FACULTAD DE INFORMÁTICA DE LA

UNIVERSIDAD DE MURCIA

PROYECTO INFORMÁTICO

RICAO v1.0:
Recuperación e Integración de Contenidos Audiovisuales empleando Ontologías

---

Alumno

Ignacio García Manotas

Director

Jesualdo Tomás Fernández Breis

Departamento de Informática y Sistemas

Febrero 2008

TABLA DE CONTENIDOS

0. RESUMEN 1

1. INTRODUCCIÓN 2
1.1 LA TELEVISIÓN DIGITAL. SEMÁNTICA EN CONTENIDOS AUDIOVISUALES 2
1.2 WEB SEMÁNTICA Y ONTOLOGÍAS. 5

2. ANÁLISIS DE OBJETIVOS Y METODOLOGÍA 8
2.1 OBJETIVOS DEL PROYECTO 8
2.2 HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS 11
2.2.1 Herramientas y Tecnologías de la Web Semántica 11
2.2.1 Herramientas y Tecnologías de Diseño 18
2.2.3 Herramientas y Tecnologías de Desarrollo 25

3. DISEÑO Y RESOLUCIÓN DEL TRABAJO 34
3.1 ARQUITECTURA DEL SISTEMA 34
3.2 MODELADO DE RECURSOS: E/R Y ONTOLOGÍA DE RECURSO 45
3.3 GENERACIÓN DE INSTANCIAS DE CONTENIDOS A PARTIR DE DOCUMENTOS HTML 58
3.4 DEFINICIÓN DE LAS CORRESPONDENCIAS ENTRE LOS RECURSOS 60
3.5 PROCESAMIENTO DE CONSULTAS 66
3.6 VISUALIZACIÓN DE RESULTADOS 76
3.7 ARQUITECTURA LÓGICA 77
3.8 COMPARACIÓN CON OTRAS HERRAMIENTAS 83

4. CONCLUSIONES Y VÍAS FUTURAS 84
4.1 CONCLUSIONES 84
4.2 VÍAS FUTURAS 86

5. BIBLIOGRAFÍA 87

6. ANEXO 92
A1. MANUAL DE USUARIO 92
Despliegue de la aplicación 92
Interfaz de Usuario del Sistema RICAO v1.0 93

INDICE DE FIGURAS

Figura 1. Evolución de la World Wide Web [18] 6
Figura 2. Protégé..... 17
Figura 3. Ontology Mapping 17
Figura 4. StarUml 21
Figura 5. Modelo-Vista-Controlador 23
Figura 6. Diagrama de Bloques del Sistema 35
Figura 7. Subsistema “Recuperación de Información” 37
Figura 8. Subsistema “Integración de Información”..... 38
Figura 9. Diagrama de Clases. Patrón “Front Controller” 39
Figura 10. Diagrama de Secuencias. Patrón “Front Controller”..... 40
Figura 11. Diagrama de Clases. Patrón DAO..... 43
Figura 12. Portal de contenidos audiovisuales ONO..... 45
Figura 13. Portal de contenidos audiovisuales Digital Plus 46
Figura 14. Portal de contenidos audiovisuales IMDb 46
Figura 15. Modelo Entidad Relación. Portal ONO 47
Figura 16. Modelo Entidad Relación. Portal Digital Plus 48
Figura 17. Modelo Entidad Relación. Portal IMDb..... 49
Figura 18. Modelo Entidad Relación Integrador 50
Figura 19. Clase Emision 52
Figura 20. Ontología ONO..... 52
Figura 21. Ontología Digital Plus..... 53
Figura 22. Ontología IMDb 54
Figura 23. Ontología Global 55
Figura 24. Ontología de definición de correspondencia entre ontologías. Jerarquía de clases..... 61
Figura 25. Formulario de búsqueda 66
Figura 26. Formulario de búsqueda. Ejemplo..... 69
Figura 27. Resultados de una búsqueda..... 74
Figura 28. Navegabilidad en la Interfaz de Usuario..... 75
Figura 29. Características de una Emisión asociada de un Contenido 75
Figura 30. Diagrama de paquetes. Sistema RICAO v1.0 78
Figura 31. Diagrama de paquetes. Subsistema “Recuperación de Información” 79
Figura 32. Diagrama de paquetes. Subsistema “Integración de Información”.. 80
Figura 33. Diagrama de Secuencias. Patrón Modelo Vista Controlador. 82
Figura 34. Avatar..... 83

0. RESUMEN

El incremento desmesurado de contenidos audiovisuales accesibles por el teleespectador hace necesario el uso de mecanismos de integración de información y búsqueda avanzados que les permitan obtener información detallada en base a sus preferencias. Los mecanismos de búsqueda sintáctica convencionales basados en la simple comparación de cadenas de texto son insuficientes para abordar el problema, siendo necesaria la incorporación de metainformación en los contenidos audiovisuales y el uso de un razonamiento semántico para las búsquedas.

RICAO constituye un intento de abarcar el problema sirviendo como base para la construcción de una metodología que abarque el proceso de desarrollo de sistemas basados en la recuperación e integración de información mediante ontologías utilizando como caso de estudio los portales de programación televisiva ONO y Digital+, y el cinematográfico IMDb.

1. INTRODUCCIÓN

1.1 LA TELEVISIÓN DIGITAL. SEMÁNTICA EN CONTENIDOS AUDIOVISUALES

La televisión ha sido desde sus comienzos un medio de comunicación que goza de una aceptación generalizada por parte de cualquier sector social, y de un grado de penetración casi absoluto en el hogar de los ciudadanos. La ya iniciada transición de la televisión analógica a la digital dará lugar no sólo a un aumento considerable de los contenidos audiovisuales que podrá recibir el teleespectador, sino también a la posibilidad de acceder a un sinfín de servicios telemáticos comerciales y administrativos disponibles hoy día en Internet.

La adopción de la norma MHP (*Multimedia Home Platform, Plataforma Multimedia del Hogar*) [40] superará los actuales problemas de normalización de los equipos descodificadores, dibujando un mercado horizontal, permitiendo el acceso a múltiples plataformas. El usuario podrá acceder a la programación de cientos de canales, dejando de ser abordable la búsqueda de contenidos haciendo uso de las soluciones actuales como las EPGs (*Electronic Program Guides, Guía electrónicas de programas*) [37] que muestran de forma organizada la programación del operador. Han surgido diferentes iniciativas para dotar a los contenidos audiovisuales de metainformación abriendo las puertas al razonamiento semántico acerca de los contenidos audiovisuales. Sin embargo, estas iniciativas no abordan procedimientos de búsqueda de contenidos en base a los intereses del usuario, ni procedimientos de integración de contenidos ofrecidos por diferentes fuentes.

Las iniciativas más importantes relativas a dotar de metainformación los contenidos audiovisuales son dos, ambas complementarias. La primera es TV-Anytime y la segunda, la norma MPEG-7.

La especificación TV-Anytime [43], normalizada por ETSI (*European Telecommunications Standards Institute, Instituto Europeo de Normas de Telecomunicaciones*) [36], trata de normalizar un procedimiento de envío al

usuario de la metainformación. Se trata de un consorcio de empresas que han desarrollado una propuesta para la normalización de: (1) una forma unívoca de identificar un contenido concreto; (2) un conjunto de elementos de metainformación para describir cualquier contenido genérico o una instancia del mismo; (3) otro conjunto de elementos de metainformación para describir los perfiles de los usuarios y su historial de compra/visionado de programas; y, finalmente, (4) una infraestructura para la publicación y localización de los contenidos por parte de creadores, operadores de televisión o terceras partes. Todo ello conforma un marco especialmente adecuado para la localización y procesamiento de contenidos, independientemente de la forma en que éstos sean distribuidos, estableciendo un excelente punto de arranque para la implementación de servicios de consultas basadas en comparaciones sintácticas. TV-Anytime no se ocupa en absoluto de definir elementos de metainformación para describir servicios telemáticos proporcionados a través de la televisión, ni de la infraestructura necesaria para localizar entidades que proporcionen esos servicios. TV-Anytime está teniendo una aceptación excelente entre los organismos internacionales encargados de normalizar las cuestiones referentes a la televisión digital.

En lo que se refiere a Europa, el DVB[1] (*Digital Video Broadcasting, Transmisión Digital de Video*) [35] ya ha iniciado el proceso de extensión de las tablas de información de servicio DVB-SI (*Digital Video Broadcasting-Service Information, Servicio de Información de Transmisión Digital de Video*) que acompaña a los contenidos, para normalizar un procedimiento de envío al usuario de la metainformación de acuerdo a lo definido por TV-Anytime.

En los últimos años, han surgido otras iniciativas de normalizar los distintos elementos que componen la metainformación de los contenidos audiovisuales. La iniciativa internacional de mayor repercusión es la del grupo MPEG[2] (*Moving Picture Experts Group, Grupo de Expertos de Imágenes en Movimiento*) [39] de ISO (*International Organization for Standardization,*

[1] Organismo encargado de crear y proponer los procedimientos de estandarización para la televisión digital compatible.
[2] Grupo de trabajo del ISO/IEC encargado de desarrollar estándares de codificación de audio y video.

