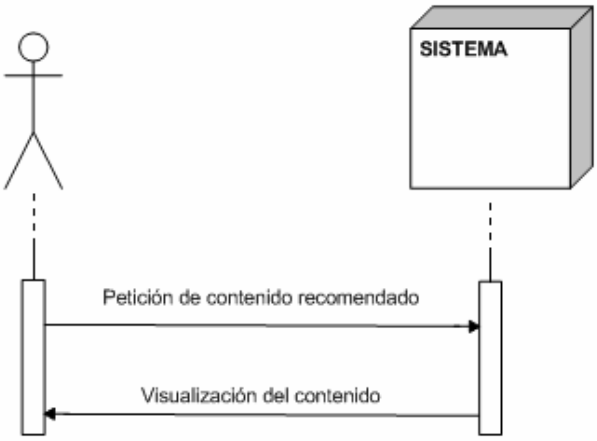



**Fig. 2.6** Ejemplo de contenido recomendado

# CAPÍTULO 3. ARQUITECTURA Y DISEÑO

En este capítulo se hará una breve explicación de la arquitectura en la que está basado este trabajo.

A continuación se muestra un esquema del diseño del sistema I3Media [8] donde se describen los módulos del proyecto y las comunicaciones entre ellos.

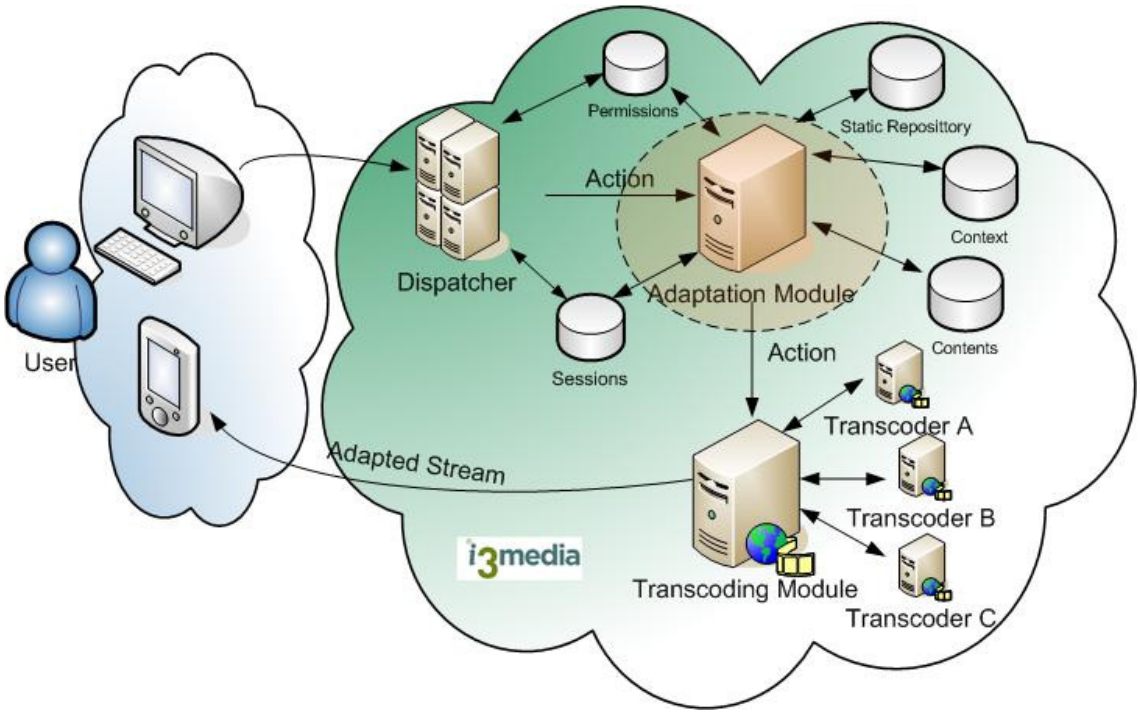



**Fig. 3.1** Esquema del sistema I3Media

Como podemos ver en la F*igura 3.1* I3Media está compuesto por varios módulos, entre los que podemos destacar el *Adaptation Module* y el *Transcoding Module*. Como se puede observar, el módulo de adaptación es el que decide sobre los contenidos a mostrar, mandando los datos necesarios al módulo de transcodificación para que éste pueda realizar las modificaciones que sean necesarias y mande el nuevo flujo de video a petición del usuario.

Para que esto sea posible el módulo de adaptación precisa de información adicional que se encuentra en los repositorios externos. Toda esta información es adquirida y organizada en el módulo de información.

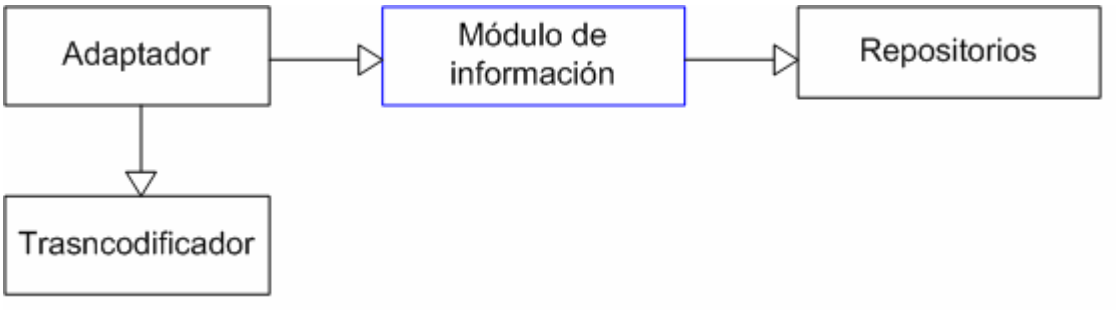



**Fig. 3.2** Esquema de la ubicación del módulo de información

Como se puede observar en la F*igura 3.2*, el módulo de información es el elemento que enlaza el módulo de adaptación con los repositorios que contienen los datos del usuario.

## 3.1. Arquitectura del módulo de información

El módulo de información es el encargado de realizar la comunicación con los distintos repositorios de datos, que son varias bases de datos externas. El servidor nos permitirá la extracción de los datos de los usuarios que hagan la demanda, el guardado de datos en los repositorios y la gestión de estos, lo que facilitará el poder organizarlos para un manejo más eficiente, ya sea para luego almacenarlos o utilizarlo en otros fines.

Mediante la interfaz de este módulo, podremos controlar el servicio de perfiles. Esto nos ofrecerá la posibilidad de realizar las funciones descritas en el capítulo 2, como por ejemplo:

- El *login* al servicio, donde se realizará una extracción de los datos personales del usuario para poder autenticarlo.
- Alta y la baja de usuarios.
- Visualización de las preferencias de usuario.
- Localización de los recursos multimedia, que realiza una extracción de datos del repositorio de recursos.

La arquitectura del sistema I3Media será simplificada para las finalidades del proyecto. El elemento servidor de perfiles que conecta con los repositorios, será extraído como un módulo más, como se puede comprobar en la *Figura 3.2* y de manera más detallada en la *Figura 3.3* o en los anexos [Anexo 3, apartado 3.1].

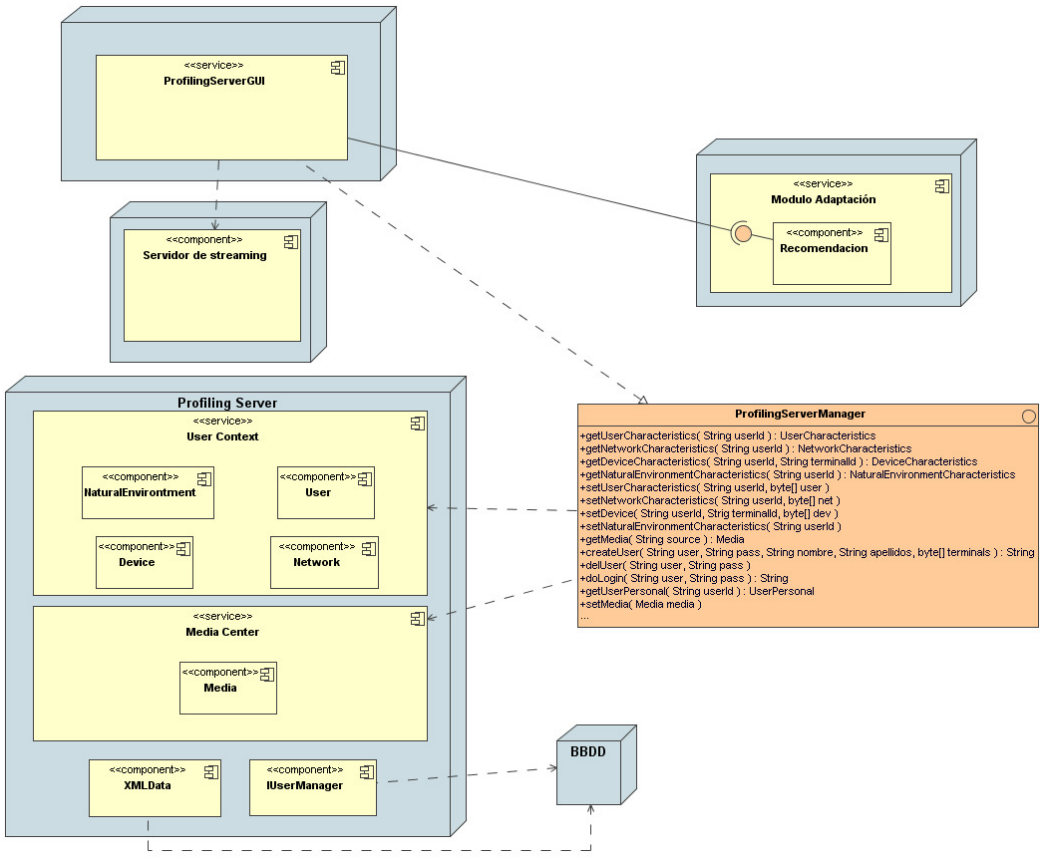



**Fig. 3.3** Arquitectura del módulo de información

La *Figura 3.3* muestra los siguientes elementos:

- **GUI:** Interfaz *web* que interacciona con los usuarios y con las interfaces de los módulos *Profiling Server* y Adaptación.

- **Profiling Server:** Contiene los componentes que ayudaran a extraer los metadatos de las BBDD, y otros componentes para organizar los datos extraídos de las BBDD.

- **Módulo de adaptación:** Contiene componente que servirá para realizar las recomendaciones.

- **Servidor de contenidos:** Servidor implementado en Red5 que realiza *streaming* en tiempo real, con un transcodificador *ffmpeg* y *mp3* para el formato de video *flv.*

Mediante el *frontend* de Usuario o interfaz Web, se realizará la comunicación entre el usuario y el módulo de adaptación, en este caso y adaptado al TFC, con el módulo *Profiling Server*. La comunicación entre la interfaz y el módulo se realizará mediante Servicios Web. Dicha interfaz la proporciona un servidor para atender las peticiones entrantes hechas por el usuario a través de un navegador. El servidor tiene desplegados los Servicios Web del servidor de perfiles y el del módulo de recomendación.

El Servicio Web del *Profiling Server* es el encargado de atender las peticiones de obtención de datos de usuario o de *media* provenientes de la interfaz. Este servicio realizará las comunicaciones pertinentes con las bases de datos para extraer la información y procesarla para poder utilizarla y enviarla a través de su interfaz *Web Service*.

Por el contrario, el Servicio Web de recomendación será el encargado de decidir qué contenidos audiovisuales recomendar a petición de usuario.

Ambos Servicios Web devolverán componentes que proporcionarán todos los datos del usuario, o bien el nombre del recurso media que se recomienda, respectivamente.

A continuación se mostrará una secuencia de interacción entre los distintos módulos del *Profiling Server* en la petición de recomendación por parte del usuario. Si se desea ver más grande, se encuentra en los anexos [Anexo 3, apartado 3.3].

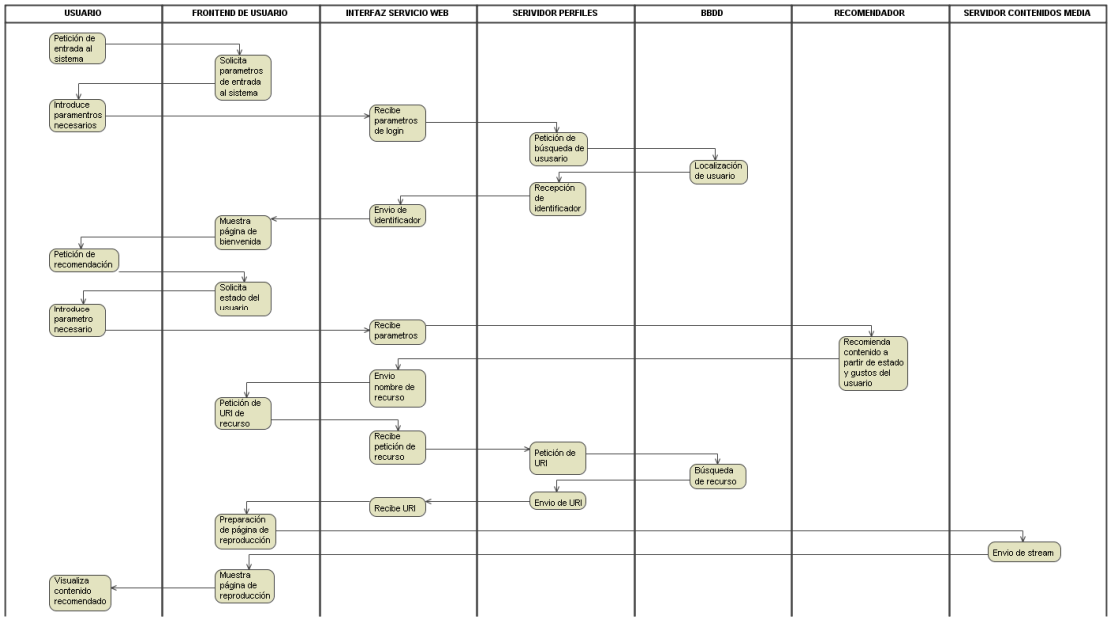



**Fig. 3.4** Petición de recurso audiovisual

Como se puede observar, la acción del usuario se limita a pedir entrada, introducir parámetros de *login*, pedir una recomendación de contenido introduciendo otro parámetro necesario y por último recibiendo el contenido para ser visualizado. Por detrás de estas peticiones hay varios movimientos entre los distintos elementos que componen el sistema, remarcando que el servidor de contenidos media es un servicio ajeno a la arquitectura del proyecto. Los elementos vitales en este servidor son los Servicios Web y sus interfaces, que realizan el paso de parámetros del cliente al servidor y viceversa.

Todas las peticiones que realice el usuario pasarán por el *frontend*, donde se recogerán y se mandarán vía Servicio Web el servidor de perfiles o al módulo de adaptación.

Dentro del módulo de adaptación, a partir del contexto del usuario, realizará una elección del contenido de posible agrado para el usuario.

### 3.1.1. Módulo Profiling Server

Como se ha comentado en el apartado 3.1, la interfaz cumple un papel importante dentro de este servidor de perfiles. Es el enlace entre los repositorios, los componentes que organizan los datos tanto de contexto de usuario como del Media Center y la interfaz gráfica.

En los siguientes apartados se explican dichos componentes y como están organizados. Las plantillas de los metadatos usados por para el contexto del usuario y del Media Center, se encuentran en los anexos [Anexo 5].

#### 3.1.1.1. Contexto usuario

Parte del componente dedicado a la gestión de los perfiles de usuario y datos personales. Se utilizan dos tipos de base de datos, dependiendo de los datos que se quieran almacenar del usuario.

Para los perfiles del usuario, organizados en MPEG-21, se pueden utilizar tanto en bases de datos relacionales con capacidad de tratado de metadatos o base de datos nativas XML.

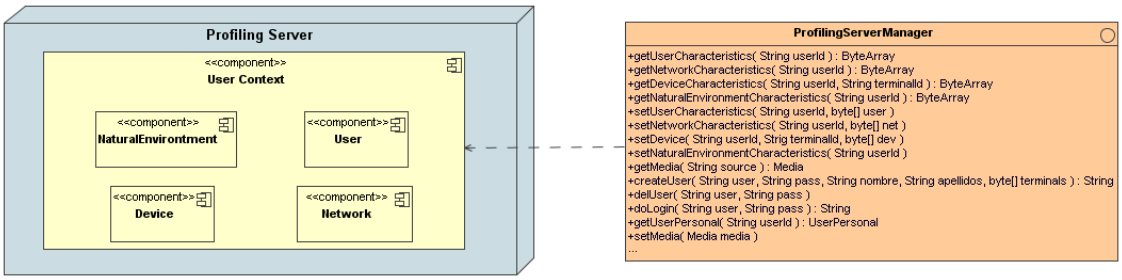



**Fig. 3.5** Vista del elemento *User Context*

La interfaz nos ofrece varias opciones para el control del contexto del usuario:

- Petición de perfiles y datos personales.
- Introducción de perfiles.
- Realización de *login*.
- Creación usuario.
- Borrado de usuario.

```
<UserCharacteristics xsi:type="PresentationPreferencesType">
      <Audio>
            <VolumeControl>0.85</VolumeControl>
      </Audio>
      <FocusOfAttention>
            <TextFocusOfAttention>
                  <Font fontColor="black" fontSize="10" fontType="arial"/>
            </TextFocusOfAttention>
      </FocusOfAttention>
</UserCharacteristics>
```

**Fig. 3.6** Ejemplo de metadatos de usuario

En la *Figura 3.6* podemos ver un ejemplo de metadatos de usuario en el que se muestra la preferencia sobre el volumen y formato de texto.

### 3.1.1.2. Media Center

Parte dedicada la gestión de los metadatos de los contenidos audiovisuales.

Los metadatos de los recursos *media*, se organizarán en MPEG-7, y por lo tanto se utilizará la misma base de datos que los perfiles de usuario, pero en distinto repositorio.

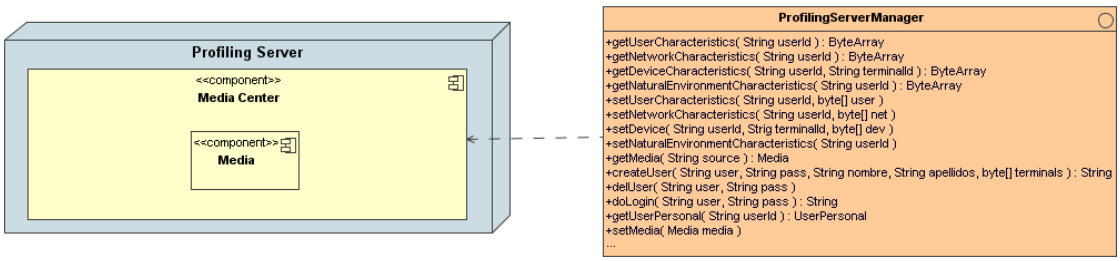



**Fig. 3.7** Vista del *Media Center*

Opciones ofrecidas por la interfaz para el *Media Center*:

- Petición de perfil del recurso media.
- Inserción de nuevo *media.*

```
<Video>
 <MediaInformation id="news1_media">
  <MediaProfile>
   <MediaFormat>
    <VisualCoding>
      <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1" colorDomain="color">
       <Name xml:lang="en">MPEG-1 Video</Name>
      </Format>
      <Pixel bitsPer="8" />
      <Frame height="288" width="352" rate="25" aspectRatio="1.333" structure="interlaced"/>
    </VisualCoding>
   </MediaFormat>
  </MediaProfile>
 </MediaInformation>
</Video>
```

**Fig. 3.8** Ejemplo de metadatos de recurso *media*

Como se puede observar, éste es un ejemplo de metadatos de un recurso *media* en el que se puede encontrar características técnicas de éste.

### 3.1.2. Módulo de Adaptación

El módulo de adaptación es el que se encarga de tomar las decisiones sobre el contenido que se ofrece al usuario.

En dichas decisiones el módulo decide sobre la mejor adaptación para el contenido teniendo en cuenta el contexto del usuario: modificación de la resolución del video para distintos dispositivos, aumento de volumen para usuarios con problemas auditivos, cambio de idioma dependiendo de la localización geográfica, etc.

Para realizar la adaptación, el módulo tiene varias opciones:

- Cargar unas reglas lógicas en base al contexto del usuario mediante *Prolog* [34]
- Lógica difusa (*FUZZY* [35]), mediante probabilidades.
- Algoritmo lineal, en el cual se realizará un *matching* de los elementos entrantes.