Organización Internacional para la Estandarización) [41], que se ha plasmado en la norma MPEG-7. Entre los objetivos de esta norma está normalizar distintos elementos de metainformación para describir ciertas propiedades de interés de recursos multimedia en general. El lenguaje empleado para ello se llama DDL (*Description Definition Language, Lenguaje de Definición de Descripción*) [38] y es una sencilla extensión del XML Schema [44] del W3C, enriquecida con varios tipos de datos adicionales para describir características habituales en el contexto de lo audiovisual. RICAO se centra, principalmente, en la integración de distintos contenidos audiovisuales a los que puede tener acceso un teleespectador a través de las diferentes plataformas de televisión y de dotar a este tipo de asistentes telemáticos una capacidad de razonamiento semántico, que enriquezca la tradicional búsqueda sintáctica basada en la simple comparación de cadenas de texto, permitiendo al usuario obtener información de contenidos audiovisuales en bases a sus intereses.

1.2 WEB SEMÁNTICA Y ONTOLOGÍAS.

Gran parte del contenido actual de la World Wide Web se encuentra en forma de documentos escritos en HTML (*HyperText Markup Language, Lenguaje de Marcas de Hipertexto*), un lenguaje de marcas que permite determinar la disposición de ciertos elementos textuales acompañados por elementos multimedia tales como imágenes, tablas y formularios interactivos. Mediante este lenguaje, el ordenador es capaz de interpretar la disposición con que debe representar cada uno de estos elementos, pero no dispone de ninguna información acerca de su contenido semántico. Es decir, las páginas web actuales están diseñadas únicamente para ser leídas por personas, y no existe la posibilidad de que su significado sea procesado automáticamente por ordenadores.

Últimamente se ha acuñado el término Web 2.0 para referirse a una importante evolución de la web que ha pasado de estar formada por un conjunto de páginas estáticas a poder ser considerada un conjunto de aplicaciones web totalmente dinámicas e interactivas, cuyo contenido se va generando en tiempo real según las preferencias y contribuciones de su comunidad de usuarios.

El tamaño de la World Wide Web ha crecido de forma espectacular en los últimos años, lo que ha provocado un incremento notable en la dificultad de encontrar los recursos más interesantes acerca de un tema en concreto debido, en gran parte, por la ambigüedad de los términos empleados en las consultas que permiten realizar las herramientas de búsqueda. La Web Semántica, denominada a veces también Web 3.0, bajo la dirección del inventor de la web original, Tim Berners-Lee [42], pretende abordar la resolución de esta carencia, creando un mecanismo universal de intercambio de información, al añadir a la web ya existente una cierta semántica o significado, permitiendo establecer un mecanismo de procesamiento automático de su contenido semántico tratable automáticamente por un ordenador.

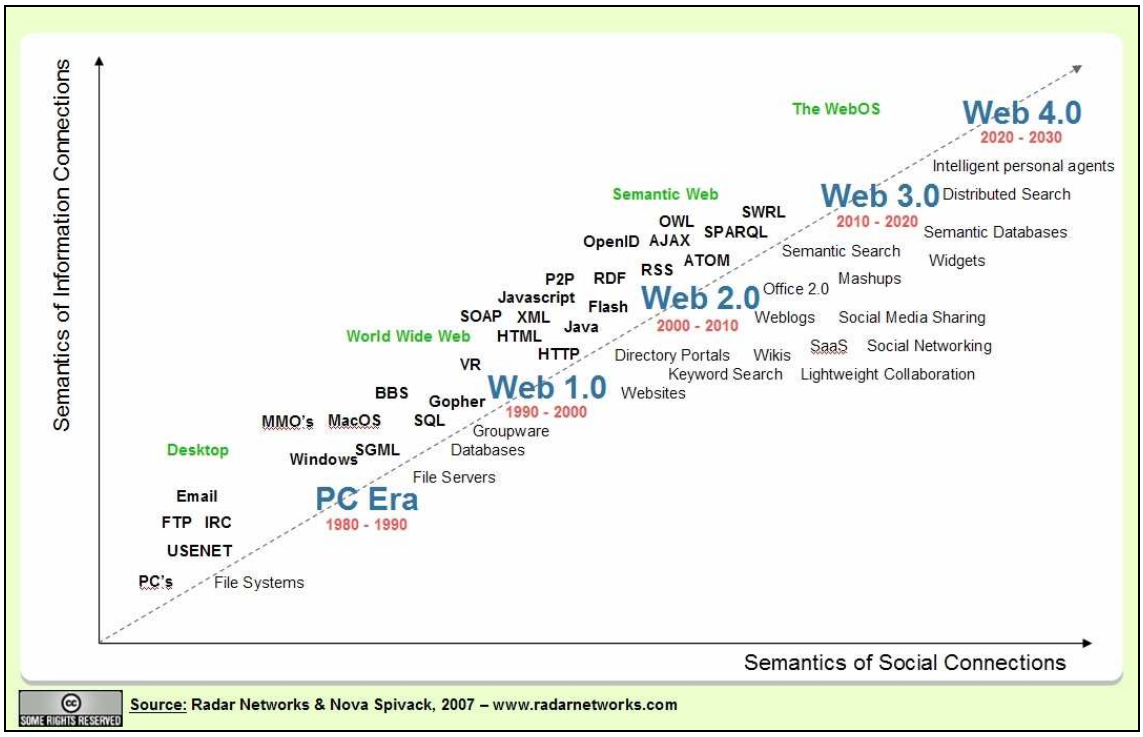


Figura 1. Evolución de la World Wide Web [18]

Gracias a ello, será posible disponer de un entorno en el que unos determinados programas denominados “agentes”, puedan llevar a cabo tareas sofisticadas accediendo a la información presente en un gran número de páginas de esta nueva Web Semántica, sin necesidad de ningún tipo de inteligencia artificial, ya que la información necesaria se encontrará en las propias páginas codificada semánticamente, junto con unas reglas de inferencia que permitirán entender la relación existente entre los diferentes recursos.

Para dotar a la web de un significado comprensible por los ordenadores, es necesario disponer de una cierta forma de representar el conocimiento. Tradicionalmente, los sistemas que pretendían almacenar un cierto conocimiento, lo hacían de forma centralizada, de modo que se compartiesen los mismos términos y que el tipo de preguntas realizables fuese fácilmente controlable y restringible. Sin embargo, la Web Semántica, dada la gran magnitud a que aspira, debe basarse en un sistema distribuido de representación y almacenamiento del conocimiento, que permita plantear un número prácticamente ilimitado de cuestiones acerca de este conocimiento.

[26] Como solución para la representación distribuida del conocimiento, la Web Semántica propone la utilización de colecciones de información conocidas como ontologías. En filosofía, una ontología es una teoría acerca de la naturaleza de la existencia y de los tipos de cosas que existen. Sin embargo, los investigadores en Inteligencia Artificial han incorporado este término a su jerga particular con el significado de documento o fichero que define formalmente un conjunto de conceptos que se organizan de forma jerárquica, y que establece las propiedades y relaciones que existen entre ellos, a lo que se añade un conjunto de reglas de inferencia que permiten manipular automáticamente esta información.

Las ontologías constituyen un mecanismo muy potente de representación del conocimiento de forma distribuida ya que permiten que cada página web desarrolle su propia ontología y posteriormente se establezcan relaciones de equivalencia entre ontologías de distintas páginas Web interrelacionadas, resolviendo a la vez los problemas de ambigüedad que pudieran presentarse al constituir representaciones formales de la información.

Debido a esta potencia en la representación del conocimiento, y a la existencia de un importante número de estándares y desarrollos nacidos alrededor suyo, hemos elegido las ontologías como estructura de representación para la información de los contenidos audiovisuales con los que vamos a trabajar.

2. ANÁLISIS DE OBJETIVOS Y METODOLOGÍA

2.1 OBJETIVOS DEL PROYECTO

El objetivo principal de este proyecto es el diseño e implementación de una arquitectura genérica, modular y multiplataforma capaz de:

1) Analizar información ofrecida por determinados recursos de contenidos audiovisuales disponibles en Internet y almacenarla en ontologías.

2) Ofrecer un mecanismo de búsqueda de contenidos, cuyos resultados sean la integración de información ofrecida por diferentes ontologías.

A continuación se presentan un conjunto de objetivos derivados del objetivo principal.

1 Estudio de los diferentes patrones de diseño a aplicar con el fin de obtener una arquitectura genérica y modular.

2 Estudio y análisis de las tecnologías y herramientas de desarrollo a utilizar en la implementación de la arquitectura.

2.1 Estudio y análisis de la plataforma de desarrollo utilizar en la implementación.

2.2 Estudio y análisis de las librerías disponibles de software libre, en el lenguaje de programación seleccionado, para su uso e implantación en la arquitectura.

2.2.1 Evaluación y selección de APIs (*Application Programming Interface, Interfaz de Programación de Aplicaciones*) de software libre para el tratamiento de ontologías en OWL, de cara a determinar la más idónea para incorporar a nuestro proyecto.

2.2.2 Evaluación y selección de APIs de software libre para el análisis de documentos HTML, teniendo en cuenta la posibilidad de encontrar errores en los documentos.