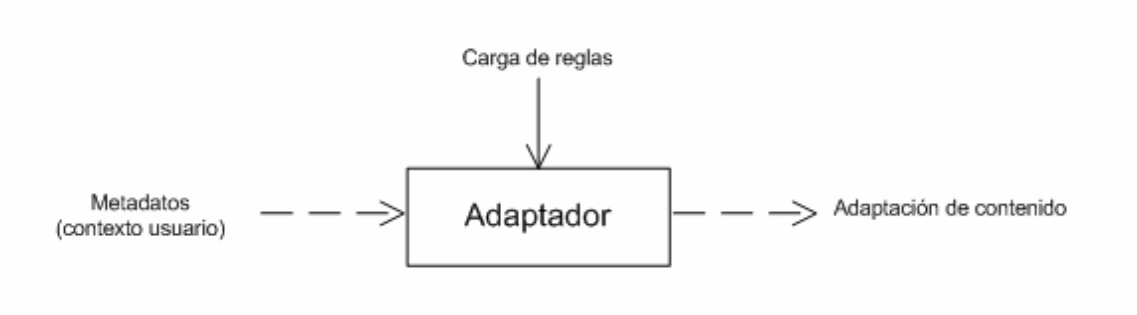



**Fig. 3.9** Entradas y salidas para el adaptador

Como se puede observar, según los metadatos de usuario que entren en el módulo y las reglas cargadas el adaptador decidirá en su respuesta de mejor adaptación.

### 3.1.3. GUI Profiling Server

El módulo *Profiling Server* GUI es la interfaz de control a través de la que el usuario podrá hacer uso de las funcionalidades del servidor. Dependiendo del tipo de permisos que el usuario posea podrá realizar más o menos funciones como hemos explicado en el capítulo 2, apartado 2.1.1.

La interfaz de este módulo interactuará con la interfaz del Módulo de Adaptación y con la interfaz del *Profiling Server,* por lo tanto será el encargado de mostrar al usuario los resultados de las funciones que pueda demandar: visualización de características, registro, recomendaciones, etc.

Esta interfaz *web* se ha desarrollado para poder proporcionar todas las funcionalidades descritas por el sistema. En ella se reflejarán los resultados obtenidos.

## 3.2. Diseño del módulo de información

En este capítulo se explica el diseño seguido para el desarrollo de los componentes del servidor y sus interfaces.

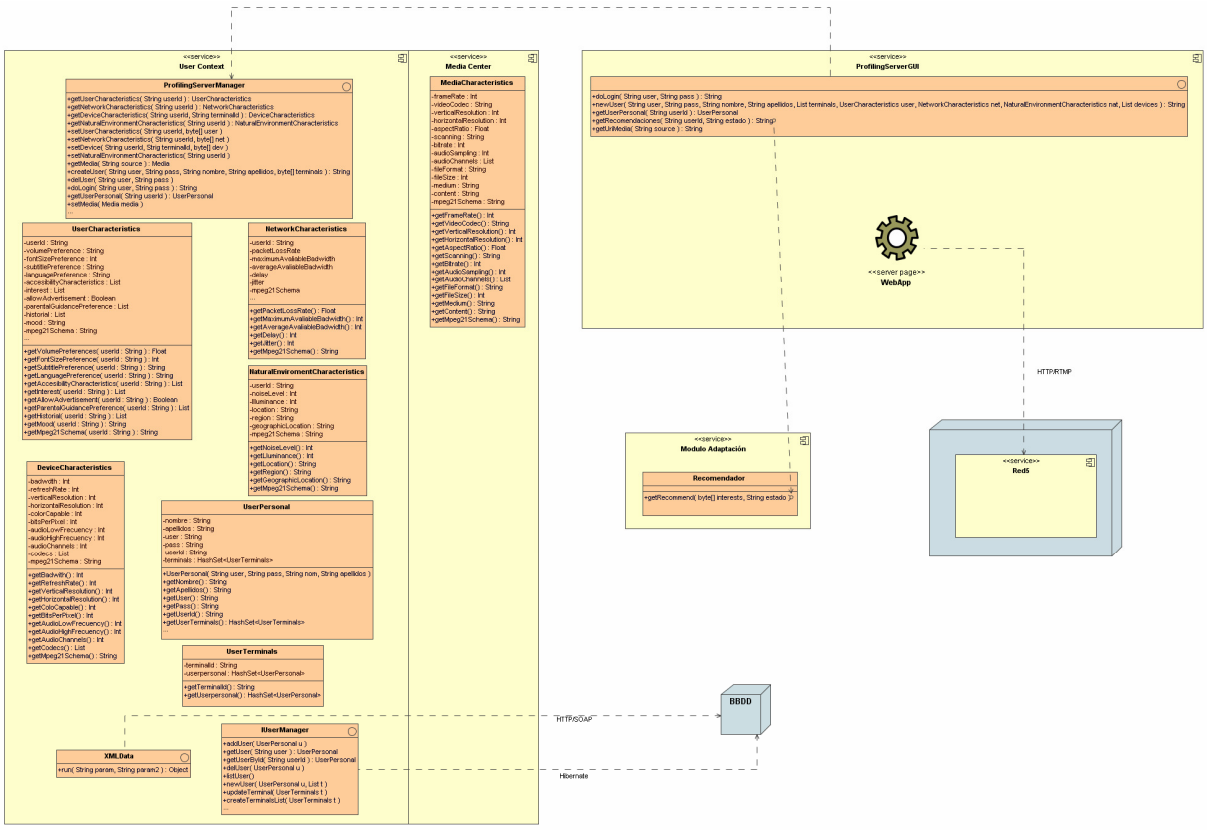



**Fig. 3.10** Diseño de la arquitectura del *Profiling Server*

En la *Figura 3.10* encontramos la arquitectura del servidor de perfiles a un nivel más bajo, donde se pueden observar los componentes más importantes de éste. Si se desea ver más grande, se encuentra en los anexos [Anexo 3, apartado 3.2].

## 3.3. Componentes del módulo de información

A continuación se mostrará la arquitectura del servidor de perfiles. En torno a el se centra este TFC. Por servidor de perfiles nos referimos a la interfaz de control, a la capa de lógica de negocio, la capa de integración y las bases de datos. En la siguiente tabla se explican que son cada una de estas capas y que función realizan dentro del servidor de perfiles.

**Tabla 3.1.** Descripción de elementos del módulo de información

| | |
|---|---|
| **Interfaz de control** | *Es la capa con la que interactúa el usuario y desde la que se atienden todas las peticiones entrantes.* |
| **Lógica de negocio** | *En la lógica de negocio encontramos todos los componentes del dominio de la aplicación que organizarán el perfil del usuario. Por ejemplo el componente de las características de la red, del entorno, etc.* |
| **Integración** | *Esta es la capa que realizará las funciones del servidor, tomando los datos de las bases de datos y pasándolos a la lógica de negocio para poder tratarlos más eficazmente.* |
| **BBDD** | *Parte del sistema donde se almacenarán todos los datos provenientes del usuario, ya sean sus preferencias o datos personales.* |

Una vez realizada la introducción a cada una de las capas, se mostrarán esquemas de cada una de ellas, con los elementos que las componen y cómo interactúan entre si.

### 3.3.1. Interfaz de control y lógica de negocio

Esta es la parte central del servidor de perfiles, donde se tratan los datos, ya sea los perfiles del recurso media o del usuario, o los datos personales de éste. La interfaz de control es la parte del servicio *web* localizada en el *ProfilingServerGUI*, que como se ha explicado en la tabla anterior será la que se comunique con el usuario.

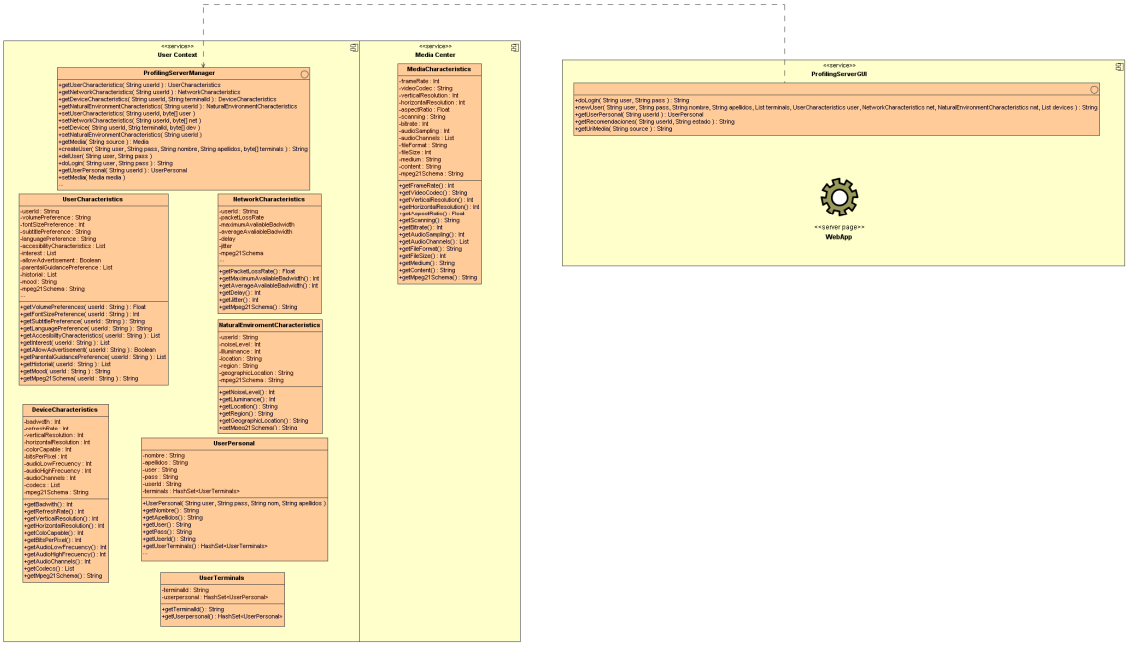



**Fig. 3.11** Interfaz de control y Capa de negocio

Desde esta capa se podrán introducir datos para poder realizar el registro de un nuevo usuario o la actualización en el perfil del usuario que lo desee. Los datos extraídos son organizados en Objetos, como pueden ser el *UserCharacteristics*, *NetworkCharacteristics* o *UserPersonal*.
La interfaz es la responsable de enlazar con las bases de datos, mediante la capa de integración y de esta manera poder realizar las tareas tanto de inserción como de extracción de datos.

Con esta interfaz es con la que se comunicará el *frontend* del usuario, vía Servicios Web, para poder realizar el tratamiento de datos de manera transparente al usuario.

### 3.3.2. Capa de integración y bases de datos

La capa de integración es, como se ha comentado anteriormente, la encargada de comunicarse con las BBDD y pasarle lo datos necesarios para poder almacenarlos.

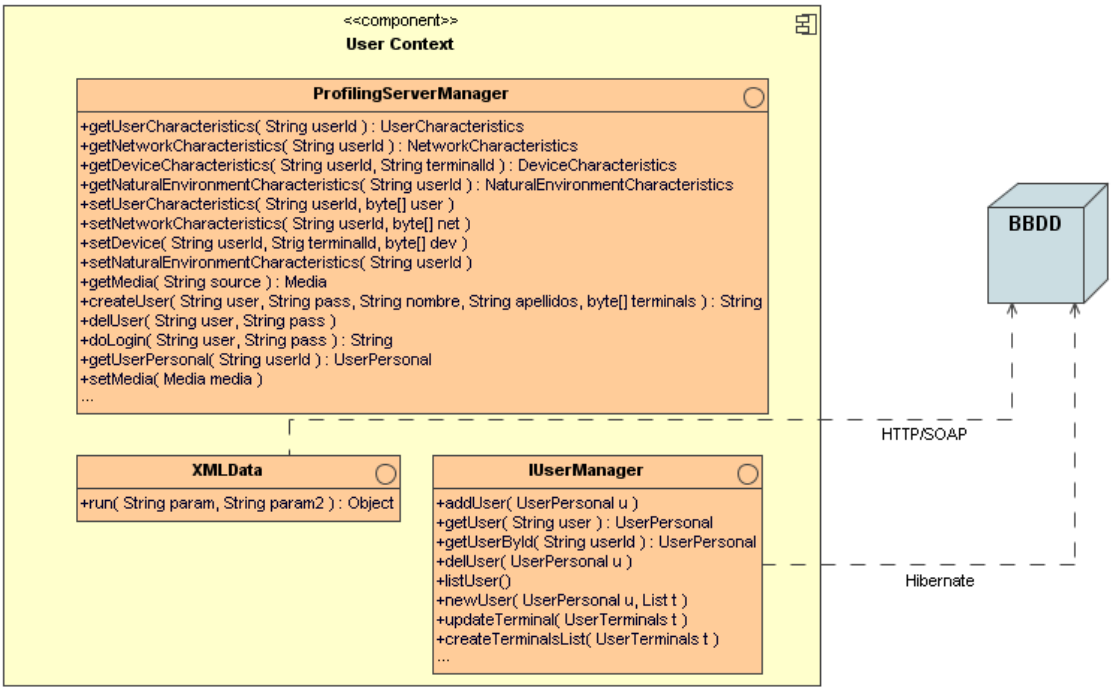



**Fig. 3.12** Capa de integración y bases de datos

La clase *XMLData*, nos ofrece la posibilidad de extraer la información de los XML almacenados en la base de datos, es decir, los perfiles del usuario o del recurso *media* que haga falta. Mediante una serie de parámetros, como por ejemplo el *userId*, se localizará dicho XML y este será tratado y transformado en el Objeto que sea necesario. Los Objetos están contenidos en la capa lógica de negocio.

La clase *IUserManager* es la encargada de gestionar los datos personales del usuario y cumple la misma función que *XMLData*, devolviendo los datos del usuario organizados en un Objeto, en este caso *UserPersonal*.

Para el almacenado de datos, los perfiles de usuario se guardan transformando los Objetos que contienen las preferencias de dicho usuario en un XML, el cual será enviado a la base de datos, enlazándolo con el identificador del usuario, y ésta los almacenará.

# CAPÍTULO 4. IMPLEMENTACIÓN

En este capítulos se explicaran los aspectos relacionados con la implementación teniendo en cuenta el diseño realizado en el capítulo 4.

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

Continuará la explicación de los pasos seguidos y las implementaciones realizadas.

## 4.1. Entorno de trabajo

El TFC ha sido desarrollado en la Fundación i2CAT, en el espacio de trabajo que poseen en la EPSC, donde han facilitado el material necesario.

Durante el desarrollo se han utilizado dos máquinas. En una de ellas, un portátil personal, se ha implementado el servidor *web* con los Servicios Web correspondientes, en otra máquina, proporcionada por i2CAT, se han implementado las bases de datos necesarias.

Se han utilizado dos sistemas operativos. Un GNU/Linux Ubuntu en el servidor y Windows XP en el servidor de repositorios. Se podría haber utilizado cualquier sistema operativo para el desarrollo de los servicio, puesto que todas las tecnologías utilizadas son multiplataforma y todo lo implementado es código abierto desarrollado en Java [18].

Para el servidor *web* se ha utilizado Apache Tomcat [17], que también nos ofrece la posibilidad de poder desplegar Servicios Web, que han sido desarrollados utilizando Apache Axis2 [18] en su versión 1.4. Tanto el servidor Tomcat como la plataforma Axis2 utilizan la máquina virtual de Java.

En las bases de datos se ha utilizado un servidor MySQL [20] para los datos relacionales y un servidor eXist [21] para los metadatos de usuario y *media* accediendo a ellos mediante *Xpath* [30] o Xquery [31].

Como plataforma de desarrollo para la implementación se ha elegido Java en su versión JDK 6 [28] de la plataforma Java 2 *Standar Edition*, de Sun Microsystems, que también incluye una máquina virtual Java JRE. Como entorno de programación se ha utilizado Eclipse [29], versión 3.2.

Para el desarrollo de la memoria se han utilizado conjuntamente Open Office de Sun Microsystems y Office 2003 de Microsoft. Para diseño de UML y esquemas ha sido utilizado MagicDraw 10 y Microsoft Visio 2003 respectivamente.

Se han utilizado dos servidores de control de versiones, CVS y posteriormente SVN. Estos servidores ofrecen la posibilidad de almacenar todo el desarrollo del proyecto. A parte de ser una medida de *backup* ofrece la posibilidad de mantener actualizada toda la información y organizarla en versiones en cualquier máquina en la que se trabaje. Es una herramienta idea para realizar trabajos en grupo ya que la información actualizada puede ser consultada desde cualquier máquina

## 4.2. Tecnologías y herramientas utilizadas

- **JDK 6 SE:** Versión de desarrollo de Java, que además proporciona el entorno de ejecución. Utilizado para desarrollar el software de la implementación, los Servicios Web, y ejecutar las herramientas necesarias.

- **J2EE:** Java 2 *Enterprise Edition*, plataforma de desarrollo sobre la que trabaja el *framework* Struts.

- **Struts 1.3:** Herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC [9]. Utilizada para crear la aplicación *web* con la que se conectarán los usuario.

- **Apache Tomcat 6:** Servidor de aplicaciones que permitirá ejecutar los Servicios Web necesarios y la aplicación *web*.

- **Apache Axis2 1.4:** Kit de desarrollo en Java que permite desplegar los Servicios Web dentro del propio servidor de aplicaciones Tomcat.

- **Hibernate 3.2:** Herramienta para el mapeo Objeto-Relacional, sobre plataforma Java, necesaria para poder almacenar objetos Java en una base de datos relacional.

- **Hibernate Annotations 3.2:** Utilidad para realizar los mapeos de las clases más rápido sin tener que utilizar un fichero extra XML para el mapeo. *Annotations* leerá las anotaciones de los ficheros .class declarados en el fichero de configuración de *Hibernate*.

- **MySQL Server 6:** Base de datos relacional utilizada para almacenar los datos personales de usuario.

- **eXist 1.2.1:** Base de datos XML nativa utilizada para almacenar todos los perfiles de usuario y de recursos *media*.

- **Digester 1.8:** Herramienta que trabaja sobre las *Application Programming Interface (*API) de parsing de ficheros XML, *Simple API for XML* (SAX) [33] y realizar proceso de extracción de datos de una manera más sencilla. Necesaria para pasar los metadatos de usuario y de *media* a clases Java.

## 4.3. Repositorios o BBDD

Para más información sobre las bases de datos tomar los anexos [Anexo 7].

Para poder gestionar correctamente todas las características del usuario estas se dividen en cuatro plantillas o esquemas. Las plantillas se pueden encontrar en los anexos [Anexo 5].

A cada uno de los esquemas se le asocia un XML, que variará dependiendo del usuario al que esté representando. Para la gestión de estos XML se pueden utilizar varias tecnologías y métodos.

Para la formación de los metadatos en XML se usan dos estándares ISO/IEC explicados en el capítulo 1, apartado 1.2. El MPEG-7, también conocido como "Multimedia Content Description Interface", se utiliza para los metadatos de los datos *media.* El MPEG-21, conocido como MPEG-21 Multimedia Framework, se utiliza para los datos relacionados con el usuario, es decir, sus preferencias. Ambos estándares están basados en XML.

El hecho de que los datos se organizan en dichos estándares utilizar bases de datos nativas es una solución idónea, las cuales ofrecen una eficiencia mucho mayor que las bases de datos relacional en términos de coste de conversión, ya que en las nativas no se hace ninguna conversión porque constantemente se introducen y se extraen datos en XML.

Para los datos personales del usuario se utiliza una base de datos relacional, que es más eficiente para organizar dichos datos en tablas y utilizar una clave única, por ejemplo el nombre del usuario, para poder realizar búsquedas.

### 4.3.1. Gestión de metadatos

Los requerimientos de almacenamiento de metadatos pueden ser divididos en dos categorías generales: centrado en lo datos que lo componen o centrado en el documento.

Para el almacenamiento de los datos XML podemos encontrar varias estrategias: almacenarlos nativamente o mapear los datos del fichero.

En el caso de escoger un almacenado por mapeo de datos, estaremos recurriendo a bases de datos relacionales con capacidades XML, por lo contrario utilizaremos bases de datos nativas.

### 4.3.2. Modelos de BBDD

En este proyecto se utilizarán dos tipos de bases de datos [4]. Dependiendo de los datos que queramos gestionar se utilizará una base de datos u otra.

#### 4.3.2.1. Modelo relacional

El modelo relacional de base de datos es el más utilizado actualmente ya que se ajusta a problemas reales y administra datos dinámicamente.