3 Estudio de las diferentes herramientas a utilizar en el diseño y en la implementación de la arquitectura.

4 Selección de los recursos de Internet a usar en la arquitectura.

5 Definición y modelado de las ontologías del dominio necesarias para la arquitectura.

5.1 Definición y modelado de una ontología para cada uno de los recursos de contenidos audiovisuales a tratar.

5.2 Definición y modelado de una ontología global que integre a cada una de las anteriores.

6 Automatización de la generación de instancias ontológicas a partir del análisis de documentos HTML ofrecidos por recursos disponibles en Internet.

6.1 Estudio de la estructura de los documentos HTML ofrecidos por los recursos para su posterior análisis.

6.2 Evaluación y selección de un buen analizador de documentos HTML.

6.3 Automatización eficiente.

6.4 Desarrollo de una interfaz de usuario dónde indicar la información de los recursos a almacenar en la ontología.

7 Establecimiento de la correspondencia entre diferentes ontologías con semántica similar o análoga para su posterior integración.

8 Diseñar de una interfaz de búsqueda de contenidos audiovisuales cuyo resultado sea la integración de la información solicitada disponible en deferentes ontologías.

8.1 Integración de la información ofrecida por las diferentes ontologías.

8.1.1 Adquisición de la información almacenada en las ontologías haciendo uso de un lenguaje de consultas.

8.1.2 Integración de los resultados de las consultas en una ontología englobadora.

8.1.3 Impresión de los resultados de la integración a través de una interfaz de usuario.

2.2 HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS

2.2.1 Herramientas y Tecnologías de la Web Semántica

Utilizaremos ontologías para la representación del conocimiento de nuestro dominio.

ONTOLOGÍAS

Según la definición de *T.R.Gruber* [4], posteriormente refinada por *R.Studer* [20], una ontología es una *"especificación formal y explícita de una conceptualización compartida"*. En esta definición, "conceptualización" se refiere a un modelo abstracto de algún fenómeno del mundo del que se identifican los conceptos que son relevantes, "explícito" hace referencia a la necesidad de especificar de forma consciente los distintos conceptos que conforman una ontología, "formal" indica que la especificación debe representarse por medio de un lenguaje de representación formalizado y "compartida" indica que la ontología describe un conocimiento aceptado por un grupo de usuarios. Las ontologías de nuestro proyecto van a estar formadas por los siguientes tres elementos:

- ***Conceptos o Clases***: Un concepto es un conjunto de objetos u entidades con cierto interés para el modelo del dominio y con unas características similares que nos permiten agruparlos. Estos conceptos se pueden organizar en una jerarquía de conceptos y subconceptos, de forma que los subconceptos son más específicos que los conceptos que los preceden en la jerarquía, al heredar todas sus características y poder añadir alguna nueva. Por simplicidad, consideraremos conceptos también a las instancias o elementos de estos conjuntos (que en la terminología de la *Web Semántica* se denominan individuos), tomándolos como si fuesen conjuntos monoelementales, y suponiendo que son subconceptos del concepto o clase al que pertenecen. Entre estos conceptos será posible establecer relaciones de equivalencia y de desigualdad.

- ***Atributos o Datatype Properties***: Un atributo representa una característica del concepto al que pertenece, y que se puede describir por medio de un valor básico. Se modela matemáticamente mediante una relación cuyo dominio es el propio concepto y cuyo rango es un tipo de dato básico (tal como int, float, string o boolean). Al igual que ocurre con los conceptos, es posible agrupar los atributos en una jerarquía de forma que los subatributos tienen un rango más restringido que los atributos que los preceden, pudiendo llegar esta restricción a implicar que el rango contenga únicamente un valor. Y de la misma forma que ocurría con los conceptos, también será posible establecer relaciones de equivalencia y desigualdad entre atributos.
- ***Relaciones u Object Properties***: Una relación representa la existencia de un nexo común entre dos conceptos. Se modela matemáticamente mediante una relación cuyo dominio es un concepto, que se denomina sujeto y cuyo rango es el otro concepto, que se denomina objeto. Tal como ocurre con conceptos y atributos, es posible establecer entre las relaciones jerarquías de relaciones y subrelaciones más específicas, así como establecer la equivalencia o desigualdad entre dos relaciones determinadas.

A conceptos, atributos y relaciones podremos asociarles unas determinadas etiquetas que constituirán un nombre o expresión lingüística en un cierto idioma, para poder así reconocerlos en textos escritos.

En el caso concreto de nuestro proyecto, definiremos cuatro ontologías, tres para almacenar los contenidos audiovisuales disponibles en portales de Internet y otra para integrar la información de las anteriores.

CONSTRUCCIÓN DE ONTOLOGÍAS. OWL

Para llevar a cabo la construcción de ontologías, se han desarrollado a nivel internacional dos iniciativas diferentes, aunque ambas toman como punto de partida los lenguajes RDF y RDF Schema, tratando de extender su capacidad para expresar las propiedades de los recursos con nuevos constructores, como puedan ser elementos para establecer restricciones de cardinalidad, relaciones de transitividad o propiedades sobre conjuntos disjuntos, complementarios o intersecciones.

Existe una relación importante entre los lenguajes XML, XML Schema, RDF, RDF Schema y OWL. A continuación se describen cada uno de ellos y su relación existente.

XML (eXtensible Markup Language, Lenguaje de Marcas Extensible), es un lenguaje que permite crear etiquetas arbitrarias de un tipo similar a las de *HTML* para anotar las páginas *web*, pero sin especificar el significado de las nuevas estructuras así creadas, quedando este significado a la discreción de su creador. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Entenderemos que un documento XML está bien formado si cumple todas las especificaciones básicas en cuanto a formato (tales como que debe tener un único elemento raíz, que cada subelemento debe tener etiquetas de inicio y de fin correctamente anidadas, etc.). Sin embargo, para poder considerar que es también válido, deberá cumplir una serie de condiciones relativas al orden y contenido de las etiquetas, que deberán especificarse en un archivo externo utilizando un lenguaje de descripción de validez estructural, tal como el *XML Schema*.

XML Schema es, por tanto, un lenguaje que nos permite especificar la estructura que debe tener un lenguaje creado con XML, imponiéndole una serie de restricciones en cuanto a los elementos que puede contener un documento, cómo están organizados estos elementos, los atributos que pueden tener y de qué tipos. Este lenguaje cumple por sí mismo el estándar *XML*, y le añade una serie de tipos de datos básicos y compuestos que podemos asociar a los

atributos de los elementos de un documento *XML* a la hora de especificar la estructura que debe tener.

RDF (Resource Description Framework, Sistema de Descripción de Recursos), es un lenguaje que permite utilizar etiquetas *XML* para expresar el significado del contenido de la página mediante afirmaciones acerca del mismo, codificadas mediante tripletas (sujeto, verbo, objeto), donde cada uno de los elementos de la tripleta queda identificado sin ambigüedad mediante un *URI (Uniform Resource Identifier, Identificador Uniforme de Recurso)*. Constituye por tanto un modelo de datos muy simple que permite describir los recursos a que se refiere y las relaciones entre ellos. El sujeto de la tripleta es el recurso que se está describiendo, el predicado la propiedad o relación que se desea establecer acerca del recurso, y por último el objeto es el valor de la propiedad o el otro recurso con que se establece la relación. Esta terminología procede de la lógica y de la lingüística, en las que las estructuras predicativas se utilizan también para dar significado a las representaciones sintácticas.

RDF Schema es un lenguaje extensible que al igual que *RDF* puede utilizar etiquetas de *XML* en sus construcciones, y cuya finalidad es la de describir vocabularios de *RDF*, es decir, ontologías, estructurando así los recursos del *RDF*. Constituye por tanto una extensión semántica de *RDF* destinada a la representación del conocimiento, y es el primero de los lenguajes que nos permiten describir directamente ontologías, si bien su potencia expresiva resulta bastante exigua. Mediante sus construcciones podemos agrupar los recursos en clases, asociar un rango y un dominio a sus propiedades y establecer mediante generalización jerarquías de clases y propiedades. También es posible asociar etiquetas a clases y propiedades.

OWL (Web Ontology Language, Lenguaje de Ontologías de la Web), como su nombre indica, es el lenguaje definitivo para la creación de ontologías e intercambio de contenidos semánticos a través de la web, que utilizaremos con este fin a lo largo de nuestro proyecto. Al igual que ocurría con *RDF Schema*, *OWL* constituye una ampliación del vocabulario de *RDF*, con la diferencia de que su potencia expresiva es muy superior, lo que le hace

totalmente apropiado para representar con la complejidad necesaria el conocimiento asociado a cualquier dominio.

Así pues, *OWL* nos permite estructurar los recursos en clases y describir instancias o individuos de estas clases, lo que para nosotros serán los conceptos de la ontología. También diferencia entre propiedades cuyo rango es un tipo de datos básico (que para nosotros serán atributos) y propiedades cuyo rango es un objeto (que para nosotros serán relaciones), siendo posible restringir la extensión de estos rangos. Incluso permite establecer la equivalencia o diferenciación entre clases, propiedades e individuos. Y, al ser una ampliación de *RDF*, puede utilizar también las construcciones de *RDF Schema* para definir relaciones jerárquicas entre clases y propiedades.

OWL se presenta en tres variantes, cada una de las cuales posee una expresividad superior a la anterior, si bien presenta mayores problemas que ésta a la hora de aplicar reglas de inferencia al conocimiento que representa. Cada una de estas variantes puede tener un interés específico para un grupo de usuarios o desarrolladores, por lo que se debe elegir la más apropiada teniendo en cuenta sus características:

- *OWL Lite*. Está especialmente indicado para usuarios que necesitan una clasificación jerárquica con restricciones simples, permitiendo una rápida migración de tesauros y taxonomías simples. Resulta más sencillo de implementar que las demás variantes, pero su expresividad formal se basa en la lógica *SHIF(D)*, que no permite un razonamiento muy potente.

- *OWL DL.* Resulta de especial utilidad para aquellos usuarios que desean una máxima expresividad, pero manteniendo la *completitud computacional* (es decir, garantizando la posibilidad de que cualquier posible conclusión pueda alcanzarse mediante las reglas de inferencia) y la *decidibilidad* (es decir, garantizando que todos los cálculos se puedan realizar en un periodo de tiempo no infinito). Su expresividad se corresponde con la de la lógica descriptiva *SHOIN(D)*.

- *OWL Full*. Se dirige a usuarios que desean una máxima expresividad y libertad sintáctica, sin garantías computacionales. Este lenguaje incluso permite a una ontología aumentar el significado de los vocabularios predefinidos en *RDF* u *OWL*, por lo que resulta imposible diseñar un sistema que pueda realizar razonamientos sobre todas las características de este lenguaje.

Las ontologías creadas en nuestro proyecto están definidas en el lenguaje OWL (Web Ontology Language, Lenguaje de Ontologías de la Web), en concreto en la variante OWL-DL. Las ontologías las hemos definido con el editor de ontologías “Protégé” y para el acceso a ellas hemos usado la API JENA, que incluye el lenguaje de consultas SPARQL. Para definir la correspondencia entre dos ontologías utilizaremos la herramienta “Ontology Mapping”.

PROTÉGÉ

Protégé [28] es una plataforma de software libre de la *Universidad de Stanford* que ofrece un conjunto de herramientas para construir modelos de dominios y aplicaciones basadas en el conocimiento con ontologías. En su núcleo, *Protégé* implementa un rico conjunto de estructuras de modelado del conocimiento y permite la creación, visualización y manipulación de ontologías en varios formatos de representación. *Protégé* puede ser personalizado según las características de un determinado dominio, y puede ser extendido mediante una arquitectura modular y una *API* en *Java* para construir herramientas y aplicaciones basadas en el conocimiento.

Si bien el desarrollo de ontologías no entra dentro de los objetivos de nuestro proyecto, debemos disponer de ontologías de prueba para desarrollar la depuración y validación de la aplicación que nos proponemos implementar, por lo que resulta interesante disponer de herramientas como ésta para su creación y edición.

Figura 2. Protégé

ONTOLOGY MAPPING

"Ontology Mapping" es herramienta capaz de establecer una correspondencia entre dos ontologías diferentes o una correspondencia entre una base de datos relacional y una ontología. El significado semántico de la correspondencia entre dos ontologías está definido en otra ontología definida por el usuario. El resultado de realizar la correspondencia entre dos ontologías es un fichero XML. La herramienta ha sido desarrollada por el grupo de investigación TECNOMOD [21] de la Universidad de Murcia.

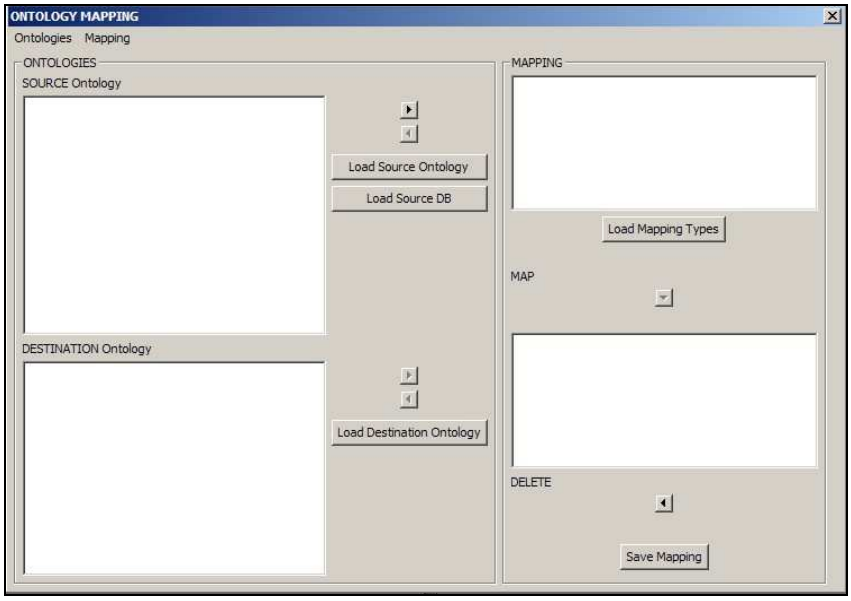


Figura 3. Ontology Mapping

2.2.1 Herramientas y Tecnologías de Diseño

Como metodología de diseño se ha elegido una metodología orientada a objetos por diversas razones:

- Proximidad de los conceptos de modelado respecto de las entidades del mundo real.
    - Mejora la captura y validación de requisitos.
    - Acerca el “espacio del problema” y el “espacio de la solución”.
- Constituye un sistema de modelado integrado de propiedades estáticas y dinámicas del ámbito del problema.
- Presenta conceptos comunes de modelado durante el análisis, diseño e implementación.
    - Facilita la transición entre las distintas fases.
    - Favorece el desarrollo iterativo del sistema.
    - Disipa la barrera entre el “qué” y el “cómo”.
- Facilita la construcción, mantenimiento y reutilización.

Como lenguaje de modelado hemos usado UML y como herramienta para el desarrollo de modelos UML la herramienta StarUML.

UML

UML (Unified Modeling Language, Lenguaje Unificado de Modelado) [9] es un lenguaje de modelado orientado a objetos que ha tenido una aceptación muy considerable dentro de la comunidad de la Ingeniería del Software. Está apoyado en gran manera por el *OMG (Object Management Group, Grupo de Gestión de Objetos)*, y se utiliza en tal cantidad de proyectos en organizaciones de todo el mundo, que se ha convertido en un estándar de facto.

UML es un lenguaje para especificar y no un método o un proceso. Es usado para visualizar, especificar, construir y documentar un sistema de software. *UML* ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes

de programación, esquemas de bases de datos y componentes de software reutilizables.

En el diseño de nuestro sistema se han utilizado cuatro tipos diferentes de diagramas de entre los trece distintos que ofrece:

- *Diagramas de despliegue*, para modelar la distribución de los distintos componentes físicos utilizados en la implementación de un sistema, así como las relaciones entre ellos.
- *Diagramas de paquetes*, para mostrar los diferentes subsistemas lógicos en que se divide el sistema y mostrar las dependencias existentes entre ellos.
- *Diagramas de clases*, para especificar los diferentes tipos de objetos relevantes al dominio de la aplicación, así como las relaciones entre los mismos.
- *Diagramas de secuencia*, en los que se muestra la evolución en el tiempo de los mensajes que intercambian los diferentes actores y objetos del sistema en un escenario determinado. Es decir, detallan la funcionalidad de los casos de uso en el tiempo.

CONCEPTUALIZACIÓN. MODELO ENTIDAD RELACIÓN

La esquemas conceptuales suelen ser utilizados como parte de la fase del diseño conceptual de bases de datos, sin embargo, hemos decidido adoptarla como paso previo a la construcción de las ontologías de nuestro sistema.

El esquema conceptual es una descripción de alto nivel de los contenidos de la información de un sistema teniendo como objetivo el entendimiento completo de la estructura, semántica, relaciones y restricciones del sistema. El esquema conceptual es independiente de los aspectos de implementación y resume los requisitos sobre la información del sistema. El esquema conceptual resulta muy útil para propósitos de documentación y

puede servir como vehículo de comunicación entre usuarios, diseñadores y analistas. Los modelos de datos de alto nivel incluyen conceptos más fáciles de entender que los modelos de datos de nivel más bajo.

Existen diferentes notaciones y diferentes tipos de esquemas conceptuales. Nosotros hemos hecho uso del Modelo Entidad Relación con la notación MPM 1999 [15] [27]. Hemos construido un esquema conceptual para cada uno de los portales de contenidos audiovisuales en los que se ha basado el proyecto.

STAR-UML

StarUML [19] es una herramienta, con licencia GNU/GPL[3], de modelado software basada en los estándares UML y MDA (*Model Driven Architecture, Arquitectura Dirigida por Modelos*) disponible para Windows y basada en UML versión 1.4. Soporta once tipos de diagramas y la notación de UML 2.0.

A diferencia de muchos productos existentes, StarUML maneja todos los archivos en el formato estándar XML siendo fácil leer y modificar sus estructuras usando un analizador gramatical XML. Este hecho es una ventaja ya que asegura que los modelos software construidos permanecerán útiles para más de una década. StarUML es flexible y extensible permitiendo añadir frameworks para extender su funcionalidad. Existen frameworks para las plataformas .NET y J2EE entre otras.