La idea principal de este modelo es el uso de relaciones entre tablas. Se pueden considerar de forma lógica como conjuntos de datos llamados *tuplas*[1], aunque a nivel físico las tablas pueden tener una estructura completamente distinta. Un punto fuerte del modelo relacional es la sencillez de su estructura lógica. Pero detrás de esa simple estructura hay un fundamento teórico importante del que carecen los BBDD de la primera generación, lo que constituye otro punto a su favor.

Para poder acceder a las BBDD y hacer consultas se utilizan dos herramientas matemáticas: el *álgebra relacional* y el *cálculo relacional*. Ambos utilizan operadores lógicos para la manipulación de los datos. El lenguaje *SQL, lenguaje declarativo*, implementado en los principales sistemas de gestión de las bases de datos, es el encargado de hacer la manipulación de los datos más transparente de cara al usuario.

#### 4.3.2.2. Modelo nativo XML

Una base de datos XML nativa [3] define un modelo lógico para un documento XML y almacena y recupera documentos de acuerdo a este modelo. Como mínimo el modelo debe incluir elementos, atributos, Parsed Character Data (PCDATA) y el orden del documento.

Estas bases de datos son diseñadas específicamente para almacenar documentos XML. Como el resto de bases de datos, se ofrecen características como transacciones, seguridad, acceso multiusuario, etc. La gran diferencia con el resto de bases de datos, y de ahí ganan el termino de *nativas,* es que almacenan los datos estructurados como XML sin necesidad de traducirlos a una estructura relacional o de objeto; por lo tanto, se preserva el orden del documento, secciones CDATA, entidad de uso, etc.

Las BBDD nativas también pueden ser utilizadas para integrar datos, aparte de poder controlar cambios en el esquema mucho más fácilmente en comparación con las BBDD relacionales. Dichas BBDD están diseñadas para trabajar con *eXtensible Query Language* (XQL), que sigue el mismo propósito que SQL. XQL está diseñado para trabajar con documento XML jerárquicamente estructurados y puede proveer características de consulta como filtros y *joins*.

[1] *Secuencia ordenada de objetos*

Esta ventaja ofrece la posibilidad de poder almacenar datos centrándose en el documento XML.

Para poder navegar por los datos y extraerlos o introducirlos se utilizan dos estándares, *Xpath* y *Xquery*.

### 4.3.3. Elección de BBDD

Se realizará una breve comparación entre dos bases de datos relacionales y otras dos nativas XML, y después un breve comentario sobre la elección de una u otra.

#### 4.3.3.1. MySQL vs. PostgreSQL

Tanto *MySQL* como *PostgreSQL [32]* son sistemas de almacenamiento de datos relacionales orientadas a objetos.

*MySQL* pertenece a MySQL AB (Subsidiaria de Sun Microsystems) y bajo licencia GNU GPL. *PostgreSQL* pertenece a PostgreSQL Global Development Group y bajo licencia BSD.

A continuación se muestra una tabla donde se comparan los dos tipos de bases de datos relacionales partiendo de las ventajas y desventajas de utilizar uno u otro sistema.

**Tabla 4.1.** Tabla comparativa entre MySQL y PostgreSQL.

| | Ventajas | Inconvenientes |
|---|---|---|
| MySQL | ▪ Mayor rendimiento. Mayor velocidad tanto al conectar con el servidor como al servir selects y demás.<br>▪ Mejores utilidades de administración (backup, recuperación de errores, etc.).<br>▪ Aunque se cuelgue, no suele perder información ni corromper los datos.<br>▪ Mejor integración con PHP.<br>▪ No hay límites en el tamaño de los registros.<br>▪ Mejor control de acceso, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos.<br>▪ *MySQL* se comporta mejor que *Postgres* a la hora de modificar o añadir campos a una tabla "en caliente". | ▪ No soporta transacciones, "roll-backs" ni subselects.<br>▪ No considera las claves ajenas. Ignora la integridad referencial, dejándola en manos del programador de la aplicación. |
| PostgreSQL | ▪ Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.<br>▪ Soporta transacciones y desde la versión 7.0, claves ajenas (con comprobaciones de integridad referencial).<br>▪ Tiene mejor soporte para triggers y procedimientos en el servidor.<br>▪ Soporta un subconjunto de SQL92 mayor que el que soporta *MySQL*. Además, tiene ciertas características orientadas a objetos. | ▪ Consume bastantes más recursos y carga más el sistema.<br>▪ Límite del tamaño de cada fila de las tablas a 8k (se puede ampliar a 32k recompilando, pero con un coste añadido en el rendimiento).<br>▪ Es de 2 a 3 veces más lenta que *MySQL*.<br>▪ Menos funciones en PHP. |

*MySQL* es el sistema de almacenado elegido puesto que sus tiempo de respuesta son más veloces con un consumo mucho menor de recursos. También posee un mejor acceso por tipo de usuario, por lo que para la implementación de permisos *MySQL* se adapta mucho mejor.

#### 4.3.3.2. Xindice vs eXist

Tanto *eXist* como *Xindice* [5] [6] son bases de datos dedicadas a la gestión y almacenamiento de metadatos o ficheros XML.

*Xindice* pertenece a Apache Software Foundation, bajo licencia Apache (parecida a GNU pero con matices). *eXist* es totalmente libre, fue fundado por Wolfgang Meier que actualmente es el coordinador de desarrollo de la aplicación.

**Tabla 4.2.** Tabla comparativa entre eXist y Xindice.

| Características | eXist | Xindice |
|---|---|---|
| Tecnología | Java | Java |
| Almacenamiento | Árbol B+ y ficheros paginados en DOM. | Ficheros indexados como texto nativo, códigos de Hoffman |
| Ficheros binarios | No | No |
| Soporte para transacciones | No | No |
| Autorizaciones | Sistema de permisos similar a Unix en colección y en documentos | No soportado |
| Estándares soportados | XPath/XQuery, XUpdate, Xinclude/XPointer | Xpath, Xupdate, AutoLinking |
| APIs | Si | No |
| Cliente GUI | Si | No |
| Índices | Estructurales, Rango, Texto completo | --- |

eXist es el sistema elegido. Este nos ofrece la posibilidad de dar permisos a los repositorios y de esta manera otorgar privilegios de acceso a los usuarios. Aparte nos ofrece un cliente GUI de control remoto que permitiría gestionar los repositorios sin tener que estar en local. Si se quiere ver la comparación en tiempos de búsqueda entre los dos sistemas, se encuentra en los anexos [Anexo 4].

En la administración de los datos desde el servidor, ya se para subir o pedir documentos, los métodos utilizados son prácticamente idénticos en los dos sistemas. Por lo tanto podría cambiarse de base de datos sin demasiadas complicaciones.

## 4.4. Gestión de datos y perfiles, Profiling Server

Ya explicadas las bases de datos que se han utilizado pasaremos a explicar como se gestionan los datos y metadatos para subirlos o pedirlos a las bases de datos.

### 4.4.1. Datos personales usuario

Estos datos son los referentes a nombre de usuario, apellidos, etc. que tendremos que convertir de objeto Java a tabla relacional.

Para esta tarea se usará *Hibernate* [11] [23], y éste utilizará el conector de *MySQL* para guardar los datos en el repositorio.

Mediante un fichero de configuración y los paquetes necesarios de *Hibernate*, se realiza dicha conversión. En el fichero de configuración se especifican los *.class* de las clases Java que queremos tratar. Realizando una llamada a la interfaz generada para *Hibernate* podremos tanto subir datos como recibirlos de la base de datos realizándose siempre la conversión objeto → tabla relacional y viceversa. En la siguiente figura se muestra un ejemplo.

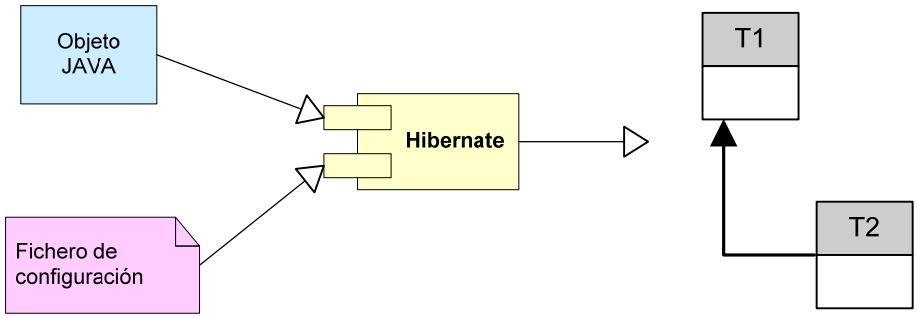



**Fig. 4.1** Ejemplo de conversión objeto → tabla

La conexión entre *Hibernate* y *MySQL* se realiza mediante el *driver JDBC*, el cual permite ejecutar operaciones sobre la base de datos. Tratados los objetos y con una conexión abierta y con un dialecto de comunicación cargado, el *driver* comprobará si están creadas las tablas donde se almacenan los objetos, que previamente se han definido dentro de la propia clase Java mediante *Hibernate Annotation.* Si no existen dichas tablas las creará automáticamente, si ya existe, subirá los datos directamente. En la siguiente figura se muestra un ejemplo.

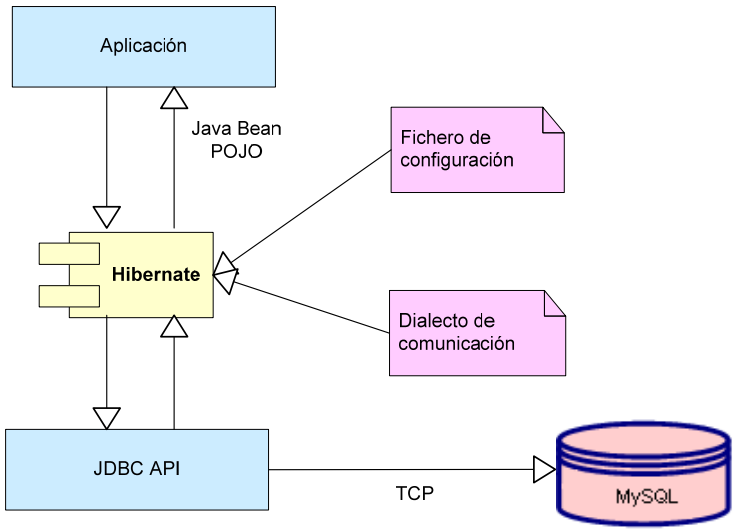



**Fig. 4.2** Ejemplo de comunicación con la base de datos

```
public void addUser(UserPersonal u) {

            Session s = HibernateSessionFactory.getSession();
            Transaction tran = s.beginTransaction();
            s.saveOrUpdate(u);
            tran.commit();
            s.flush();
            s.close();

}
```

**Fig. 4.3** Ejemplo de subida de datos a la base de datos

En este ejemplo de subida de datos de la *Figura 4.4*, vemos como se abre la sesión e inicia la transacción. Con la conexión establecida con el *saveOrUpdate* se realizará la conversión necesaria para que la base de datos pueda entender los datos y con el *flush* enviaremos todos los datos, cerrando después la sesión.

```
public UserPersonal getUserById(String userId){

            Session s = HibernateSessionFactory.getSession();
            Transaction tran = s.beginTransaction();
            UserPersonal u = (UserPersonal) s.createQuery(
            "FROM
i2cat.i3media.server.profilingserver.bl.userpersonal.UserPersonal
            as u where u.userId = :userId")
            .setParameter("userId", userId).uniqueResult();

            tran.commit();
            s.flush();
            s.close();

            return u;
}
```

**Fig. 4.4** Ejemplo de petición de datos a la base de datos

Este ejemplo de la *Figura 4.5* es justo lo contrario del de la figura anterior. En este caso realizaremos una petición de objeto mediante el *userId*. El proceso es el mismo que el de subida de datos, pero en este caso se mandará un *SQLQuery* el cual nos permitirá realizar la búsqueda mediante el identificador.

### 4.4.2. Metadatos perfiles de usuario y Media

Cuando nos referimos a los metadatos de usuario o recurso *media*, nos referimos a los ficheros XML que contienen los datos con las preferencias del usuario, de sus dispositivos, información sobre el recurso, etc.

La comunicación con la base de datos XML se realiza mediante *driver* de *eXist* y la *API* de conexión *XML:DB*, y la *API* de comunicación *XML-RPC*. El *driver* de *eXist* y XML:DB nos permiten realizar las peticiones a la base de datos, mientras que el *XML-RPC* nos permite enviar esas peticiones codificadas en XML y HTTP como protocolo de comunicación.

Tanto la entrada de nuevos documentos como la extracción de estos de la base de datos se realizan de manera muy similar. Los perfiles de usuario y los metadatos de los recursos *media* están organizados en colecciones o repositorios distintos. Para ver como están formadas las clases de petición y almacenado de metadatos, revisar los anexos [Anexo 2].

Para guardar los perfiles, se utiliza un método implementado en los *POJO* que contienen los datos. Este método toma todos los atributos y los organiza en un XML que será almacenado en la base de datos con el nombre de perfil y el *userId* mediante una clase implementada para la comunicación y subida de datos. De esta manera cuando se realice la búsqueda de perfiles, se utilizará este *userId*. En la siguiente figura se muestra un ejemplo.

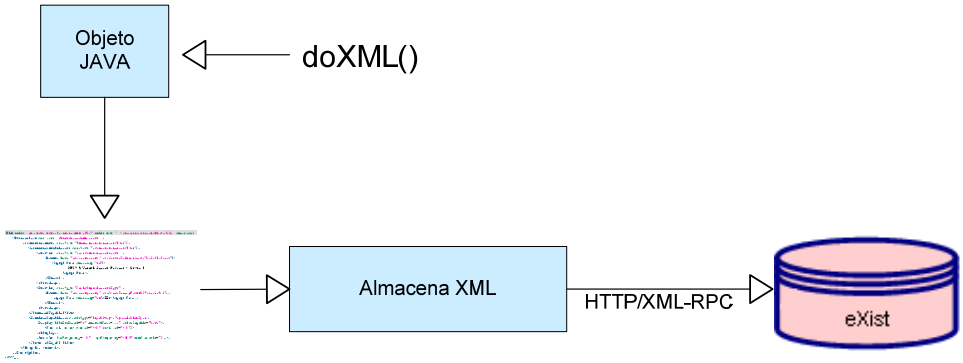



**Fig. 4.5** Ejemplo de almacenado de perfil

Si realizamos una petición de documento, pasaremos el *userId* a una clase implementada para la petición de documentos. Una vez tenemos el documento XML, hace falta volver a transformarlo en un objeto Java, por lo que se utiliza el *parser Digester* [24], que trabaja sobre *SAX*, el cual permite extraer datos de un XML y cargarlos en un *POJO.* Para la utilización de la herramienta *Digester* se necesitan cargar previamente las reglas de *parsing* [10]. En la siguiente figura se muestra un ejemplo.

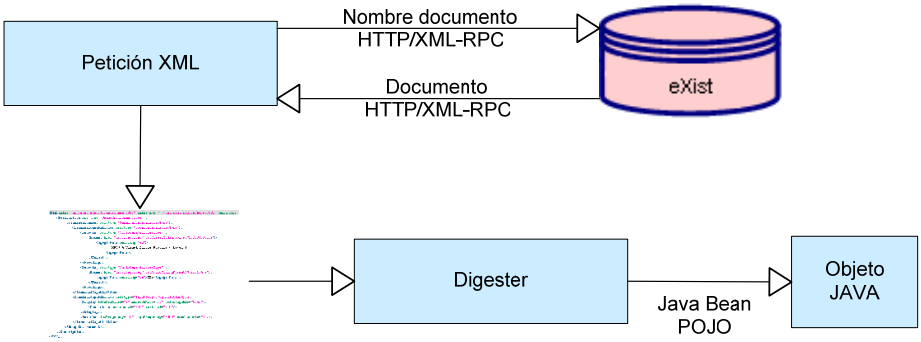



**Fig. 4.6** Ejemplo petición de documento y *parsing* con Digester

```
digester.addObjectCreate("DIA",UserCharacteristics.class);
digester.addSetProperties("DIA","xmlns","mpeg21Schema");
digester.addBeanPropertySetter(path+"Audio/VolumeControl","volumePreferences" );
```

**Fig. 4.7** Ejemplo de regla de *parsing*

Como se puede observar en la *Figura 4.7*, primero se indica el *POJO* donde se desean los datos y seguidamente se indica que se desea realizar cuando encuentre el *tag*, que quieres obtener, en el caso de que sea un atributo del *tag*, y el atributo del *POJO* donde se quiere que lo guarde.

*Digester*, al trabajar sobre SAX, obtiene las ventajas de éste, pero también sus desventajas. El procesador de los documentos se realiza de una manera diferente a la herramienta *Document Object Model* (DOM) [36] el cual genera un árbol jerárquico del documento en memoria (necesita el documento entero cargado en memoria) mientras que SAX procesa la información en XML conforme esta se va presentando (evento por evento). En la siguiente tabla podremos comprobar sus ventajas y desventajas:

**Tabla 4.3.** Tabla sobre características de SAX

| Ventajas | Desventajas |
|---|---|
| • SAX es un *parser* ideal para manipular archivos de gran tamaño, ya que no ocupa generar un *árbol en memoria* como es requerido en DOM.<br><br>• Es más rápido y sencillo que utilizar DOM | • La sencillez antes mencionada tiene su precio, debido a que SAX funciona *por eventos* no es posible manipular información una vez procesada, en DOM no existe esta limitación ya que se genera el *árbol jerárquico en memoria* y es posible regresar a modificar nodos. |

### 4.4.3. Interfaz de control

Mediante esta interfaz podremos realizar todas las funciones, menos la de recomendación, especificadas en el capítulo 2, apartado 2.2.

## 4.5. Servicios Web

A continuación, una vez hecha la implementación de los diseño UML de las clases Java, se explicará el desarrollo seguido para la creación de los Servicios Web del servidor de perfiles y del módulo de adaptación.

Mediante los Servicios Web, que son un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, podremos hacer el paso de datos entre la interfaz *web* con la que interactúa el usuario y la interfaz del *Profiling Server*. La conexión entre las peticiones de usuario desde la interfaz y los Servicios Web se realiza mediante HTTP/SOAP. Los Servicios Web están desplegados en un servidor Tomcat con el *kit* de desarrollo de Web Services de Apache, Axis 2 [14].

Para el desarrollo de los Servicios Web de este proyecto se ha utilizado unos *plugins* para el entorno de programación Eclipse:

- *Axis Code Generator Wizard* → Generación del fichero WSDL o de los clientes java del Servicio Web.
- *Axis Service Archive Wizard* → Genera el paquete que contiene el Servicio Web.

Estos *plugins* nos ayudan a generar el fichero de mapeo de servicios *WSDL* [12], montar el servicio [13], desplegarlo en el Tomcat, y generación de los clientes a partir del *WSDL*. En los anexos hay un ejemplo de desarrollo de Servicios Web [Anexo 6].

El *framework* utilizado para el enlace de los datos o *databinding* es el *ADB*. Este *framework* nos realizará la conversión de los objetos para poder intercambiarlos con el Servicio Web.

*WSDL* describe la interfaz pública a los Servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen de manera abstracta y se ligan después al protocolo concreto de red y al formato del mensaje.

### 4.5.1. Servicio Profiling Server

Para la generación de este servicio, el *plugin* de Axis2 realizará un *maping* de la clase que se le indique, en este caso será la interfaz que contiene todas las funciones, y generará el fichero *WSDL*. Una vez montado el servicio y desplegado en el servidor de aplicaciones Tomcat, se procede a probar el correcto funcionamiento del propio servicio.

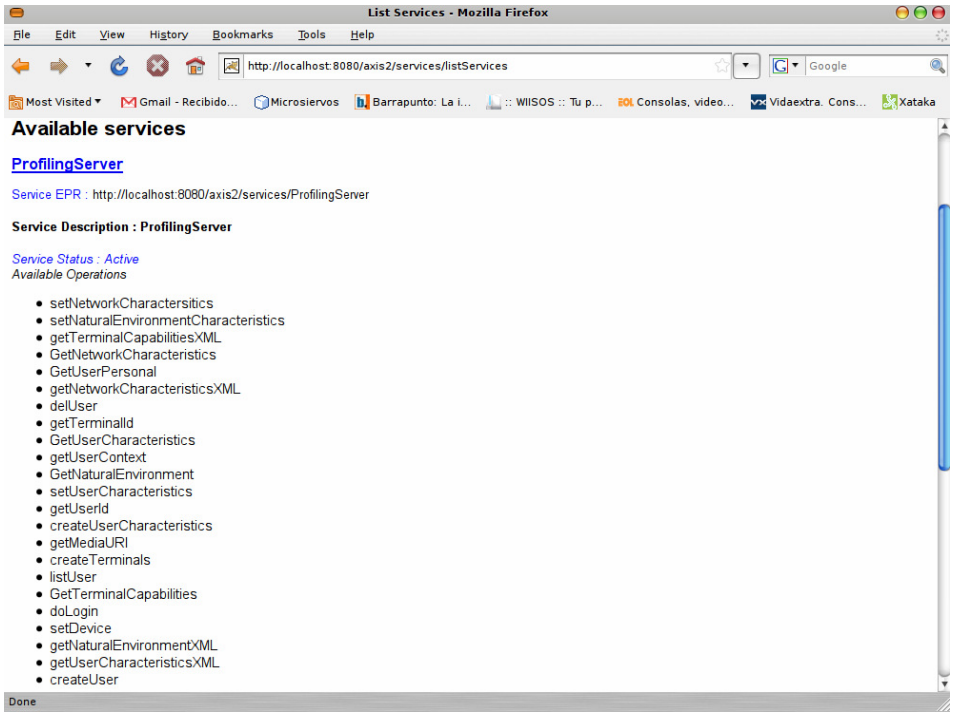



**Fig. 4.8** Servicio Web operativo

Como se puede observar, el Servicio Web, ofrece todas las funcionalidades de la interfaz del *Profiling Server* que ha sido mapeada por el *WSDL*. Mediante este Servicio Web se puede hacer *login,* registrar a un usuario, realizar una petición de perfil de características de usuario, de características de Terminal, etc.

El paso de los perfiles de usuario o de datos personales se realiza mediante serializado de objetos, es decir, convertir los *POJOs* en una cadena de bytes o *ByteArray*. Este pase se realiza para poder saltar el problema de envío de objetos complejos vía Servicio Web como puede ser un *ArrayList* o un *Vector*. También facilita la gestión de los datos ya que tanto en el cliente como en el servidor se usan los mismos *POJOs*.

Los objetos serializados, es decir, los *ByteArray* se encapsularan en un *ByteArraySource* y el *contentType* específico para la cadena de bytes, *base64binary*. Con el nuevo objeto encapsulado se podrá mandar a la interfaz del cliente del Servicio Web *ProfilingServer* mediante un *DataHandler*. Gracias a estos pasos, los objetos podrán ser transferidos de un lado a otro sin problemas, al llegar al servidor simplemente se deserializa el objeto en cuestión ya que el propio Servicio Web desencapsula el *DataHandler*, devolviendo directamente el *ByteArray.* En la siguiente figura se muestra un ejemplo.

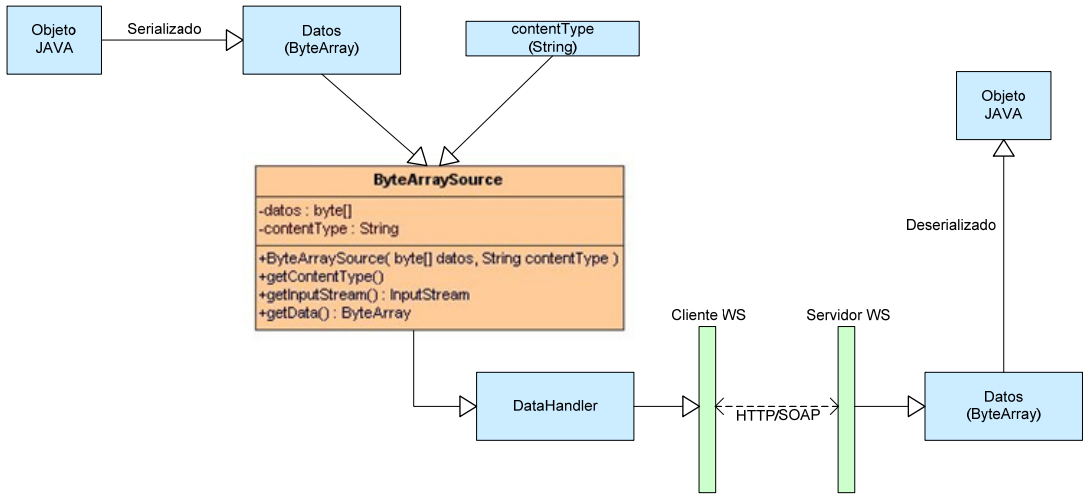



**Fig. 4.9** Ejemplo de encapsulado y desencapsulado

### 4.5.2. Módulo Adaptación

El módulo de adaptación, es una pequeña recreación del módulo real, que es el que se implementa en el sistema I3Media, este TFC está centrado en la gestión de perfiles, por lo tanto para la recomendación de contenidos, se ha creado un componente *dummy* que decidirá con los datos que le pase la interfaz *web*, que recurso audiovisual recomendar al usuario.

Este componente implementa un algoritmo lineal, que realiza un *matching* o unión de elementos para dar un resultado. Según el tipo de unión que resulte, el algoritmo otorgará una puntuación, con la que posteriormente determinará que recurso mandar al cliente. En la siguiente figura se muestra un ejemplo.

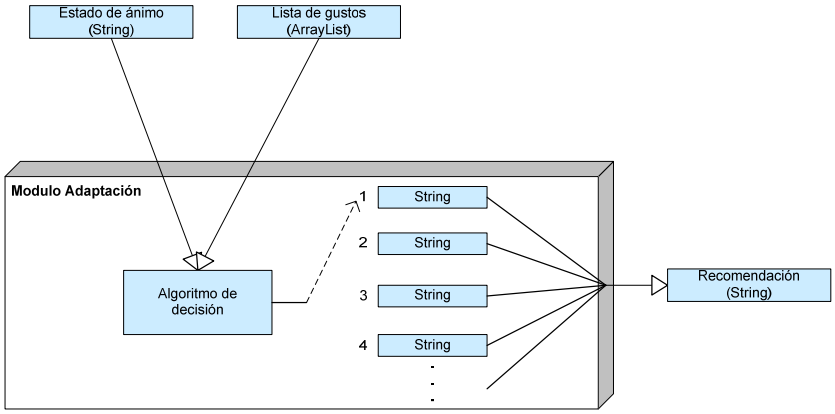



**Fig. 4.10** Ejemplo de decisión

El algoritmo utiliza como parámetros de entrada los gustos de géneros de película de un usuario y su estado de ánimo. De la puntuación resultante se elegirá el recurso a recomendar. Estos recursos están en una lista, ordenados por un número que es utilizado para determinar si es recomendable.

### 4.5.3. Interfaz web

La interfaz se ha desarrollado para poder mostrar los resultados del funcionamiento del servidor de perfiles, ya que en el sistema I3Media no está diseñada. Para el desarrollo de ésta se ha utilizado el patrón de arquitectura de software MVC, la cual separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- **Modelo:** Esta es la parte del sistema formada por las clases *form*. Es la parte encargada de pasar datos desde la vista al controlador.

- **Vista:** La vista son los JSPs creados para la interfaz *web* los cuales mostraran los resultados a los usuarios.

- **Controlador:** Parte formada por el *servlet que* responde a las peticiones de los usuario y realiza una u otra acción dependiendo del resultado obtenido por el modelo, con lo cual aplica cambios en la vista.

La interfaz *web* nos ofrece la posibilidad de registrarnos en el sistema, realizar un *login*, comprobar las características más relevantes del usuario en el sistema y realizar una petición de recomendación de contenidos. Los JSPs se dinamizarán con el *framework Struts* [22], lo que nos facilitará el control sobre las páginas *web* mostrando una u otra dependiendo del resultado que devuelva el controlador.

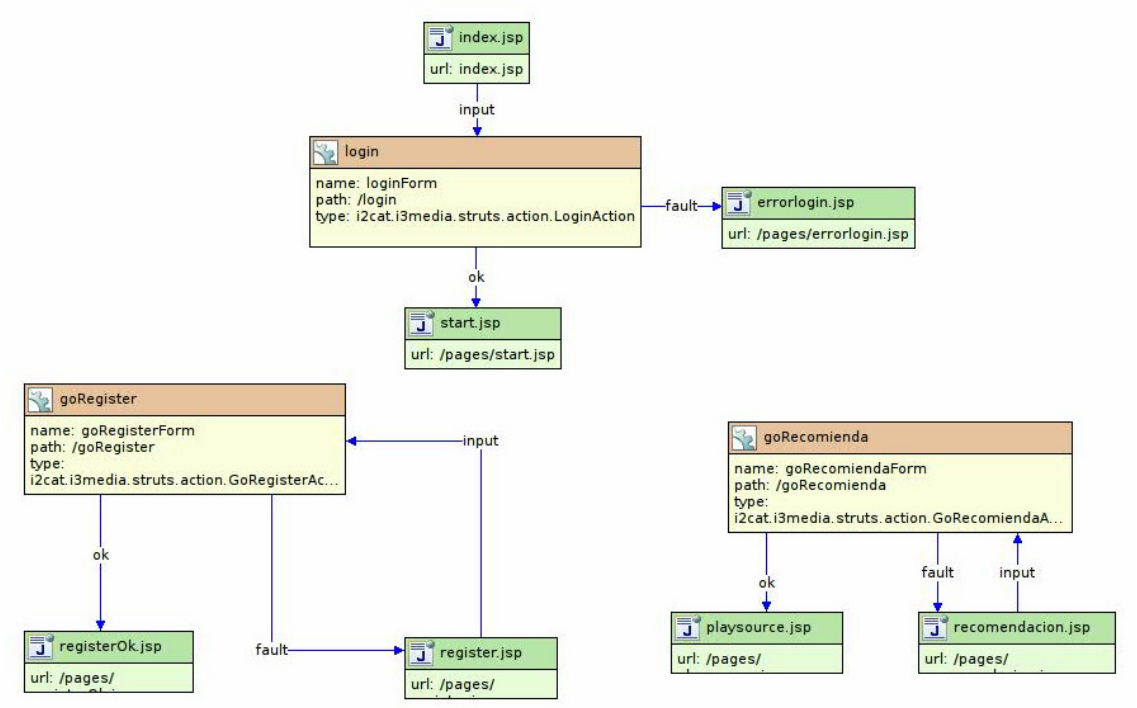



**Fig. 4.11** Esquema de configuración de *Struts* para la interfaz *web*

Como se puede comprobar en la *Figura 4.11* los datos extraídos de los formularios en los JSPs (vista) se mandarán *form* (modelo) y se tratarán en los *ActionForm*, que son los encargados de efectuar las operaciones con dichos datos. Después de realizar las operaciones, el *ActionForm* (controlador) devolverá un resultado con el cual, y mapeando el fichero de configuración, se devolverá un JSP u otro.

Para realizar un registro, se debe de rellenar un formulario con los datos personales del usuario y algunas características de éste. Una vez enviado al servidor se nos informará del éxito del registro o de cualquier error producido, con lo que se nos pedirá de nuevo el rellenar el formulario.
Para realizar un *login* simplemente tendremos que introducir el *user* y el *pass* especificados en el registro.

En la visualización de las características simplemente deberemos de esperar a que se muestren en la página organizadas en tablas.

Cuando realicemos una petición de recomendación, se nos pedirá que introduzcamos el estado de ánimo en el que nos encontramos. En la siguiente vista se nos presentará el titulo del recurso recomendado y un reproductor en *flash* para poder reproducirlo.

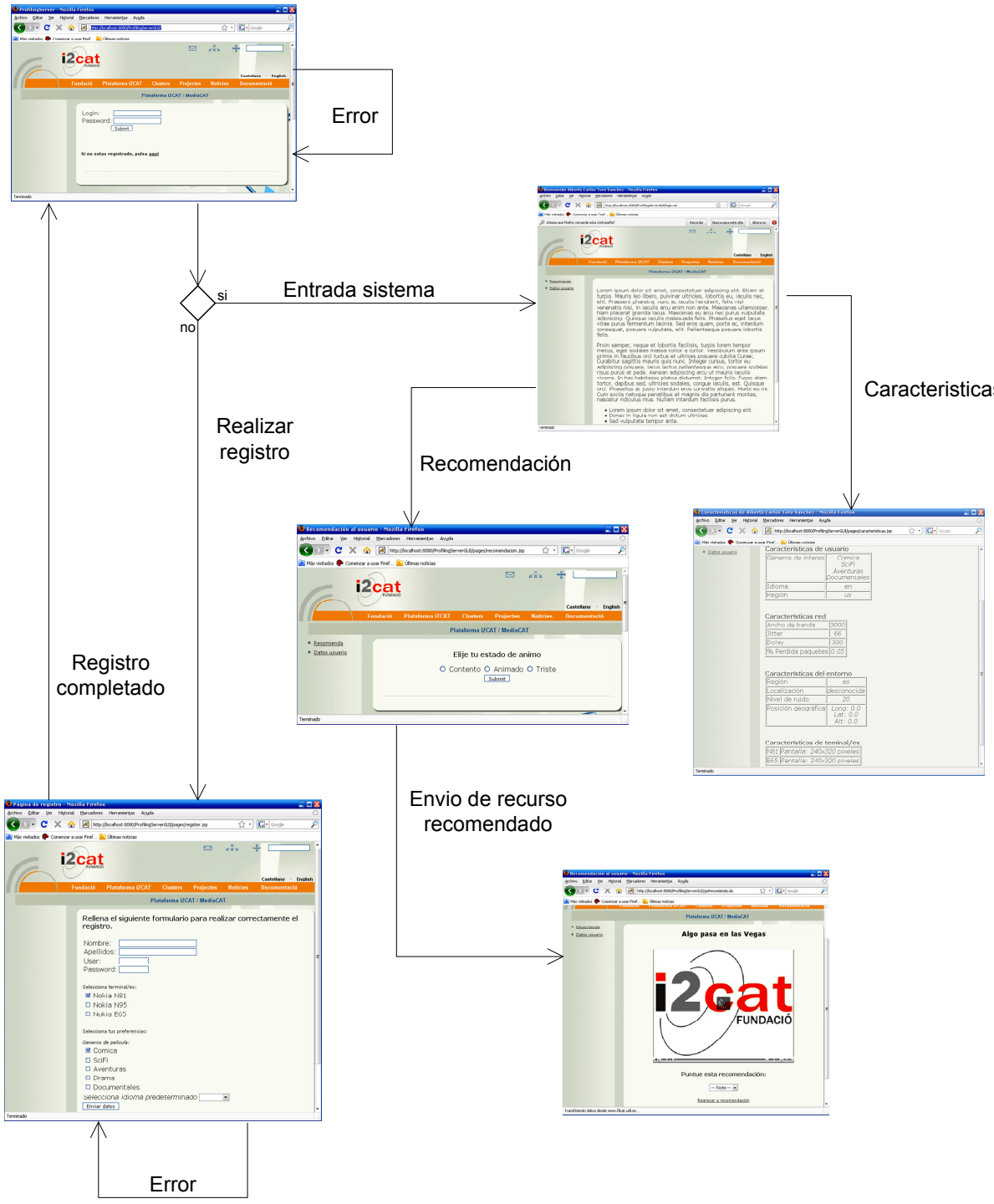



**Fig. 4.12** Interacción entre las páginas *web*

## 4.6. Servicio de contenidos

Este servicio puede ser implementado por varios servicios; se contemplan dos opciones en este proyecto: con una aplicación VLC, la cual transcodifique el vídeo a petición, y el otro servicio es utilizar un servidor de Red5 [27], parecido al servicio ofrecido por Youtube, el cual es muy interesante ya que es muy flexible, es una tecnología emergente y muy útil a la hora de ofrecer servicio vía Web. Se utilizará Red5, aprovechando que ya hay un servidor en marcha.

Servicio externo al sistema basado en la tecnología Red5. Es un servidor Open Source para entregar contenido en streaming en Flash. Para ello utiliza el protocolo *Real Time Messaging Protocol* (RTMP) con lo cual se puede transmitir contenido en tiempo Real. Utiliza la sintaxis A*ctionScript Communication*, y que contiene todas las cualidades del *Flash Media Server* de *Adobe*, con lo cual se pueden utilizar aplicaciones de comunicación en tiempo real.

El servidor esta desarrollado enteramente en JAVA que le otorga la cualidad de ser multiplataforma. Caracteristicas más importantes:

- Soporta los formatos FLV (*Flash Video, codec ffmpeg)* y MP3.
- Guardado de Streaming del cliente.
- Share Objects.
- Publicación Live Streaming.
- Soporte para AMF o *Flash Remoting.*

Cuando se realice la recomendación de contenido, se buscará el metadato relacionado al contenido recomendado. De este se extraerá al URI que conectará con el servidor de contenidos y al recurso solicitado.

Para la reproducción del contenido en tiempo real se utilizará un *Flash Player* llamado *JW Player* [26], que será cargado desde Internet y empotrado en el JSP.


**Fig. 4.13** Ejemplo del *Flash Placer*

# CAPÍTULO 5. PLANIFICACIÓN Y COSTES

Este capítulo está dedicado a mostrar las distintas tareas realizadas durante todo el proyecto para conseguir los objetivos marcados en un comienzo y el tiempo de dedicación para cada una. Se realiza también una división en grupos de tareas que se muestra a continuación:

- **Estudio previo**: Tarea que engloba el trabajo de búsqueda de información, estudio de tecnologías posibles a utilizar y desarrollos de pequeños prototipos o aplicaciones para probar distintas opciones.

- **Diseño**: Grupo que asocia todas las tareas de diseño de componentes, esquemas UML, definición de la arquitectura y de la aplicación web.

- **Implementació*n***: Tareas relativas al desarrollo de software teniendo en cuenta los esquemas diseñados.

- **Testeo**: Tarea dedicada a las pruebas realizadas la implementación del software y pruebas finales.

- **Documentación**: Realización de documentos y posteriormente de esta memoria durante la realización de este trabajo.

## 5.1. Tiempo dedicado

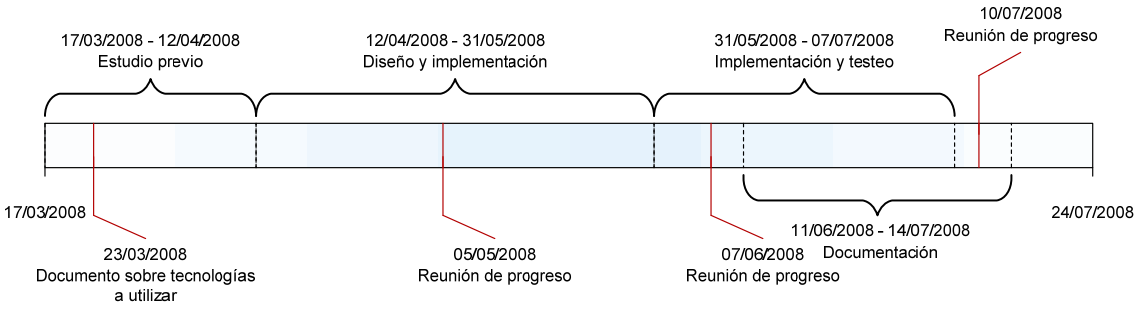



**Fig. 5.1** Escala de tiempo de trabajo

Como se puede comprobar en la escala, las tareas se han ido desarrollando de manera secuencial. Durante el estudio previo, se hicieron varias pruebas con las distintas tecnologías que se han ido utilizando, con su correspondiente documento para encontrar diferencias entre ellas, búsqueda de prestaciones que ayudasen o facilitasen llegar a los objetivos propuestos en este proyecto.

Se puede observar también que durante el diseño de componentes, se fueron implementando de otros, probándolos por separado; para finalizar, se hicieron

pruebas de funcionamiento entre los distintos componentes y por último, pruebas globales.