StarUml ofrece una excelente extensibilidad permitiendo que cualquier lenguaje que soporta COM (Script Básico Visual, Java Script, VB, Delfos, C ++, C #, VB.NET, Python, etc.) puede ser usado para desarrollar complementos para la aplicación. StarUml incluye multitud de complementos con diferentes funcionalidades, genera código fuente en diferentes lenguajes de programación y viceversa (Java, C#, C++), puede importar archivos de la aplicación propietaria “Rational Rose”, puede intercambiar metadatos con otras

[3] **GPL** (Licencia Pública General) es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

herramientas haciendo uso de XMI (*XML Metadata Interchange, Intercambio de Metadatos XML*) y tiene soporte para multitud de patrones de diseño. Por último StarUML puede verificar automáticamente los modelos construidos por el usuario, facilitando un temprano descubrimiento de errores.

Figura 4. StarUml

PATRONES DE DISEÑO

o PATRÓN DAO (*Data Access Object, Objeto de Acceso de Datos*) [13].

Muchas aplicaciones del mundo real necesitan utilizar persistencia de datos. Cada fuente de datos suele tener su propia API de acceso y en caso de que una aplicación cambie de fuente de datos, tendrá que realizar los cambios necesarios en la lógica de negocio para adaptarla a la nueva API. El objetivo del patrón DAO es abstraer y encapsular el acceso a las diferentes fuentes de datos de modo que se pueda mantener la lógica de negocio de forma independiente a la fuente de datos utilizada. Los componentes de negocio harán uso de una interfaz común para todas las fuentes de datos ocultando los detalles de implementación de acceso. El DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos y su correspondiente conexión. En caso de que se desea cambiar de fuente de datos dinámicamente se puede utilizar el patrón Abstract Factory creando una factoría abstracta.

El patrón DAO usualmente ha estado ligado a las bases de datos como fuente de datos, sin embargo, en nuestro proyecto la fuente de datos son las

ontologías. Usar este patrón tiene una serie de ventajas e inconvenientes que hay que tener en cuenta según las características de la aplicación.

- *Permite la Transparencia*. Los objetos de negocio pueden utilizar la fuente de datos sin conocer los detalles específicos de su implementación.
- *Permite una migración más fácil*.
- *Reduce la complejidad del código de los objetos de negocio*.
- *Centraliza todos los accesos a datos en una capa independiente*.
- *Añade una capa extra*. Los DAO crean una capa de objetos adicional entre el cliente y la fuente de datos que necesitamos diseñar e implementar para obtener los beneficios de este patrón.
- *Necesita diseñar un árbol de clases***.** Cuando se utiliza una estrategia de factorías, necesitamos diseñar e implementar el árbol de factorías concretas y el árbol de productos concretos producidos por las factorías. Necesitamos justificar este esfuerzo adicional para ver si merece la pena dicha flexibilidad. Esto incrementa la complejidad del diseño.

o PATRÓN MODELO-VISTA-CONTROLADOR [3]

La arquitectura MVC (*Model/View/Controller*) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Sus características principales son que el modelo, las vistas y los controladores se tratan como entidades separadas.

En la figura siguiente, vemos la arquitectura MVC en su forma más general. Hay un modelo, múltiples controladores que manipulan ese modelo, y hay varias vistas de los datos del modelo, que cambian cuando cambia el estado de ese modelo.

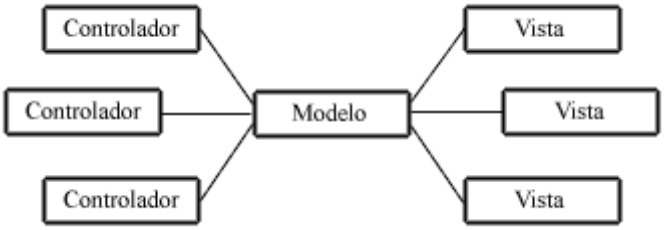


Figura 5. Modelo-Vista-Controlador

Definición de las partes

El *modelo* es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El modelo no tiene conocimiento específico de los controladores o de las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notificar a las vistas cuando cambia el modelo.

La *vista* es el objeto que maneja la presentación visual de los datos representados por el modelo. Genera una representación visual del modelo y muestra los datos al usuario. Interactúa con el modelo a través de una referencia al propio modelo.

El *controlador* es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del modelo o por alteraciones de la vista. Interactúa con el modelo a través de una referencia al propio modelo.

Este modelo de arquitectura presenta varias ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado.
- La conexión entre el modelo y sus vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.
- Nos permite utilizar los mismos objetos del modelo para diferentes vistas.
- Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.

Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas Java.

o PATRÓN FRONT-CONTROLLER [12]

Las peticiones de los usuarios en una aplicación web se pueden gestionar de forma centralizada o distribuida. Una gestión distribuida da lugar a una aplicación de baja calidad debido a los siguientes problemas:

- Se requiere que cada vista gestione sus peticiones, lo que normalmente ocasiona una duplicación de código.
- El control distribuido es más difícil de mantener, ya que los cambios se tienen que realizar en numerosos lugares.

Con el patrón Front Controller se consigue una gestión centralizada llevada a cabo por un controlador que recibe todas las peticiones entrantes de los clientes. El controlador será el encargado de implementar servicios de seguridad, manejar errores y delegar las peticiones a otros componentes de la aplicación que se encargarán de generar la vista adecuada para el usuario. Centralizar el control en el controlador y reducir la lógica de negocio en la vista permite reutilizar el código entre peticiones. Es una aproximación preferible a la alternativa de embeber código en varias vistas porque esta aproximación trata con entornos más propensos a errores, y menos reutilizables.

Aunque el patrón Front Controller sugiere la centralización del manejo de peticiones, no limita el número de manejadores en el sistema. Además una aplicación podría utilizar varios controladores en un sistema, cada uno correspondiente a un conjunto de servicios distintos.

2.2.3 Herramientas y Tecnologías de Desarrollo

Para el desarrollo e implementación de RICAO, hemos decidido utilizar el paradigma de programación orientado a objetos, de forma paralela a la metodología de diseño elegida, que también es orientada a objetos, de modo que a la hora de desarrollar nos resultará más sencillo representar los conceptos que forman parte del diseño.

Las razones que nos han llevado a esta decisión son muy similares a las expuestas a la hora de valorar la elección de metodología de diseño, y se derivan de las principales características de este paradigma de programación:

- *Abstracción*. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar una tarea, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características.

- *Encapsulamiento* u ocultación de la información. Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone a otros objetos una interfaz que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas.

- *Polimorfismo*. comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre y una cierta semejanza conceptual. Al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto concreto que se esté usando.

- *Herencia*. Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el *polimorfismo* y el *encapsulamiento* permitiendo a los

objetos ser definidos y creados como tipos especializados de objetos preexistentes.

JAVA

Uno de los condicionantes que nos han guiado a la hora de abordar el desarrollo del sistema informático resultado de este proyecto ha sido el de conseguir una herramienta multiplataforma, capaz por tanto de funcionar en diferentes sistemas operativos. También se ha planteado el seleccionar un lenguaje de programación conocido, ampliamente extendido, y que permita fácilmente la reutilización del código desarrollado en futuras revisiones o versiones de la herramienta. La solución ha estado clara desde el principio, y ha sido la de utilizar *Java* como lenguaje de programación de la herramienta.

Además de los dos requisitos mencionados, se han valorado también sus otras no menos importantes características, que han hecho de él el principal lenguaje de programación multiplataforma:

- ***Orientación a objetos***. El diseño de la aplicación se ha realizado en base a clases de objetos percibidos del dominio del sistema. Es por ello muy importante la elección de un lenguaje de programación que permita representar estos objetos. Además, *Java* es un lenguaje de herencia simple, aspecto que permite la reutilización de código sin complicar la jerarquía de clases, pero que soporta herencia múltiple de interfaces, lo que resulta de gran utilidad.
- ***Robusto***. Frente a otros lenguajes orientados a objetos, *Java* es tipado estáticamente, lo que permite la comprobación de tipos en tiempo de compilación. Además, el uso de excepciones y la comprobación de límites de tablas proporcionan una gran seguridad en tiempo de ejecución: ante un fallo en la programación, el programa no se cuelga, sino que produce una excepción.
- ***Seguro***. La ejecución de programas en *Java* pasa por una serie de comprobaciones, tanto de privilegios como de *bytecodes*, que hacen que se produzcan excepciones cuando un programa intente acceder a un

fichero para el que no tiene permiso o cuando intente acceder a una zona del sistema.

- ***Multiplataforma***. Ésta es una de las características más importantes y reconocidas de *Java*. El hecho de que los programas Java no se ejecuten directamente sobre el sistema operativo, sino sobre la máquina virtual *Java*, hace que puedan escribirse y ejecutarse programas *Java* independientemente de la máquina y el sistema operativo.
- ***Interpretado***. El hecho de que *Java* sea interpretado hace de él un lenguaje lento, pero es la clave de otras características como la multiplataforma o el chequeo de *bytecodes*.
- ***Distribuido***. *Java* ofrece mecanismos para interactuar con otras máquinas mediante *sockets* vía *TCP/IP*.
- ***JDBC***. Java ofrece la interfaz *JDBC* para acceder a una base de datos relacional, como la utilizada para almacenar toda la información del sistema desarrollado.
- ***En expansión***. *Java* es uno de los lenguajes que ha alcanzado mayor popularidad en los últimos años, y parece claro que seguirá siendo así.
- ***Gratuito***. No es necesario pagar ninguna licencia para utilizar el lenguaje ni la máquina virtual sobre la que se ejecutará. Simplemente se descarga de *Internet* y se instala.