## 5.2. Tareas realizadas

**Tabla 5.1.** Desarrollo de tareas con sus respectivas horas

| Tarea | Descripción | Tiempo |
|---|---|---|
| Estudio previo | Concretar requisitos y objetivos, búsqueda de información. | 40 h |
| Estudio previo | Búsqueda de tecnologías y pruebas. | 80 h |
| Estudio previo | Realización de documentación guía. | 15 h |
| Estudio previo | Instalación de aplicaciones necesarias, realizar pequeñas pruebas con las tecnologías seleccionadas. | 60 h |
| Diseño | Creación de los primeros esbozos del sistema. | 30 h |
| Implementación | Desarrollo de pequeñas aplicaciones para comunicación con las bases de datos. | 3 h |
| Diseño | Creación de los UML de los componentes. | 40 h |
| Diseño e implementación | Diseño y desarrollo de componente especializado en la extracción de XML de la base de datos. | 6 h |
| Implementación | Agregar *parser* a la capa de integración para el tratado de datos provenientes de la base de datos XML. | 15 h |
| Implementación | Desarrollo de las clases de la capa lógica de negocio y juntarlo con el *parser*. | 4 h |
| Implementación | Desarrollo métodos para la creación de XML de las clases de la capa lógica de negocio. | 4 h |
| Implementación | Agregar a la capa de integración componente para la subida de metadatos a la base de datos XML. | 4 h |
| Diseño e implementación | Diseño y desarrollo de componente para la extracción e inserción de datos de la base de datos relacional y agregarlo a la capa de integración. | 25 h |
| Implementación y testeo | Desarrollo de las clases encargadas de tratar los datos personales del usuario. | 4 h |
| Implementación | Desplegar Servicio Web con los componentes de tratado de datos y los conectores de las bases de datos. | 3 h |
| Implementación y testeo | Desarrollo de un cliente del Servicio Web y testeo de la extracción de datos remotamente | 8 h |
| Implementación | Agregar un cliente del Servicio Web a la interfaz *web* | 4 h |
| Diseño e implementación | Creación de interfaz *web* para el control del Servicio Web. | 10 h |
| Implementación | Desarrollo de un sistema de autenticación para la interfaz *web*. | 2 h |
| Implementación | Agregar a la interfaz opción de registro de nuevo usuario. | 3 h |

| | | |
|---|---|---|
| Implementación | Agregar a la interfaz opción para que el usuario pueda revisar sus características. | 5 h |
| Diseño e implementación | Diseñar y desarrollar el módulo de adaptación. | 4 h |
| Implementación y testeo | Desplegar un nuevo Servicio Web con el módulo de adaptación y probar ejecución remota. | 2 h |
| Implementación | Agregar un cliente del Servicio Web del módulo de adaptación | 2 h |
| Implementación | Desarrollo de un recomendador de contenidos para la interfaz *web.* | 4 h |
| Implementación | Agregar un reproductor *Flash* para mostrar los contenidos recomendados al usuario. | 6 h |
| Testeo | Probar todo el sistema en conjunto y corregir errores. | 50 h |
| Documentación | Realización de esta memoria. | 130 h |

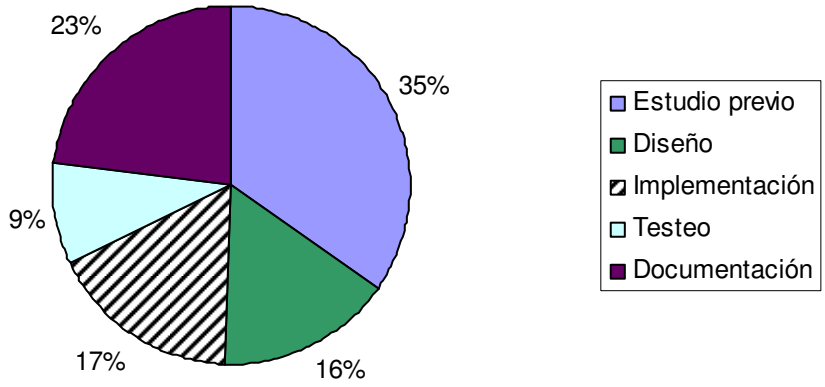



**Fig. 5.2** Desglose de horas por grupo de tareas

Realizando la suma de horas se puede observar que el total son 563 horas, que entraría dentro del tiempo de desarrollo de un TFC según los mínimos recomendados por la EPSC (450 horas). Se ha sobrepasado esta recomendación por el uso de distintas tecnologías, las cuales necesitan un periodo de investigación y estudio previo, pruebas y la posterior implementación con el sistema.

Dentro de la tarea de estudio previo, hay varias tecnologías que han llevado más tiempo de aprendizaje.

Una de ellas es el despliegue de Servicios Web, con la preparación de su entorno, Apache Axis2, y la preparación de cliente para la comunicación con los servicios. En este estudio previo de Servicios Web, el despliegue de estos se realizó a mano, por lo que requiere un tiempo de aprendizaje bastante elevado. Una vez se realizaron los primeros Servicios Web de prueba el resto fueron más rápido, puesto que se utilizó un *plugin* de Axis2 para el entorno de programación Eclipse.

La siguiente tarea que más tiempo ha llevado de aprendizaje y desarrollo es la de la creación del *parser* de fichero XML. Hubo dos pasos para el aprendizaje, utilizar Digester para pasear y llamar a métodos para almacenar y posteriormente cambiar la llamada a métodos por la creación de los objetos, que almacenan los datos, directamente.

Otra de las tareas que ha producido retrasos en el desarrollo ha sido la de la implementación de los componentes que conectan con la base de datos relacional.

## 5.3. Estimación de costes

Para realizar una estimación de los costes, hay que tener en cuenta varios factores:

- **Tiempo dedicado:** Teniendo en cuenta las fechas indicadas en el apartado anterior, 5.1 Tiempo dedicado, se ha dedicado aproximadamente un cuatrimestre para al realización de este proyecto. Es un tiempo reducido para poder llevar acabo la mayoría de las funciones descritas en el apartado 2.1 Funcionalidades.

  De todo este tiempo, se han dedicado aproximadamente 93 días.

- **Material utilizado:** En este proyecto se han utilizado dos máquinas, una de ellas era mi portátil. El sistema se puede desarrollar en una simple máquina, pero es mucho más fácil realizarlo en dos para poder separar servicios.

  Mi portátil no será incluido en el coste, pues que ha sido utilizado para muchas otras tareas y no fue adquirido expresamente para realizar el TFC.

  En cuanto a la otra máquina, se ha utilizado bajo un sistema operativo Windows XP, con lo que es necesario poseer una licencia, licencia que fue adquirida por la UPC, no se tendrá en cuenta ya que es una licencia corporativa y reutilizada en muchas más máquinas. La máquina ha sido proporcionada por i2CAT, y se ha utilizado para varios proyectos y se puede volver a reutilizar con lo que está asumido por i2CAT en la adquisición de equipos informáticos.

La electricidad y la conexión a Internet también está asumida por i2CAT por lo que sería demasiado complejo comprobar que gastos han venido por parte de la realización de este proyecto.

Para el diseño del proyecto y redactado de la memoria se ha utilizado Microsoft Office 2003 y Visio 2003, con licencia UPC al igual que el Windows y MagicDraw, con el mismo tipo de licencia que los anteriores.

Los servidores han sido montados con Apache Tomcat, para la aplicación *web* y Servicios Web con Apache AXIS2. MySQL y eXist para los repositorios. Para el desarrollo de software se ha utilizado la plataforma Eclipse. Todas estas herramientas son totalmente gratuitas.

- **Personal:** Durante la realización del TFC, si hizo bajo una beca. El precio por hora para un estudiante de Ingeniería sin titulación es aproximadamente 5,8 €/hora.

Teniendo en cuenta todos estos factores, no se puede estimar un coste específico pero es totalmente asumible ya que es parte de la formación del estudiante.

Teniendo en cuenta el valor del tiempo dedicado en el mercado laboral y los aspectos comentados:

- Horas diarias dedicadas: 6
- Días totales de realización de proyecto: 93
- Precio por hora para un Ing. Técnico: 6€/hora
    - TOTAL: 3348€

# CAPÍTULO 6. CONCLUSIONES

## 6.1. Objetivos cumplidos

El objetivo principal de este TFC era documentar, diseñar y desarrollar un sistema capaz de almacenar y gestionar información referente al usuario y a su contexto, para posteriormente poder utilizarla para poder ofrecer por ejemplo servicios a medida.

Este desarrollo ha sido completado con éxito, aunque se ha centrado implementando un servicio, el de recomendación de recursos multimedia.

Por falta de tiempo y conocimientos, el resto queda como futuras mejoras para este sistema y la implementación de este dentro del sistema I3Media.

## 6.2. Trabajo futuro

- Implementación de un sistema de permisos para poder identificar a usuarios. Útil por ejemplo para servicios de pago:
    - Usuario básico: *usuario con tarifas mínimas*
    - Usuario premium: *usuario con tarifas más elevadas y por lo tanto con más servicios*
    - Administrador: *usuario necesario para administrar el sistema.*
- Incluir un servidor de contenidos audiovisuales para no depender de un sistema externo.
- Desarrollo de nuevas funcionalidades para la aplicación *web*:
    - Listado de usuarios
    - Historial de consumo de usuario
    - Última entrada del usuario al sistema
    - Baja de usuario
- Utilizar el módulo de adaptación del sistema I3Media para la recomendación de recursos. Esto se realizaría mediante una integración mediante Servicios WS
- Desarrollo del sistema de *feedback* para las puntuaciones recomendadas.
- Un motor para realizar estadísticas de consumo y hacerlas accesibles por internet.

## 6.3. Impacto medioambiental

El impacto medioambiental es muy importante en el diseño de un proyecto en la época en la que vivimos. La viabilidad medioambiental se tiene en cuenta para la puesta en marcha de nuevos proyectos o para su rechazo.

Con este sistema se podría lograr un cambio, tanto social, como medioambiental.

Este proyecto se podría utilizar por ejemplo para intentar reducir la publicidad actual. Este cambio se basaría en conseguir una publicidad más selecta, teniendo en cuenta el contexto del usuario, y por lo tanto reducir el consumo de papel utilizado para realizar la publicidad. Esto conllevaría a reducir el consumo eléctrico, de agua y de celulosa extraída de los árboles para la fabricación de papel.

Influiría también en la sociedad. Se lograría tener un videoclub personalizado a cada usuario en casa, con lo cual las personas no perderían tiempo en ir a los videoclubes físicos. Hay que tener muy en cuenta que la mayoría de estos desplazamientos se realizan en vehículo particular, con la consiguiente reducción del consumo de combustible. Con este servicio de videoclub en casa también se ahorraría en soporte físico del recurso y por lo tanto de plásticos.

Este proyecto es también una oportunidad de negocio para empresas de servicios y consumo, donde se podría tener caracterizados a los usuarios o clientes y de esta manera poder venderles sabiendo de antemano lo que éstos quieren. Esto abre nuevos debates sobre hasta donde puede llevar la monitorización de usuario y la seguridad de los datos de éstos.

## 6.4. Conclusiones personales

La realización de este trabajo ha resultado ser muy interesante y sobre todo importante para mi formación en la carrera. El tema elegido ha sido un total acierto y la finalización de éste ha sido satisfactorio.

El tema que se ha desarrollado es de gran utilidad y puede ser aplicado a muchos contextos, por lo tanto posiblemente sea muy importante en un futuro no muy lejano. También se ha realizado un estudio a tecnologías muy interesantes y potentes.

Hay que destacar que la formación no se ha centrado solo en los conocimientos adquiridos con el estudio y utilización de estas tecnologías, si no también en desarrollar nuevas técnicas para la búsqueda de información y documentarse. La planificación y el redactado del proyecto siguiendo las pautas marcadas también juegan un papel importante en la formación.

Aparte de la formación que haya podido adquirir, con este proyecto también he aprendido otras cosas. Este último cuatrimestre lo he dedicado al Trabajo Final de Carrera y a mi beca en i2CAT. El TFC aparte de terminar de formarte en la carrera te hace vivir momentos buenos, como por ejemplo la satisfacción de conseguir un sistema en funcionamiento, pero también momentos no tan buenos, momentos de estrés o de frustración por no encontrar información necesaria o por un desarrollo incorrecto lo que acarrea problemas y errores en este. Mi suerte ha sido poder contar con compañeros de i2CAT y un tutor que siempre que han podido me han ayudado a resolver mis dudas y hacer pasar esos momentos no tan buenos.

Estos momentos son los que te enseñan nuevos valores y apreciar la ayuda de los compañeros. Te enseñan a mantener la calma en todo momento. Te ayudan a afrontar los problemas desde otro punto de vista, puesto que todo tiene solución y sobretodo, superar las adversidades que el durante el desarrollo de este TFC me han ido surgiendo.

Las fuerzas para poder superarse a uno mismo, aprender más como persona y como estudiante, realizar un trabajo bien hecho y que funcione es la motivación que he tenido durante todo este tiempo para poder llevar a cabo este TFC.

# BIBLIOGRAFÍA

## Libros consultados

[1] CHIARIGLIONE, L. 2002. "*Introduction to MPEG-7: Multimedia Content Description Interface. Introduction to MPEG-7: Multimedia Content Description Interface*".

[2] VETRO, A. 2004. "*MPEG-21 digital item adaptation: enabling universal multimedia access. Multimedia", IEEE* 11, 84-87.

[3] CHAUDHRI, A.B., RASHID, A. AND ZICARI, R. 2003. "*XML Data Management: Native XML and XML-Enabled Database Systems*". Addison-Wesley Professional.

[4] DATE, C. 2001. "*Introducción a los sistemas de bases de datos*". Pearson Educación.

## Artículos

[5] KOLAR, P. AND LOUPAL, P. "*Comparison of Native XML Databases and Experimenting with INEX*". *Desna–CaRıca*.

[6] SRIVASTAVA, A.V. "*Comparison and Benchmarking of Native XML Databases*" CS497 Report Supervisor: Prof. TV Prabhakar.

[7] TSENG, B., LIN, C.Y., SMITH, J., CENTER, I.B.M.T.J.W.R. AND HAWTHORNE, N. 2004. "*Using MPEG-7 and MPEG-21 for personalizing video*". *Multimedia, IEEE* 11, 42-52.

[8] Alberto José González Cela, Oriol Solé Molina, Guillem Cabrera Añón, "*Proyecto I3Media, Definición y Arquitectura de servicio de adaptación", 2007*

## Tutoriales

[9] IZURA, P.P.X.A. Implementación del patrón MVC en aplicaciones Web, Útilmo acceso: 15-07-08.

[10] Manual de Java, "*Creación de reglas personalizadas con Apache Commons Digester*", Útilmo acceso: 15-07-08.

[11] ChuWiki, "*Introducción a Hibernate"*, Útilmo acceso: 15-07-08.

[12] Apache Software Foundation, Web Services Apache Axis2, "*Code generador Wizard",* Útilmo acceso: 15-07-08.

[13] Apache Software Foundation, Web Services Apache Axis2, "*Service Archive Generator Wizard*", Útilmo acceso: 15-07-08.

[14] IBM, "*Develop Web Services with Axis2*", Útilmo acceso: 15-07-08.

## Enlaces Web

[15] Fundació i2CAT, URL: <http://www.i2cat.net>

[16] Proyecto I3Media, URL: <http://www.i3media.org>

[17] Página oficial del Servidor Apache Tomcat, "*Apache Jakarta Project*", URL: <http://tomcat.apache.org>

[18] Página oficial de proyectos para Servicios Web Apache,"*Apache Axis2*", URL: <http://ws.apache.org>

[19] Sun Microsystems Java API, "Java™ Platform, Standard Edition 6 API Specification", URL:<http://java.sun.com/javase/6/docs/api/>

[20] MySQL Sun Microsystems,"*MySQL Server*", URL: <http://www.mysql.com>

[21] Página oficial eXist,"*Open Source Native XML Database*", URL: <http://exist.sourceforge.net>

[22] Página oficial Apache Struts,"*Struts Framework*", URL: <http://struts.apache.org>

[23] Página oficial Hibernate, contiene Hibernate Core, Hibernate Annotations, Hibernate Tools, URL: <http://www.hibernate.org>

[24] Apache Commons, Digester es un parser Java para XML,"*Commons Digester*", URL: <http://commons.apache.org/digester>

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

[26] Página oficial Jeroen Wijering Flash Placer, "*JW Placer*", URL: < http://www.jeroenwijering.com/ >

[27] Página oficial Red5, "Open Source Flash Server", URL: < http://osflash.org/red5 >

[28] Página oficial Java de Sun Microsystems, contiene JDK, JRE, "*Source for Java developers*", URL: <http://java.sun.com/>

[29] Página oficial Eclipse Project, "*Eclipse open development platform*", URL: <http://www.eclipse.org/>

[30] W3C, XML Path Language (Xpath) v 2.0, 2007,
URL: <http://www.w3.org/TR/xpath>

[31] W3C, XML Query Language (Xquery) v 1.0, 2007,
URL: <http://www.w3.org/TR/xquery/>

[32] Daniel Pecos, "*Postgre vs MySQL"*,
URL: <http://www.netpecos.org/docs/mysql_postgres/index.html>

[33] Página oficial SAX, "*Simple API for XML",*
URL: <http://www.saxproject.org/>

[34] Prolog, Lenguaje de programación lógico e interpretado,
URL: <http://www.swi-prolog.org/>

[35] FUZZY, Lógica difusa,
URL: <http://www.geocities.com/icatercera/electronica/fuzzy.html>

[36] DOM, *Document Object Model*,
URL: <www.w3.org>

Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANEXOS

**TÍTULO DEL TFC: Arquitectura avanzada de adaptación de recursos multimedia**

**TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática**

**AUTOR: Alberto Carlos Toro Sánchez**

**DIRECTOR: Toni Oller Arcas**

**FECHA: Julio 2008**

# ANEXO 1: ACRÓNIMOS

API – Application Programming Interface
BBDD – Bases de Datos
BDD – Base de Datos
BLOB – Binary Large OBjects
CPU – Central Processing Unit
CVS – Concurrent Versioning System
DI – Digital Item
DOM – Document Object Model
EPSC – Escola Politècnica Superior de Castelldefels
FLV – Flash Video
GUI – Graphical User Interface
HTTP – HyperText Transfer Protocol
JDBC – Java Database Connectivity
JDK – Java Development Kit
JSP – JavaServer Pages
PCDATA – Parsed Character Data
POJO – Plain Old Java Object
RAM – Random Access Memory
RTMP – Real Time Messaging Protocol
SAX – Simple API for XML
SOAP – Simple Object Access Protocol
SQL – Structured Query Language
SVN – Subversion
TFC – Trabajo Final de Carrera
UML – Unified Modeling Language
URI – Uniform Resource Identifier
WSDL - Web Services Description Language
XDM – Xpath / Xquery Data Model
XML – EXtensible Markup Language
XPath – XML Path Language
XQL – Extensible Quero Language
XQuery – XML Query Language
XSLT – Extensible Stylesheet Language for Transformations
WSDL – Web Service Description Language

# ANEXO 2: GESTIÓN DE METADATOS CON EXIST

Para la gestión, en el servidor de perfiles, de los metadatos con la base de datos se utilizan dos clases Java que nos permiten tanto insertar ficheros, como recuperarlos de la base de datos.

## 2.1. Petición de documento XML

```
package i2cat.i3media.server.integration.xmldb;

import javax.xml.transform.OutputKeys;

import org.apache.log4j.Logger;
import org.xmldb.api.DatabaseManager;
import org.xmldb.api.base.Collection;
import org.xmldb.api.base.Database;
import org.xmldb.api.modules.XMLResource;

public class PeticionXML {

	static String URI = "xmldb:exist://localhost:8082/exist/xmlrpc/";

	public static String getXML(String collection, String document) throws Exception {
		String driver = "org.exist.xmldb.DatabaseImpl";

		Class cl = Class.forName(driver);
		Database database = (Database) cl.newInstance();
		DatabaseManager.registerDatabase(database);

		Collection col = DatabaseManager.getCollection(URI+"db/"+collection);
		col.setProperty(OutputKeys.INDENT, "yes");
		XMLResource res = (XMLResource)col.getResource(document);

		if(res == null) {
			Logger.getRootLogger().error("document not found!");
			return null;
		}
		else{
			return res.getContent().toString();
		}

	}
}
```

**Fig. 1** Ejemplo de petición de documento

Con este ejemplo, lo primero que se realiza es indicarle a la implementación de la base de datos (*DataBaseImpl*) cual es el *driver* que debe de utilizar para realizar la conexión. Una vez definido se realizará una petición de entrada a la colección donde está almacenado el documento que deseamos recuperar

(DatabaseManager.getCollection(URI+"db/"+collection)), donde la URI, es la dirección del servidor al que se desea preguntar:

*xmldb:[ID-BaseDatos]://[Dirección-máquina]/exist/xmlrpc/db/collection*

Una vez dentro de la colección, se le indicará el documento que se desea recibir:

*XMLResource res = (XMLResource)col.getResource(documento)*

Si todo ha ido bien, el objeto *res* contendrá el documento, simplemente se debe de extraer de la siguiente forma:

*res.getContent().toString()*

Donde el *toString* nos permite pasarlo a una cadena de caracteres *String*.

## 2.2. Almacenado de documento XML

```
import org.apache.log4j.Logger;
import org.xmldb.api.DatabaseManager;
import org.xmldb.api.base.Collection;
import org.xmldb.api.base.Database;
import org.xmldb.api.modules.XMLResource;


public class AlmacenaXML {

      static String URI = "xmldb:exist://192.168.48.200:8082/exist/xmlrpc/";


      public void setXML(String collection, String data, String xmlName) throws Exception{

        String driver = "org.exist.xmldb.DatabaseImpl";
        Class cl = Class.forName(driver);
        Database database = (Database)cl.newInstance();
        DatabaseManager.registerDatabase(database);

        Collection col =  DatabaseManager.getCollection(URI +"db/"+ collection);

        XMLResource document = (XMLResource)col.createResource(xmlName, "XMLResource");

        document.setContent(data);

       Logger.getLogger(this.getClass()).info("Almacenando documentos " + document.getId()
+ "...");

        col.storeResource(document);
        Logger.getLogger(this.getClass()).info("Almacenado.");

      }
```

**Fig. 2** Ejemplo de subida de documento a la base de datos

El proceso seguido para esta ocasión es parecido al anterior, diferenciándose únicamente en que cuando estamos en la colección, instanciamos un objeto *XMLResource*.

A este objeto, que será el que cree el documento en la colección se le indica el nombre (*xmlName*) que tendrá ese documento:

*XMLResource document = (XMLResource)col.createResource(xmlName, "XMLResource")*

Seguidamente se introducirá el contenido del objeto *data* el cual contiene los caracteres del documento:

*document.setContent(data)*

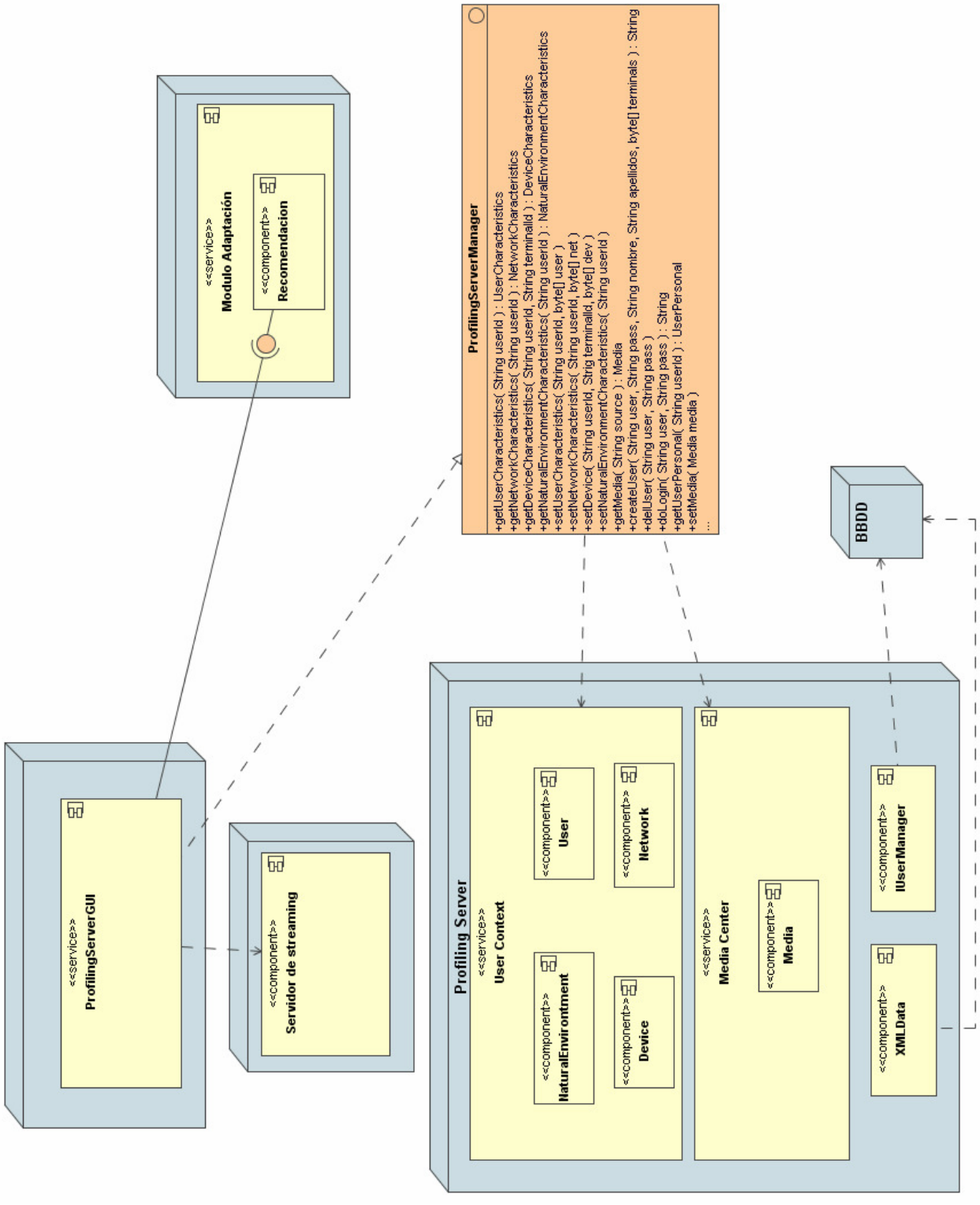
Preparado el objeto *document*, realizaremos una llamada al método *storeResource* que nos permite, pasando por parámetros el objeto *document*, almacenar el documento que deseamos en la colección en la que se encuentra conectado:

*col.storeResource(document)*

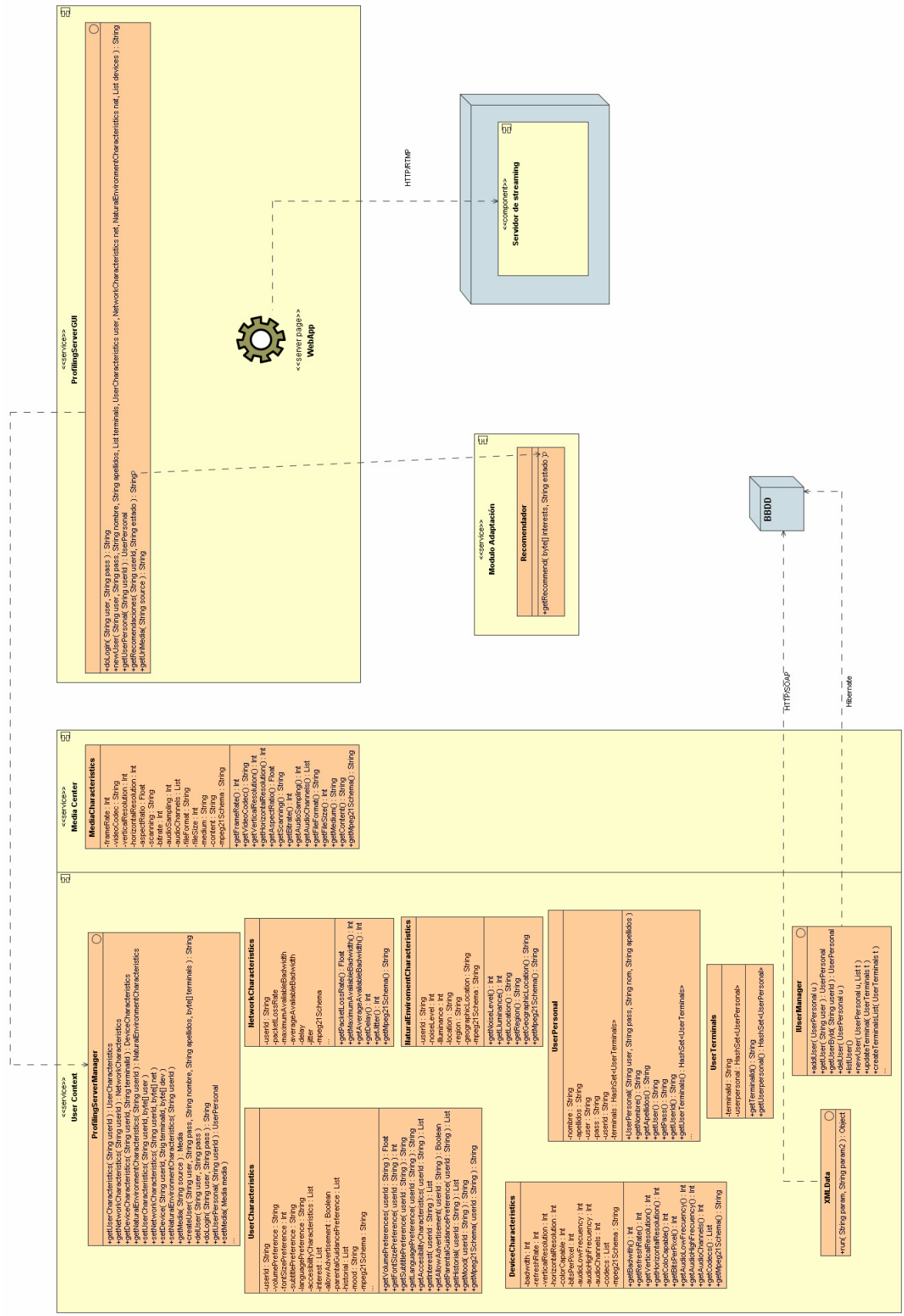
Con este último comando se enviará a eXist este documento, el cual será almacenado e indexado

# ANEXO 3: DIAGRAMAS DE LA ARQUITECTURA DEL SISTEMA

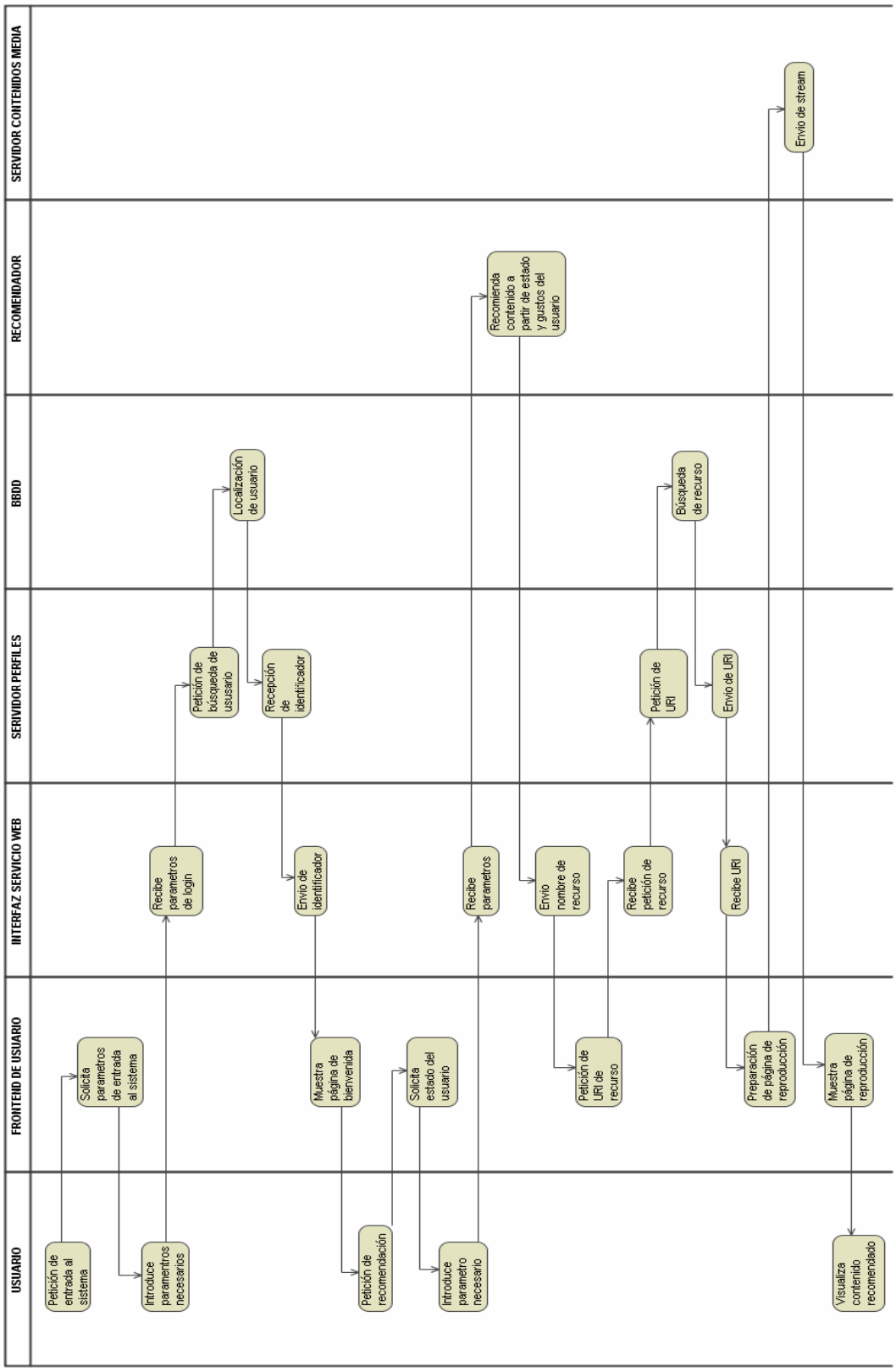
## 3.1. Componentes módulo de información nivel alto



## 3.2. Componentes módulo de información nivel bajo



## 3.3. Diagrama de secuencia entre módulos



# ANEXO 4: COMPARATIVA ENTRE EXIST Y XINDICE

La siguiente comparación ha sido extraída del documento "*Comparison of Native XML Databases and Experimenting with INEX*" [5].

Para realizar la comparación de *benchmarking* (prueba de rendimiento), en este articulo utilizan *INitiative for the Evaluation of XML* (INEX). La versión utilizada de esta herramienta es la 2003 (1.4), y está compuesta por a colección de 536MB en texto con formato XML. Exactamente 12.107 artículos provenientes del IEEE. Cada XML está compuesto en media por 1532 nodos o *tags*.

Todos estos artículos se almacenan en el directorio *root* (raíz) de las bases de datos y ordenando los XMLs por años. El tiempo que necesitaron cada base de datos para almacenar todos los datos fueron de 25 min para Xindice y 97 min para eXist. Los datos almacenados necesitaron 600MB y 1300MB de espacio en disco para ser almacenado, respectivamente.

Para la realización de la prueba se utiliza un ordenador con un procesador Celaron 1.7 Ghz, 51MB RAM y Windows XP (SP2).

Los resultados obtenidos fueron que para la mayoría de las pruebas de búsqueda realizadas con *Xpath*, Xindice devolvió resultados vacíos mientras que eXist mostró un mejor comportamiento.

En la siguiente tabla se muestran los resultados obtenidos de las distintas pruebas que se realizaron y de sus distintos reintentos en las peticiones de búsqueda por *Xpath*. Los tiempos obtenidos por las dos bases de datos se toman en segundos.

| No. | Query | Records retrieved | Query duration time [s] eXist | Xindice |
|---|---|---|---|---|
| 1 | /article | 12104 | 1,3 | 230 |
| 2 | /article/fm/hdr/hdr1/crt/issn | 11666 | 2,2 | 98 |
| 3 | //issn | 11666 | 1,3 | 447 |
| 4 | /article/bdy/sec[1] | 11955 | 1,9 | NA |
| 5 | /article/bdy/sec[last()] | 11955 | 5,6 | NA |
| 6 | /article/bdy/sec[last() − 1] | 11019 | 5,8 | NA |
| 7 | /article/bdy/sec[position() < 3] | 22974 | 8,1 | NA |
| 8 | //sec[@type] | 868 | 1,0 | more than 10 min |
| 9 | //sec/p/ref[@type =′ bib′] | 108496 | 81,3 | NA |
| 10 | /article/fm/hdr/hdr2/pdt[yr =′ 1995′] | 1623 | 2,6 | NA |
| 11 | /article/fm/hdr/hdr2/pdt[yr =′ 1995′andmo =′ Spring′] | 72 | 4,0 | NA |
| 12 | /article/∗ | 58472 | 164,3 | NA |
| 13 | /∗/∗[@∗] | 49 | 352,0 | NA |
| 14 | //fig[@∗] | 52857 | 70,6 | NA |
| 15 | //article/fm/hdr\| //article/bdy/sec | 77487 | 8,6 | NA |
| 16 | //article/fm/hdr/hdr1\| //article/fm/hdr/hdr2 | 24208 | 3,8 | NA |

**Fig. 1** Tabla de resultados

Estos resultados nos muestra que, aunque eXist necesita más del doble de espacio en disco que Xindice, sus resultados en búsqueda son muy superiores debido a la estructura de indexación muy eficiente utilizada por eXist.

A parte de los resultados, eXist ofrece una interfaz de usuario fácil de utilizar, lo cual deja a eXist como mejor base de datos.

# ANEXO 5: PLANTILLAS XML

En este anexo se encuentran las plantillas utilizadas para los metadatos del servidor.

## 5.1. User Characteristics

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA
 xmlns="urn:mpeg:mpeg21:dia:schema:2003"
 xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema"
 xsi:schemaLocation="urn:mpeg:mpeg21:dia:schema:2003">

 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironment xsi:type="UserCharacteristicsType">
   <UserCharacteristics xsi:type="PresentationPreferencesType">
    <Audio>
            <VolumeControl>0.85</VolumeControl>
    </Audio>
    <FocusOfAttention>
     <TextFocusOfAttention>
       <Font fontColor="black" fontSize="10" fontType="arial"/>
     </TextFocusOfAttention>
    </FocusOfAttention>
   </UserCharacteristics>
   <UserCharacteristics xsi:type="AccessibilityCharacteristicsType">
    <Visual>
     <LowVisionSymptoms>
      <LackOfContrast>
             <TextualDegree>Severe</TextualDegree>
            </LackOfContrast>
     </LowVisionSymptoms>
    </Visual>
    <Audio>
            <RightEar>
             <Freq250Hz>0.0</Freq250Hz>
             <Freq500Hz>5.5</Freq500Hz>
             <Freq1000Hz>-0.2</Freq1000Hz>
             <Freq2000Hz>-2.0</Freq2000Hz>
             <Freq4000Hz>1.5</Freq4000Hz>
             <Freq8000Hz>5.5</Freq8000Hz>
            </RightEar>
            <LeftEar>
             <Freq250Hz>9.0</Freq250Hz>
             <Freq500Hz>-1.5</Freq500Hz>
             <Freq1000Hz>9.0</Freq1000Hz>
             <Freq2000Hz>9.0</Freq2000Hz>
             <Freq4000Hz>9.0</Freq4000Hz>
             <Freq8000Hz>10.0</Freq8000Hz>
            </LeftEar>
     </Audio>
    </UserCharacteristics>
    <UserCharacteristics xsi:type="ContentPreferencesType">
            <UserPreferences>
             <mpeg7:FilteringAndSearchPreferences>
              <mpeg7:ClassificationPreferences>
               <mpeg7:LanguageFormat>subTitles</mpeg7:LanguageFormat>
               <mpeg7:CaptionLanguage>en</mpeg7:CaptionLanguage>
               <mpeg7:Language>en</mpeg7:Language>
               <mpeg7:Genre>
                       <mpeg7:Name>Sports</mpeg7:Name>
               </mpeg7:Genre>
               <mpeg7:Genre>
                       <mpeg7:Name>Entertainment</mpeg7:Name>
               </mpeg7:Genre>
               <mpeg7:Genre>
                       <mpeg7:Name>Movies</mpeg7:Name>
```