JENA Ontology API

JENA es una plataforma de desarrollo para el desarrollo de aplicaciones de la Web Semántica. Destaca por su utilidad para nuestro proyecto ya que permite trabajar con ontologías en *OWL* y *RDF Schema*, con una potencia que cubre todos los aspectos del *OWL FULL*, haciendo posible incluso utilizar algoritmos de inferencia, sin perder robustez en ningún momento. Es por ello que sobresale por encima de otras *APIs* que tienen el mismo objetivo y que consideramos como opciones a la hora de realizar la implementación, pero que descartamos por su clara inferioridad frente a JENA, tal como ocurre con el caso de OWL API.

JENA permite usar el lenguaje de consultas SPARQL (*Protocol and RDF Query Language, Lenguaje de Consulta SPARQL para RDF*), del cual hemos hecho uso.

ECLIPSE

Eclipse es una poderosa plataforma de software libre que permite integrar diferentes aplicaciones y herramientas para construir un completo entorno integrado de desarrollo que presenta una gran potencia y flexibilidad debidas a una arquitectura de *plugins* que hace posible añadir funcionalidades a la plataforma de modo sencillo y transparente. Además de su potencia y flexibilidad es muy cómoda de utilizar por el gran número de herramientas de apoyo a la programación que contiene, y es por ello que lo hemos seleccionado como plataforma de desarrollo para nuestro proyecto.

El proyecto *Eclipse* se lanzó originalmente en Noviembre de 2001, cuando *IBM* donó 40 millones de dólares del código fuente de *Websphere Studio Workbench* y formó el *Eclipse Consortium* para controlar el desarrollo continuado de la herramienta. El objetivo de Eclipse era desarrollar una plataforma industrial robusta, con todas las características y de calidad industrial para el desarrollo de herramientas altamente integradas.

Como ventajas que aporta *Eclipse*, cabe destacar explícitamente:

- Es una herramienta de software libre.
- Soporta la construcción de una variedad de herramientas para el desarrollo de aplicaciones.
- Soporta el desarrollo de aplicaciones basadas en *GUI* y *non-GUI*.
- Soporta *plugins* que manipulan diferentes tipos de lenguajes y ficheros, como *Java, C, C++, GIF, JPG*, etc.
- Está disponible para una gran cantidad de sistemas operativos incluyendo los más extendidos *Windows* y *Linux*.

- Ofrece a los desarrolladores herramientas que facilitan la creación de *plugins*.
- Mediante *JDT* facilita la creación de aplicaciones programadas en *Java*.

JERICHO HTML PARSER

El parser HTML Jericho [7] es una simple pero potente librería java que permite el análisis y la manipulación de partes de un documento HTML. Está bajo licencia EPL (*Eclipse Public License*) y LGPL (*Lesser General Public License*). Las características más importantes de esta librería se citan a continuación:

- La presencia de código incorrecto HTML no interfiere al realizar el análisis del resto documento haciendo este analizador ideal para usarlo en el mundo real.

- Las etiquetas ASP, JSP, PSP, PHP y Mason son reconocidas por el analizador.

- No está basado ni en eventos ni en árboles. Se basa en una combinación de simples búsquedas de texto, reconocimiento eficiente de etiquetas y su posición. El texto del documento es cargado completamente en memoria y solo los segmentos relevantes son utilizados.

- Comparado a analizadores basados en la construcción de árboles como DOM, los requisitos de memoria y de recursos pueden ser bastante menores si solo es necesaria una pequeña sección del documento. Si el documento contiene errores puede ser fácilmente ignorable a diferencia de los analizadores basados en árboles los cuales deben identificar todos los nodos en el documento desde arriba hacia abajo.

- Comparado con analizadores basados en eventos como SAX, la interfaz es de un nivel mucho más elevado e intuitivo.

Este analizador es perfecto para nuestro sistema, donde, como se explicará más adelante, es necesario ignorar los errores encontrados en los documentos HTML ofrecidos por los portales de Internet que vamos a tratar.

ARQUITECTURA JAVA PARA ENLACES XML

JAXB (*Java Architecture for XML Binding, Arquitectura Java para enlaces XML*) proporciona una manera rápida de enlazar los documentos XML y los objetos Java. Dado un XML Schema el compilador JAXB genera un conjunto de clases Java que contienen todo el código para analizar los documentos XML que cumplen el XML Schema. Una aplicación que utilice las clases generadas puede construir un árbol de objetos Java que representa un documento XML, manipular el contenido del árbol, y regenerar los documentos del árbol, todo ello en XML sin requerir que el desarrollador escriba código de análisis y de proceso complejo.

En nuestro sistema, JAXB ha sido utilizado en diferentes ocasiones, para generar un documento XML de peticiones de búsqueda de usuario, para analizar las respuestas de consultas SPARQL realizadas y, por último, para analizar los documentos XML generados por la herramienta Ontology Mapping descrita anteriormente.

SERVLET, JSP y ETIQUETAS JSP

Los servlets y Java Server Pages (JSP) son dos métodos de creación de páginas web dinámicas usando el lenguaje Java. En ese sentido son similares a otros métodos o lenguajes tales como el PHP, los CGI (*Common Gateway Interface, Interfaz de Entrada Común*), programas que generan páginas web en el servidor, o los ASP (*Active Server Pages*), un método específico de Microsoft. Sin embargo, se diferencian de ellas por lo siguiente. Los JSP y servlets se ejecutan en una máquina virtual Java, lo cual permite que, en principio, que se puedan usar en cualquier tipo ordenador, siempre que exista una máquina virtual Java para él. Cada servlet o JSP se ejecuta en su propia hilo, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo. Su persistencia le permite también hacer una serie de cosas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo.

Un servlet es un componente java que puede ser instalado en un servidor para ampliar su funcionalidad. Los servlets usan la API Java Servlet, una extensión estándar de Java. Los servlets pueden ser incluidos en muchos servidores diferentes porque el API no asume nada sobre el entorno o protocolo del servidor.

Los JSP son en realidad servlets. La principal diferencia entre los servlets y los JSP es el enfoque de la programación, un JSP es una página web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página web. Las etiquetas están agrupadas en librerías, las librerías extienden de la especificación de JSP (la cual a su vez extiende de la especificación de Servlet). Su API nos permite además desarrollar nuestras propias librerías de etiquetas.

El controlador de nuestra aplicación será un Servlet y las vistas serán páginas JSP haciendo uso de etiquetas JSP personalizadas.

CSS

Las hojas de estilo en cascada (*Cascading Style Sheets, CSS*) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores.

Las ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.

- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.

- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o la elección del usuario.

- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

JAVASCRIPT

JavaScript es un lenguaje interpretado orientado a las páginas web, con una sintaxis semejante a la del lenguaje Java. El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que fabricó los primeros navegadores de Internet comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Se utiliza en páginas web HTML, para realizar tareas y operaciones en el marco de la aplicación cliente. En nuestro proyecto JavaScript es utilizada en la interfaz de usuario para chequear los campos de los formularios antes de ser enviados al controlador de la aplicación.

SERVIDOR TOMCAT

Tomcat [23] es un contenedor web con soporte de servlets y JSPs desarrollado en la Apache Software Foundation. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. Es un desarrollo de software libre multiplataforma al estar implementado en Java.

El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache, si bien Tomcat puede funcionar como servidor web por sí mismo. En sus inicios, existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción, y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

En nuestro proyecto lo hemos utilizado de forma autónoma como contenedor web para dar servicio a la aplicación basada en JSPs que hemos desarrollado de cara a que los usuarios del sistema puedan indicar qué recursos quieren dar de alta en las ontologías y puedan visualizar los contenidos de las mismas.

3. DISEÑO Y RESOLUCIÓN DEL TRABAJO

3.1 ARQUITECTURA DEL SISTEMA

RICAO v1.0 supone la consecución del objetivo principal que nos planteábamos al principio del proyecto "*diseño e implementación de una arquitectura genérica, modular y multiplataforma capaz de: (1) analizar información ofrecida por determinados recursos de contenidos audiovisuales disponibles en Internet y almacenarla en ontologías, (2) ofrecer un mecanismo de búsqueda de contenidos, cuyos resultados sean la integración de información ofrecida por diferentes ontologías*". Hemos desarrollado una aplicación capaz de almacenar la información ofrecida por tres portales de Internet en tres ontologías independientes, y ofrecer una interfaz de búsqueda cuyos resultados serán la integración de la información de cada una de las ontologías.

Comenzaremos describiendo su arquitectura, exponiendo la distribución de los distintos elementos físicos necesarios para el despliegue y explotación de nuestra herramienta, así como las relaciones existentes entre ellos. Veremos también las distintas aplicaciones en que se divide el sistema, según el papel que el usuario vaya a tomar en ellas, y los módulos de que consta cada una.

El sistema RICAO ha sido implementado utilizando tecnologías de desarrollo basadas en software libre y multiplataforma, pudiendo ser instalado prácticamente en cualquier ordenador actual sin necesidad de pagar ningún tipo de licencia para ello.