```
            </mpeg7:Genre>
                    <mpeg7:ParentalGuidance>
                      <ParentalRating href="urn:mpeg:mpeg7:cs:MPAAParentalRatingCS:2001:PG-13">
                       <Name>PG-13</Name>
                      </ParentalRating>
                      <Region>us</Region>
                    </mpeg7:ParentalGuidance>
                    <mpeg7:Form>noAdvertisements</mpeg7:Form>
                    </mpeg7:ClassificationPreferences>
                     </mpeg7:FilteringAndSearchPreferences>
          </UserPreferences>
          <UsageHistory>
           <mpeg7:UserActionHistory>
            <mpeg7:ObservationPeriod>
                    <mpeg7:TimePoint>2000-10-09T18:00-08:00</mpeg7:TimePoint>
                    <mpeg7:Duration>PT6H</mpeg7:Duration>
            </mpeg7:ObservationPeriod>
            <mpeg7:UserActionList>
             <mpeg7:ActionType>
                    <mpeg7:Name>PlayStream</mpeg7:Name>
             </mpeg7:ActionType>
             <mpeg7:UserAction>
             <mpeg7:ProgramIdentifier>
                     urn:mymedia:av:02-mnf-109
             </mpeg7:ProgramIdentifier>
             </mpeg7:UserAction>
             <mpeg7:UserAction>
                    <mpeg7:ProgramIdentifier>
                urn:mymedia:av:14-znn-623
                    </mpeg7:ProgramIdentifier>
             </mpeg7:UserAction>
             <mpeg7:UserAction>
                    <mpeg7:ProgramIdentifier>
                      urn:mymedia:av:73-mov-814
                    </mpeg7:ProgramIdentifier>
             </mpeg7:UserAction>
            </mpeg7:UserActionList>
           </mpeg7:UserActionHistory>
          </UsageHistory>
    </UserCharacteristics>
   </UsageEnvironment>
  </Description>
</DIA>
```

## 5.2. Network Characteristics

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA
 xmlns="urn:mpeg:mpeg21:dia:schema:2003"
 xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema"
 xsi:schemaLocation="urn:mpeg:mpeg21:dia:schema:2003">

 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironment xsi:type="NetworkCharacteristicsType">
            <NetworkCharacteristics xsi:type="NetworkConditionType">
              <AvailableBandwidth maximum="256000" average="80000"/>
              <Delay packetTwoWay="330" packetOneWay="150" delayVariation="66"/>
              <Error packetLossRate="0.05"/>
            </NetworkCharacteristics>
  </UsageEnvironment>
 </Description>
</DIA>
```

## 5.3. Natural Environment Characteristics

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA
 xmlns="urn:mpeg:mpeg21:dia:schema:2003"
 xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema"
 xsi:schemaLocation="urn:mpeg:mpeg21:dia:schema:2003">

 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironment xsi:type="NaturalEnvironmentCharacteristicsType">
   <NaturalEnvironmentCharacteristics xsi:type="AudioEnvironmentType">
            <NoiseLevel>20</NoiseLevel>
   </NaturalEnvironmentCharacteristics>
   <NaturalEnvironmentCharacteristics xsi:type="IlluminationCharacteristicsType">
            <Illuminance>500</Illuminance>
   </NaturalEnvironmentCharacteristics>
   <NaturalEnvironmentCharacteristics xsi:type="LocationType">
             <Location>

                       <mpeg7:GeographicPosition>
                        <mpeg7:Point longitude="135.75" latitude="35.00" altitude="10.00"/>
                       </mpeg7:GeographicPosition>
                       <mpeg7:Region>jp</mpeg7:Region>
                       <mpeg7:Name>kitchen</mpeg7:Name>
             </Location>
   </NaturalEnvironmentCharacteristics>
  </UsageEnvironment>
 </Description>
</DIA>
```

## 5.4. Terminal Capabilities Characteristics

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA
 xmlns="urn:mpeg:mpeg21:dia:schema:2003"
 xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema"
 xsi:schemaLocation="urn:mpeg:mpeg21:dia:schema:2003">

 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironment xsi:type="TerminalCapabilitiesType">
            <TerminalCapabilities xsi:type="CodecCapabilitiesType">
             <Decoding xsi:type="VideoCapabilitiesType">
              <Format
               href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
```

```
            <mpeg7:Name xml:lang="en">
                MPEG-4 Visual Simple Profile @ Level 1
            </mpeg7:Name>
           </Format>
          </Decoding>
          <Decoding xsi:type="AudioCapabilitiesType">
           <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
                <mpeg7:Name xml:lang="en">MP3</mpeg7:Name>
           </Format>
          </Decoding>
         </TerminalCapabilities>
         <TerminalCapabilities xsi:type="InputOutputCapabilitiesType">
          <Display bitsPerPixel="8" refreshRate="25" colorCapable="true">
                <Resolution horizontal="176" vertical="144"/>
          </Display>
          <AudioOut lowFrequency="30" highFrequency="8000" numChannels="2"/>
         </TerminalCapabilities>
   </UsageEnvironment>
  </Description>
</DIA>
```

## 5.5. Media Description

```
<?xml version="1.0" encoding="UTF-8"?>

<Mpeg7 xmlns:xsi="http://www.w3.org/2001/XMLSchema"
 xmlns="urn:mpeg:mpeg7:schema:2001"
 xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
 xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001">

 <Description xsi:type="ContentEntityType">
  <MultimediaContent xsi:type="VideoType">
   <Video>
          <MediaInformation id="news1_media">
           <MediaProfile>
            <MediaFormat>
             <Content href="MPEG7ContentCS">
                <Name>audiovisual</Name>
             </Content>
             <Medium          href="urn:mpeg:mpeg7:cs:MediumCS:2001:1.1">
                <Name xml:lang="en">CD</Name>
             </Medium>
             <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">
                <Name xml:lang="en">mpeg</Name>
             </FileFormat>
             <Bitrate variable="true" minimum="233000" maximum="512000" average="453000">512000</Bitrate>
             <FileSize>666478608</FileSize>
             <VisualCoding>
                <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1" colorDomain="color">
                  <Name xml:lang="en">MPEG-1 Video</Name>
                </Format>
                <Pixel bitsPer="8" />
                <Frame height="288" width="352" rate="25" aspectRatio="1.333" structure="interlaced"/>
             </VisualCoding>
             <AudioCoding>
                <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3">
                 <Name xml:lang="en">MPEG-1 Audio</Name>
                </Format>
                <Sample bitsPer="8" rate="40"/>
                <AudioChannels front="3" rear="1" side="2">6</AudioChannels>
             </AudioCoding>
            </MediaFormat>
     </MediaProfile>
    </MediaInformation>
   </Video>
  </MultimediaContent>
 </Description>
</Mpeg7>
```

# ANEXO 6: CREACIÓ DE SERVICIOS WEB CON AXIS2

Con este anexo se explicarán los pasos necesarios para poder crear un Servicio Web. Antes de continuar repasaremos las herramientas necesarias para poder continuar.

- **Herramientas**

  - Máquina virtual de Java: Necesaria para poder ejecutar el resto de herramientas:
    http://www.java.com/es/download/

  - Apache Tomcat: Servidor de aplicaciones *web* que se puede descargar desde:
    http://tomcat.apache.org/download-60.cgi

  - Apache Axis2: Kit de desarrollo de Servicios Web, nos permite la ejecución de estos. *Link* de descarga:
    http://ftp.udc.es/apache-dist/ws/axis2/1_4/axis2-1.4-war.zip

  - Eclipse Project: Plataforma de desarrollo necesaria para realizar el código fuente y para la generación de los servicios mediante los *plugins* de Axis2 para Eclipse:
    http://www.eclipse.org/downloads/

    - Apache Axis2 Service Archive Generator: *Plugin* para la generación del Servicio Web:
      http://www.apache.org/dyn/mirrors/mirrors.cgi/ws/axis2/tools/1_4/axis2-eclipse-service-archiver-wizard-1.4.zip

    - Apache Axis2 Code Generator : *Plugin* para la generación del WSDL y los clientes del Servicio Web:
      http://www.apache.org/dyn/mirrors/mirrors.cgi/ws/axis2/tools/1_4/axis2-eclipse-codegen-wizard-1.4.zip

Descargadas todas las herramientas necesarias procederemos a instalarlas. Lo primero es instalar la máquina virtual de Java si no lo hemos hecho ya. Si ya está instalada, pasamos a instalar Tomcat y Eclipse.

Una vez tenemos el Tomcat y el Eclipse funcionando, pasamos al *Kit* Axis2. Este simplemente habrá que descomprimir el **.war* y copiarlo en la carpeta *webapps* del directorio raíz del Tomcat, y arrancaremos el servidor para finalizar la instalación.

Los dos *plugins* de Axis2 para el Eclipse, habrá que descomprimirlos y copiarlos en la carpeta *plugins* del directorio raíz del Eclipse.

## 6.1. Generar WSDL a partir de código Java

1. Para la creación del WSDL escogeremos la interfaz que proporciona las funcionalidades. Sobre el proyecto que queremos utilizar, hacemos botón secundario, *New, Other,* y seleccionamos *Axis Wizards*, *Axis2 Code Generator*.

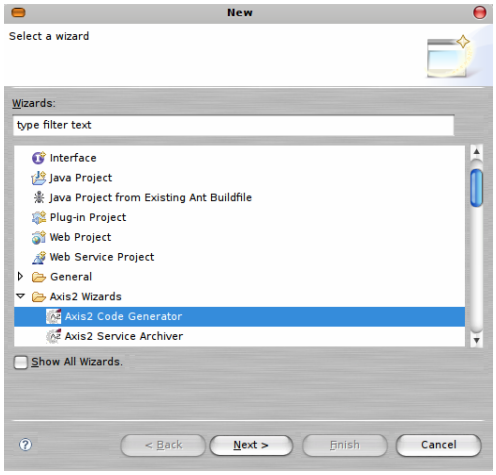



**Fig. 0.1** Selección de *plugin*

2. Seguidamente seleccionaremos la opción de creación de WSDL.

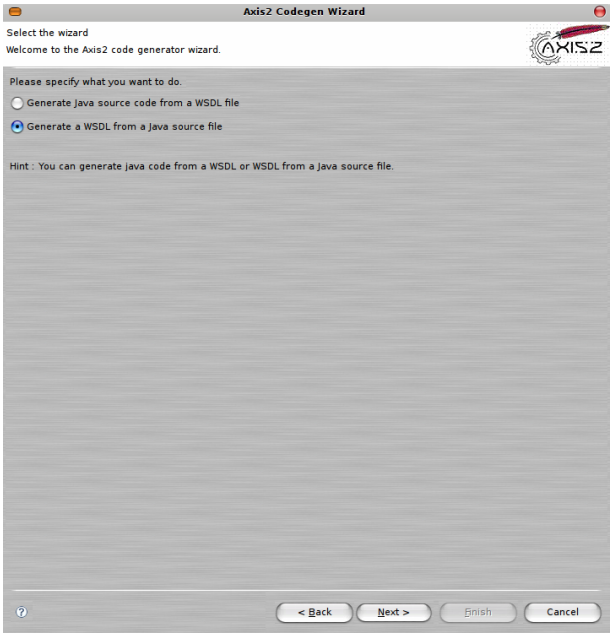



**Fig. 0.2** Elegir tipo de generación

3. Seleccionamos la clase que se leerá indicando su ruta, "*Add folder*".

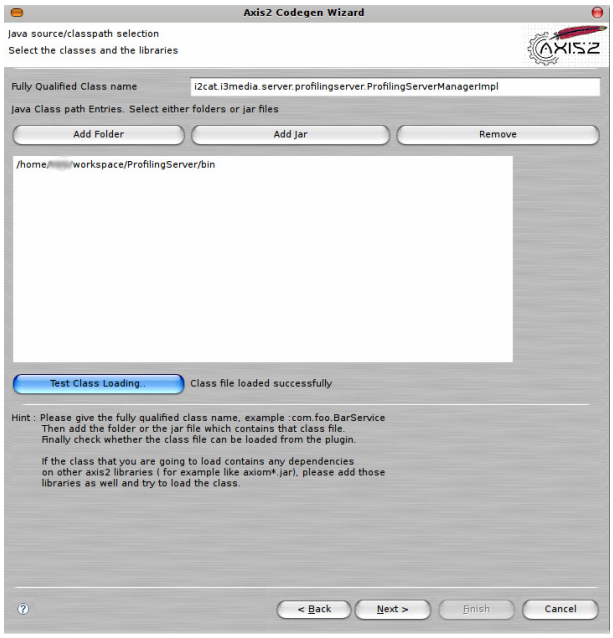



**Fig. 0.3** Selección de interfaz

4. A continuación podremos cambiar el nombre del Servicio Web, si lo preferimos.

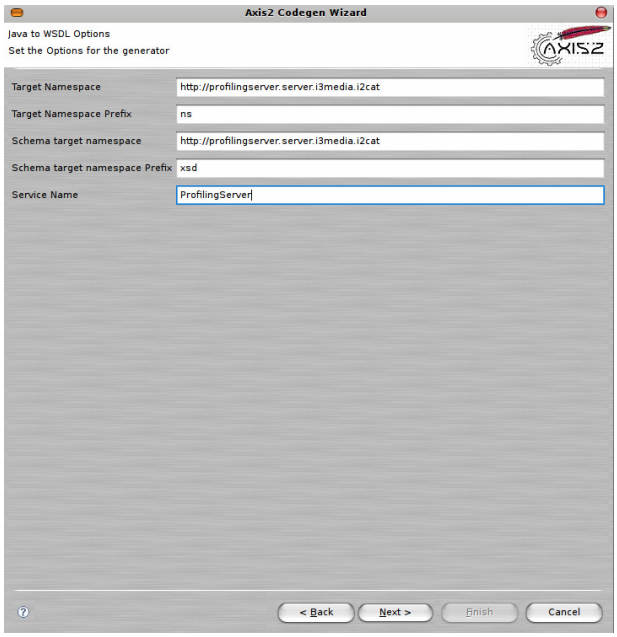



**Fig. 0.4** Nombre para el Servicio Web

5. Seleccionamos donde guardar y nombre del fichero WSDL, después guardamos.

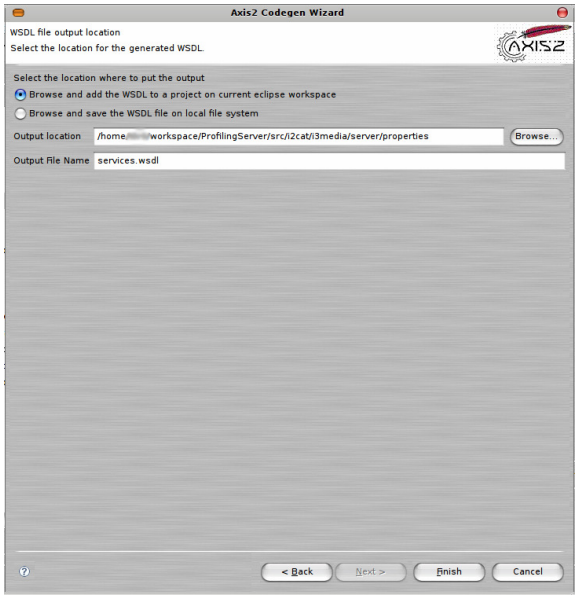



**Fig. 0.5** Último paso antes de finalizar

## 6.2. Generar Servicio Web

1. Para montar el servicio, hacemos sobre el proyecto que queremos utilizar, hacemos botón secundario, *New, Other,* y seleccionamos *Axis Wizards*, *Axis2 Service Archive Generator*.

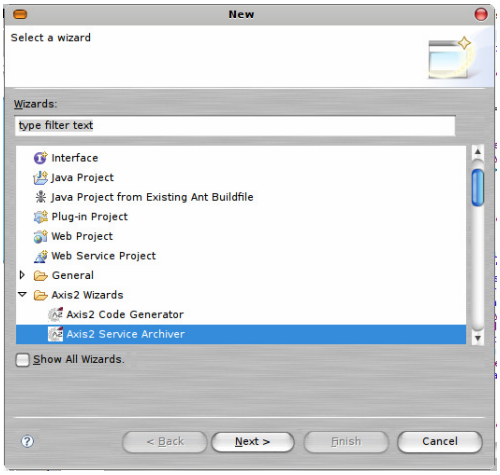



**Fig. 0.6** Selección de *plugin*

*2.* El siguiente paso es indicar el *path* de lo binarios del servicio. No marcar la casilla "*Include .class files only"* si el servicio necesita ficheros de otro tipo

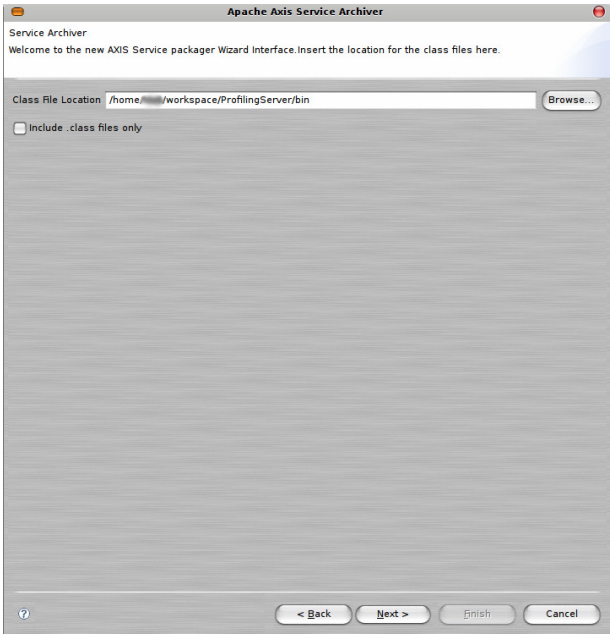



**Fig. 0.7** Selección de *path* de los binarios

3. Seguidamente seleccionaremos el WSDL que se ha creado en el apartado anterior [Apartado 6.1]

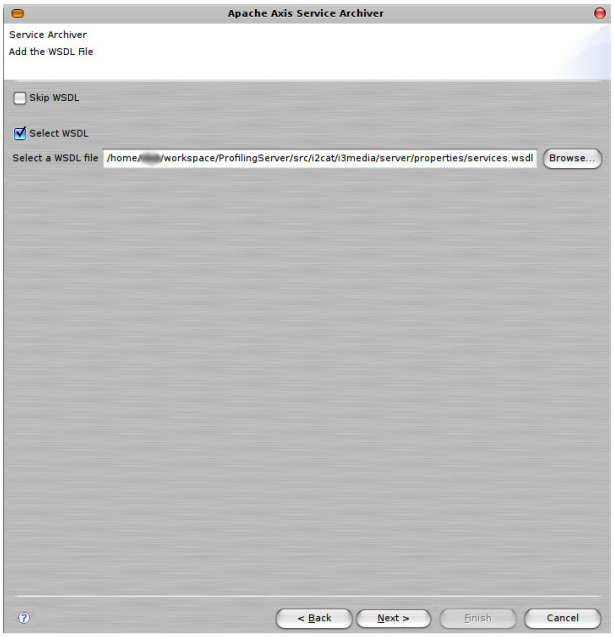



**Fig. 0.8** Selección WSDL

4. En este punto, seleccionaremos todas las librerías que hicieran falta para el funcionamiento del Servicio Web.

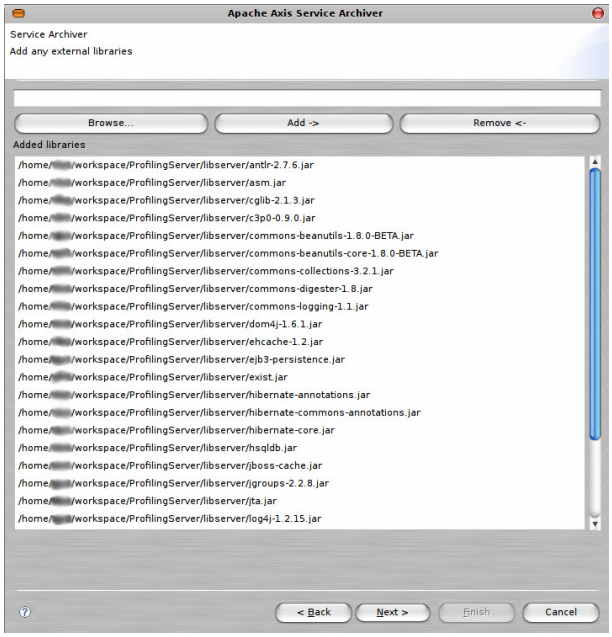



**Fig. 0.9** Lista de *JARs* seleccionados

5. Dejamos marcado que genere el fichero XML a no ser que se quiera especificar uno.

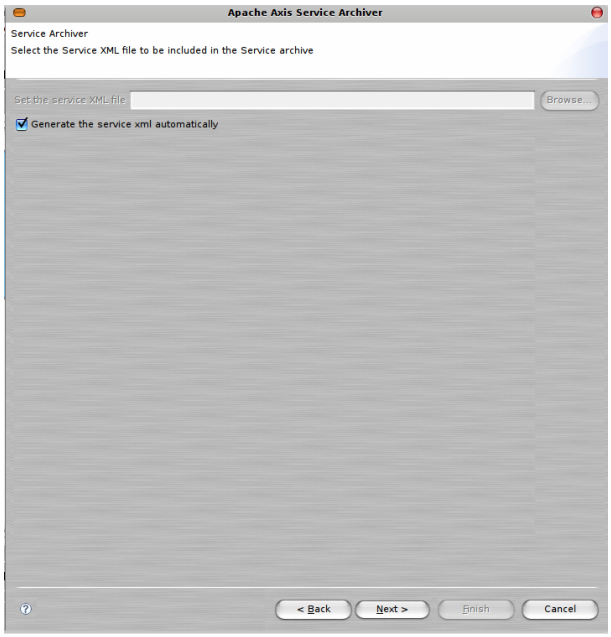



**Fig. 0.10** Generación *services.xml*

6. En el siguiente paso podremos indicar que métodos se desean incluir en el Servicio Web. Para ello hemos de seleccionar la interfaz, que debe de ser la misma que se utilizó para generer el WSDL [Apartado 6.1] .

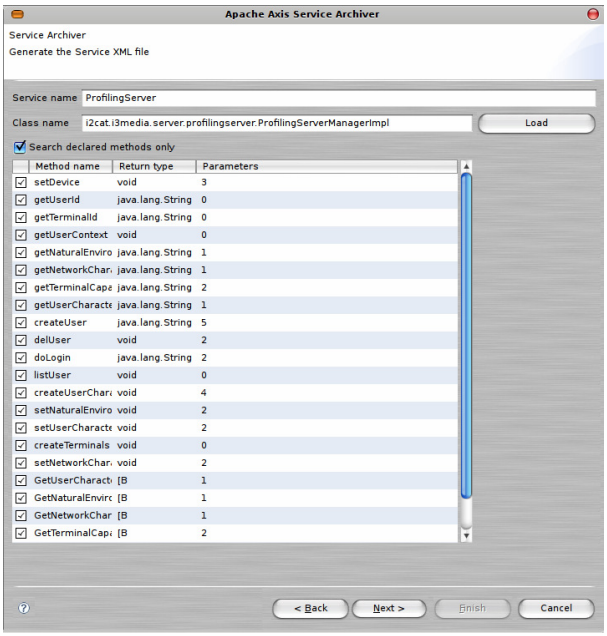



**Fig. 0.11** Lista de métodos de la interfaz

7. Por último indicaremos donde queremos guardar el Servicio Web y el nombre del futuro fichero. Para desplegarlo en el Axi2 del Tomcat, guardarlo en TOMCAT_DIR/webapps/axis2/WEB-INF/services, donde TOMCAT_DIR es el directorio raíz de éste.

## 6.2. Generar código fuente del cliente

1. Para la creación del cliente del Servicio Web, sobre el proyecto que queremos utilizar, hacemos botón secundario, *New, Other,* y seleccionamos *Axis Wizards*, *Axis2 Code Generator*.

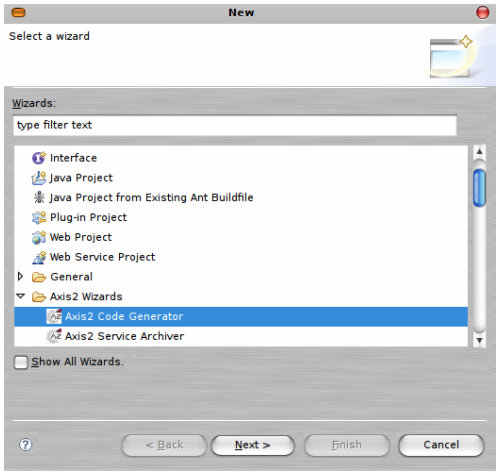



**Fig. 0.12** Selección de *plugin*

2. Ahora seleccionamos la opción de crear código Java a partir del WSDL.

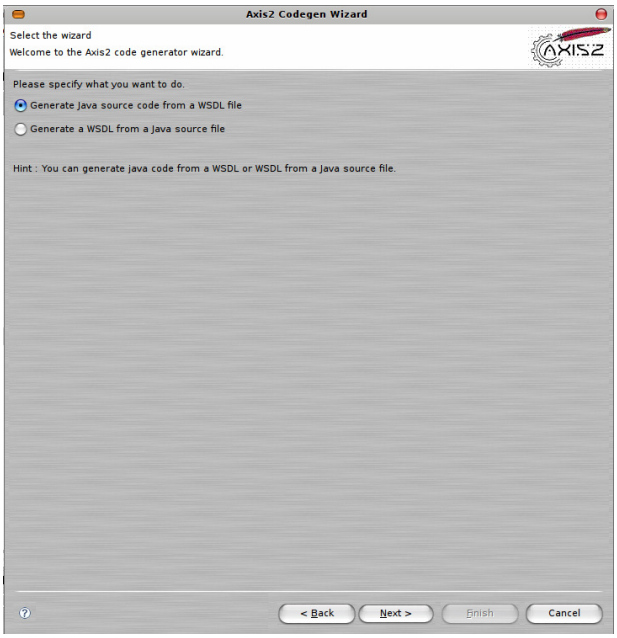



**Fig. 0.13** Elegir tipo de generación

3. Ahora seleccionamos el WSDL que hemos generado en el primer apartado [Apartado 6.1].

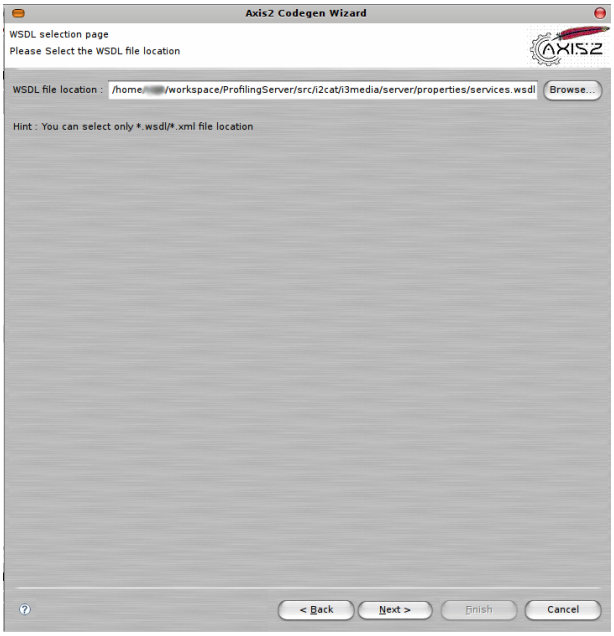



**Fig. 0.14** Selección del WSDL

4. En la siguiente pantalla en "*Codegen Option*" seleccionamos "*custom*" si deseamos cambiar las preferencias del cliente. En "*Custom package name*" seleccionamos, si se desea, el *path* destino del código fuente generado.

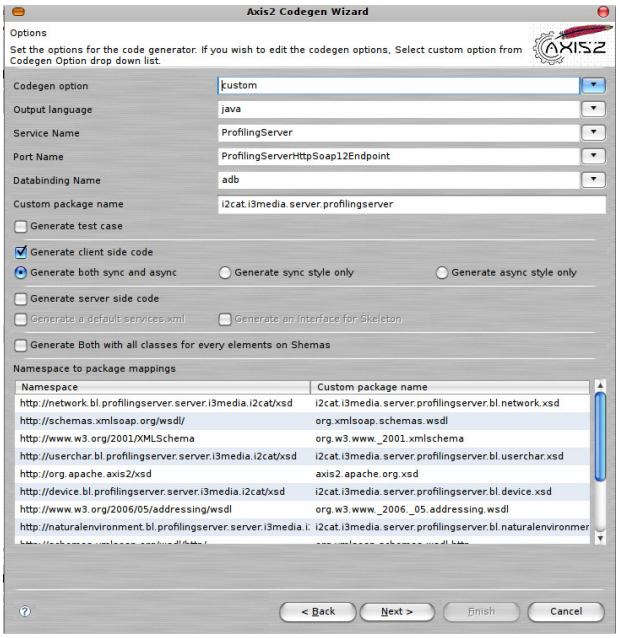



**Fig. 0.15** Configuración del cliente

5. Por último le indicamos al *path* de destino del código indicando que éste se encuentra en el directorio *workspace* del *Eclipse*. Seleccionar "*Add the Axis2 codegen jar…"* si queremos que el código prescinda de los *imports*, llamando a las clases por su *path*; ejemplo: *com.apache.common. …*

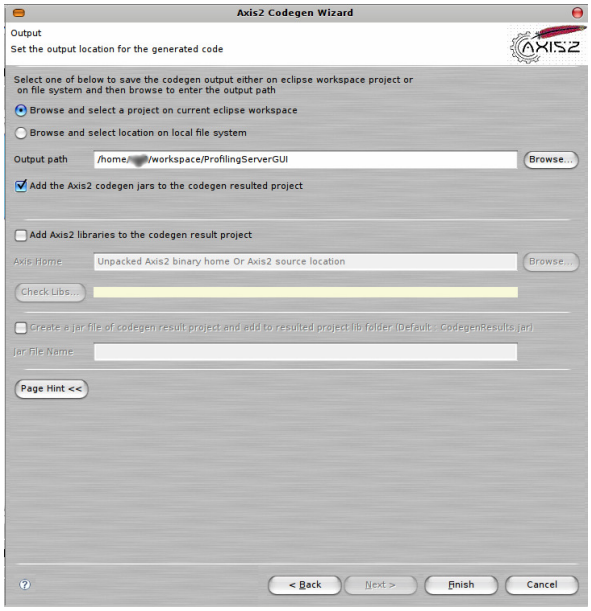



**Fig. 0.16** Último paso

6. Una vez tenemos el cliente del Servicio Web, el siguiente paso es crear una clase que interactué con éste. En el siguiente ejemplo se muestra una petición de *login*.

```
public String doLogin(String user, String pass) {

            try {
                  ProfilingServerStub stub = new ProfilingServerStub(uriWS);
                  ProfilingServerStub.DoLogin login = new ProfilingServerStub.DoLogin();
                  login.setUser(user);
                  login.setPass(pass);

                  ProfilingServerStub.DoLoginResponse loginRes = stub.doLogin(login);

                  return loginRes.get_return();
            } catch (AxisFault e) {
                  log.fatal(e.getCause(), e);
            } catch (Exception e) {
                  log.fatal(e.getCause(), e);
            }
            return null;
      }
```

**Fig. 0.17** Ejemplo de interacción con el cliente del Servicio Web

Siguiendo este ejemplo, logramos llamar a los métodos que están en el servidor como si los tuviésemos en local.

# ANEXO 7: DOCUMENTACIÓN BBDD

Anexo que hace referencia a lo explicado en el capítulo 4, apartado 4.3.

## 7.1. Gestión de metadatos

Los requerimientos de almacenamiento de metadatos pueden ser divididos en dos categorías generales:

- Centrado en los datos

  Método usado cuando el documento o metadato XML tiene bien definida la estructura y contiene datos actualizables. Son documentos predecibles en tamaño el cual suele ser limitado.

- Centrado en el documento

  Al contrario que los centrados en los datos, la estructura del documento no está bien definida, lo que conlleva que el tamaño del documento sea impredecible. Contiene datos que no serán actualizados.

Para el almacenamiento de los datos XML podemos encontrar varias estrategias: almacenarlos nativamente o mapear los datos del XML.

Las estrategias existentes pueden ser descompuestas en tres métodos básicos:

- Almacenar el documento completo en forma de texto, como un gran objeto binario (BLOB) en una base de datos relacional. Buena estrategia si el documento almacenado sólo será modificado cuando se reemplace el documento entero. Es una estrategia fácil ya que no hace falta mapear ni traducir el documento, pero limita también la búsqueda, indexamiento y granulidad de la obtención del documento XML.

- Almacenar una versión modificada del documento completo en el sistema de archivos. La más eficiente si el numero de documentos es pequeño y los documentos XML no son actualizados (o muy poco). Esto implica limitaciones en escalabilidad, flexibilidad en almacenamiento y recuperación, y seguridad ya que los datos se almacenan fuera de la base de datos (BDD).

- Mapear la estructura de los datos en el documento en la base de datos. Si la estructura del documento XML no es compatible con la estructura de la BDD, el documento debe ser transformado para poder ajustarlo para poder almacenarlo.

La última estrategia, es la más popular en las BBDD y la más utilizada. En el apartado siguiente se comentarán las distintas diferencias entre estrategias de almacenado de datos.

## 7.2. Modelos de BBDD

En este proyecto se utilizarán dos tipos de bases de datos [4]. Dependiendo de los datos que queramos gestionar se utilizará una base de datos u otra.

### 7.2.1. Modelo relacional

El modelo relacional de base de datos es el más utilizado actualmente ya que se ajusta a problemas reales y administra datos dinámicamente.

La idea principal de este modelo es el uso de relaciones entre tablas. Se pueden considerar de forma lógica como conjuntos de datos llamados *tuplas*[1], aunque a nivel físico las tablas pueden tener una estructura completamente distinta. Un punto fuerte del modelo relacional es la sencillez de su estructura lógica. Pero detrás de esa simple estructura hay un fundamento teórico importante del que carecen los BBDD de la primera generación, lo que constituye otro punto a su favor.

Para poder acceder a las BBDD y hacer consultas se utilizan dos herramientas matemáticas: el *álgebra relacional* y el *cálculo relacional*. Ambos utilizan operadores lógicos para la manipulación de los datos. El lenguaje *SQL, lenguaje declarativo*, implementado en los principales sistemas de gestión de las bases de datos, es el encargado de hacer la manipulación de los datos más transparente de cara al usuario.

- **Tipos de relaciones**

  - *Relaciones base:* Son relaciones reales que tienen nombre y forman parte directa de la base de datos almacenada.

  - *Vistas:* Son relaciones con nombre y derivadas, se representan mediante su definición en términos de otras relaciones con nombre. No poseen datos almacenados propios.

  - *Instantáneas:* La diferencia que poseen con las *vistas* es que éstas son reales están representadas no sólo por su definición en términos de otras relaciones con nombre, sino también por sus propios datos almacenados. Relaciones de solo lectura refrescadas periódicamente.

[1] *Secuencia ordenada de objetos*

- *Relaciones de consultas:* Son las relaciones resultantes de alguna consulta especificada. Pueden o no tener nombre y no persisten en la base de datos.

- *Relaciones de intermedios:* Son las relaciones que contienen los resultados de las subconsultas. Normalmente no tienen nombre y tampoco persisten en la base de datos.

- *Relaciones temporales:* Son relaciones con nombre, similares a la relaciones base o a la instantáneas, pero la diferencia es que se destruyen automáticamente cuando toca.

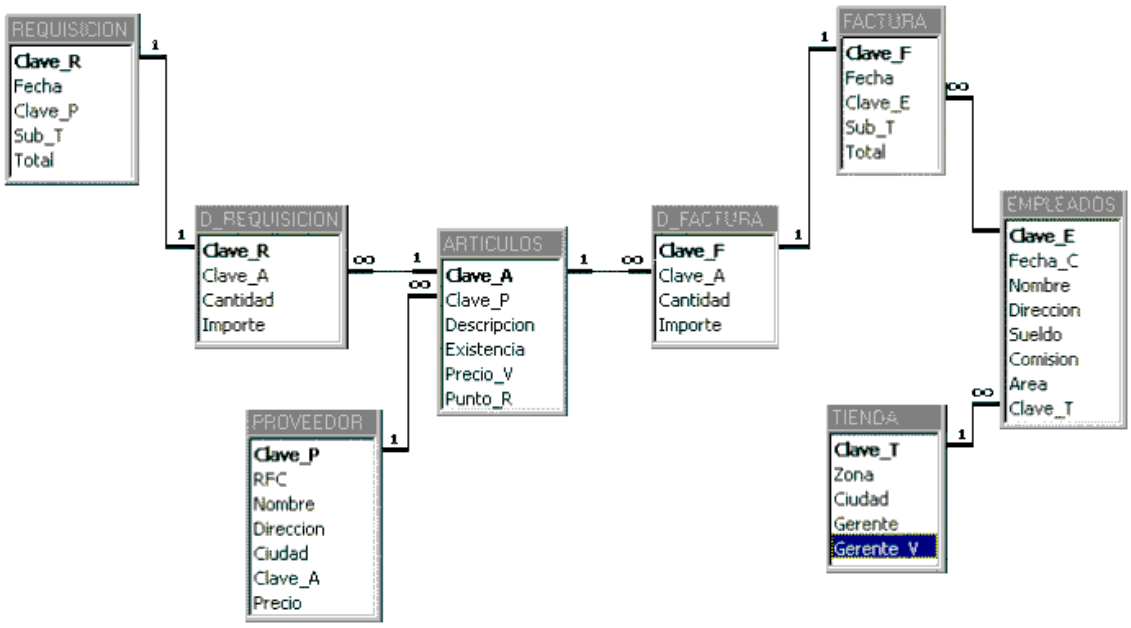



**Fig. 0.18** Ejemplo de BBDD relacional

## ■ Propiedades de las relaciones

- Cada relación tiene un nombre y éste es distinto del nombre de todas las demás.

- Los valores de los atributos son atómicos: en cada *tupla,* cada atributo toma un solo valor.

- No hay dos atributos que se llamen igual.

- El orden de los atributos no importa: los atributos no están ordenados.

- Cada *tupla* es distinta de las demás, no hay duplicaciones.

- El orden de las *tuplas* no importa, éstas no están ordenadas.

### 7.2.2. Modelo nativo XML

Una base de datos XML nativa [3] define un modelo lógico para un documento XML y almacena y recupera documentos de acuerdo a este modelo. Como mínimo el modelo debe incluir elementos, atributos, Parsed Character Data (PCDATA) y el orden del documento.

Estas bases de datos son diseñadas específicamente para almacenar documentos XML. Como el resto de bases de datos, se ofrecen características como transacciones, seguridad, acceso multiusuario, etc. La gran diferencia con el resto de bases de datos, y de ahí ganan el termino de *nativas,* es que almacenan los datos estructurados como XML sin necesidad de traducirlos a una estructura relacional o de objeto; por lo tanto, se preserva el orden del documento, secciones CDATA, entidad de uso, etc.

Las BBDD nativas también pueden ser utilizadas para integrar datos, aparte de poder controlar cambios en el esquema mucho más fácilmente en comparación con las BBDD relacionales. Dichas BBDD están diseñadas para trabajar con *eXtensible Query Language* (XQL), que sigue el mismo propósito que SQL. XQL está diseñado para trabajar con documento XML jerárquicamente estructurados y puede proveer características de consulta como filtros y *joins*. Esta ventaja ofrece la posibilidad de poder almacenar datos centrándose en el documento XML.

Para poder navegar por los datos y extraerlos o introducirlos se utilizan dos estándares, *Xpath* y *Xquery*.

- **XPath**

XPath *(XML Path Language)* es un estándar del W3C, el cual es utilizado para poder acceder a partes de un documento XML, utilizado conjuntamente con el *eXtensible Stylesheet Language for Transformations* (XSLT).

Una vez se ha accedido a la parte, XPath proporciona herramientas básicas para la manipulación de cadenas, números y boleanos. XPath utiliza una sintaxis compacta no propia de los XML para poder facilitar el uso en URIs y en valores de atributos XML. XPath trabaja en la estructura lógica del documento en lugar de la sintaxis superficial.

- **XQuery**

XQuery fue diseñado para poder satisfacer las especificaciones del W3C Query Working Group. Se diseñó para ser un lenguaje en las que sus peticiones fueran sencillas y fáciles de entender. También se pretendía que fuese flexible a la hora de realizar peticiones a cualquier tipo de documento XML, incluidos los utilizados por las bases de datos. XQuery es la evolución de un lenguaje llamado Quilt, que estaba basado en otros lenguajes como XPath (XML Path Language), XQL y SQL.

XQuery opera sobre el nivel lógico de los documentos XML, es decir, su sintaxis. Al ser una extensión del XPath 2.0, cualquier expresión que se pueda ejecutar en éste y el XQuery 1.0, devolverá el mismo resultado en los dos casos.

Al ser unos lenguajes tan cercanos, todas las descripciones se generan a partir de una fuente común para poder asegurar coherencia entre los dos lenguajes; por lo tanto no es difícil pensar que los desarrolladores de ambos trabajos trabajen conjuntamente.

- Especificaciones:
    - Xquery/XPath Data Model (XDM) define el modelo de datos del cuál parten todas las expresiones.
    - El sistema usado por XQuery se basa en XML Schema.
    - Una de las principales características es que las peticiones XML tengan una sintaxis inteligible y que esté basado en la sintaxis de los propios XML.