Para la primera toma de contacto con el sistema presentamos el siguiente diagrama de bloques que muestra de forma resumida el funcionamiento del sistema.

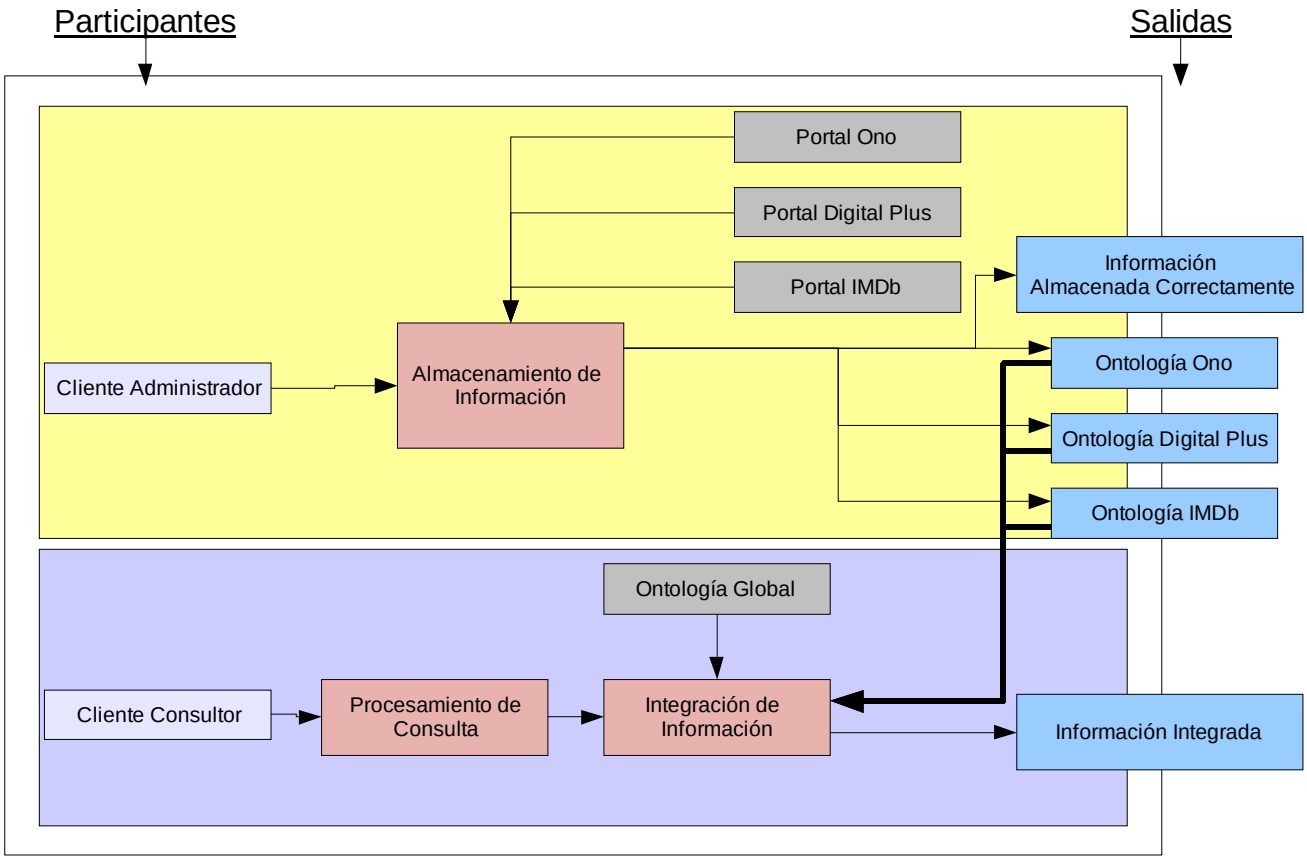


Figura 6. Diagrama de Bloques del Sistema

El sistema depende en gran medida de tres portales de Internet (Ono, Digital Plus, e IMDb), para cada uno de los cuales se ha implementado una ontología. El sistema se puede dividir, a su vez, en dos subsistemas, un primer subsistema llamado “Recuperación de Información” y un segundo subsistema llamado “Integración de Información”. En la figura 6, se pueden observar los dos subsistemas y la relación que hay entre ellos por las ontologías asociadas a los recursos.

El subsistema “Recuperación de Información” será el encargado de realizar un análisis de los documentos HTML ofrecidos por los portales de Internet y almacenarlos en las ontologías asociados a los mismos. El segundo, “Integración de Información”, será el encargado de ofrecer al usuario una interfaz de búsqueda y llevar a cabo las consultas dando como resultado una integración de la información ofrecida por las ontologías utilizadas en el subsistema “Recuperación de Información”.

Se distinguen dos roles en el sistema, el rol “cliente administrador” del subsistema “Recuperación de Información” y el rol “cliente consultor” del subsistema “Integración de Información”. El rol de “clientes administradores” lo desempeñarán aquellos usuarios cuya misión sea la de indicar qué información de los portales almacenar en las ontologías asociadas a ellos. Lo indicarán mediante la inserción de URLs en un formulario de la interfaz de usuario. El rol de “cliente consultor” lo desempeñarán los usuarios que, a través de la interfaz de la aplicación, realicen búsquedas en el sistema. A continuación se explica de forma detallada cada uno de los subsistemas.

SUBSISTEMA “RECUPERACIÓN DE INFORMACIÓN”

Han sido utilizados tres portales de Internet que ofrecen información acerca de cine y de programación de televisión, Ono, Digital Plus, e IMDb. La información ofrecida por cada uno de los portales será almacenada de forma independiente, para ello han sido implementadas tres ontologías diferentes e independientes. Puesto que la información ofrecida por cada uno de los portales está en formato HyperText Markup Language (HTML) será necesario realizar un análisis para poder almacenarla en la ontología.

En el siguiente diagrama de despliegue de UML (ver figura 7), podemos percibir a grandes rasgos la arquitectura del subsistema “Recuperación de Información”. En el diagrama se muestra como la aplicación “ModuloWeb.war” es desplegada en un servidor “J2EEServer” al cual puede acceder un usuario “Cliente administrador” a través de un “Navegador Web” convencional. La aplicación accederá a los portales de contenidos audiovisuales “Portal Ono”, “Portal Digital Plus” y “Portal IMDb” para la recuperación de información y generación automática de instancias en las ontologías. Por último la aplicación mostrará los resultados de la generación de instancias en “Paginas JSP”.

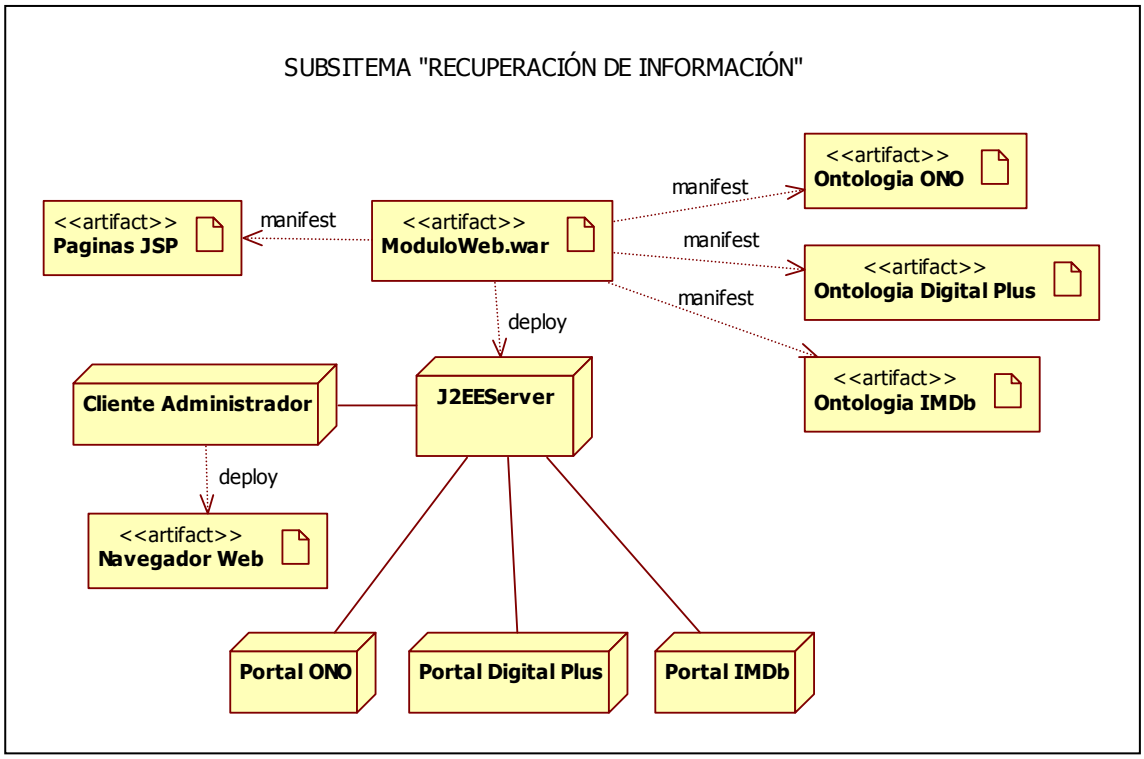


Figura 7. Subsistema "Recuperación de Información"

SUBSISTEMA "INTEGRACIÓN DE INFORMACIÓN"

Un usuario puede acceder al sistema y realizar una búsqueda de contenidos. El resultado será una página con el resultado de la integración de la información de cada una de las ontologías. Internamente se realiza una consulta SPARQL a cada una de las ontologías y se integra el resultado obtenido almacenándolo en una instancia de la ontología global.

En el siguiente diagrama de despliegue de UML (ver figura 8), podemos percibir a grandes rasgos la arquitectura del subsistema "Integración de Información". En el diagrama se muestra como la aplicación "ModuloWeb.war" es desplegada en un servidor "J2EEServer" al cual puede acceder un usuario "Cliente Consultor" del subsistema a través de un "Navegador Web" convencional. La aplicación realizará consultas a las ontologías "Ontología Ono", "Ontología Digital Plus" y "Ontología IMDb" para obtener la información de la consulta realizada e integrarla posteriormente en una instancia de la

"Ontología Global". Por último la aplicación mostrará los resultados de la integración en "Paginas JSP".

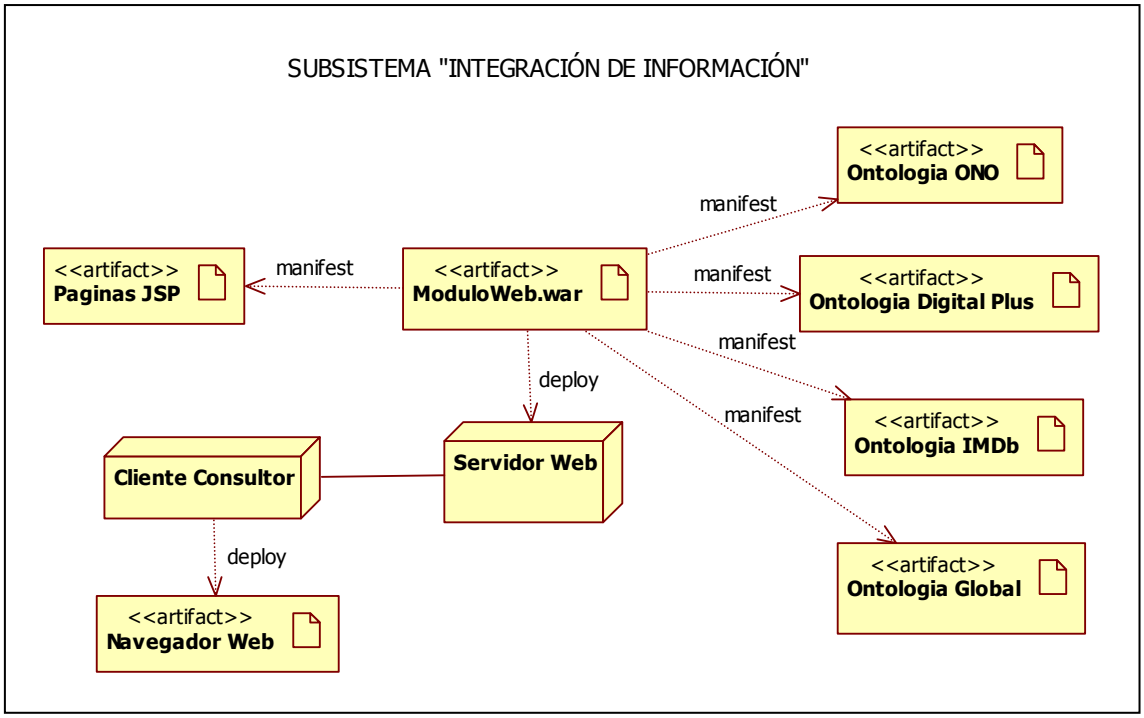


Figura 8. Subsistema "Integración de Información"

Para que los clientes puedan hacer uso de la aplicación no es necesario que instalen nada en su ordenador (suponiendo que disponga ya de un navegador web), y le bastaría con acceder al servidor web donde habríamos desplegado la aplicación web (ModuloWeb.war). La aplicación es la encargada de acceder a las ontologías. Para la correcta ejecución de la aplicación web, es necesario que el servidor web soporte servlet y Java Server Page (JSP), cosa que ocurre con el servidor Apache Tomcat, que como ya expusimos anteriormente, es el que elegimos para nuestro desarrollo, si bien cualquier otro servidor compatible resultaría perfectamente válido sin necesidad de realizar cambios en la aplicación.

CONTROLADOR

El controlador se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, y con el modelo, para acceder a las ontologías.

Para la implementación del controlador se ha hecho uso del patrón FrontController. Todas las clases asociadas al patrón y al controlador están incluidas en el paquete "controlador" y sus paquetes anidados correspondientes, que son "controlador.acciones", "controlador.acciones.digital", "controlador.acciones.ono", "controlador.acciones.imdb" y "controlador.acciones.global". A continuación se presenta un diagrama de clases del patrón FrontController aplicado a nuestra aplicación.

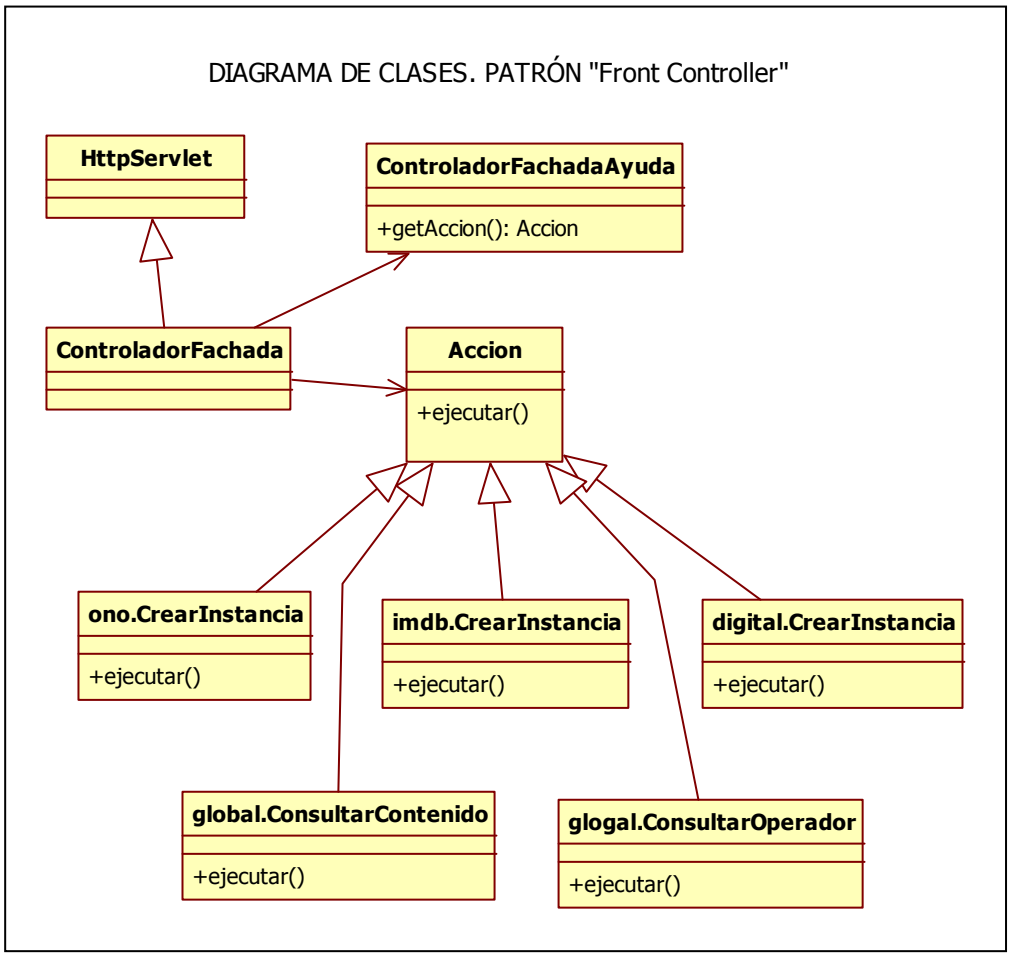


Figura 9. Diagrama de Clases. Patrón "Front Controller"

Las clases más importantes utilizadas en el controlador son tres, la clase que representa al Servlet “ControladorFachada”, la clase ControladorFachadaAyuda, utilizada por el Servlet para identificar la clase de la acción concreta que se debe ejecutar y la clase abstracta “Accion” que deben de heredar todas las clases que representen acciones concretas y definir su método abstracto “ejecutar()”.

La clase “ControladorFachada” es el único servlet utilizado en la aplicación, definido en el fichero “web.xml”. El servlet “ControladorFachada” recibe todas las peticiones de la interfaz de usuario que tengan el patrón “*.ctrl”. La clase “ControladorFachadaAyuda” hace uso del fichero de propiedades “accion.properties” para identificar la clase de la acción concreta a instanciar.

Las clases que representan las acciones son las responsables de seleccionar las vistas que serán mostradas al usuario. A continuación se muestra el diagrama de secuencias del patrón FrontController utilizado.

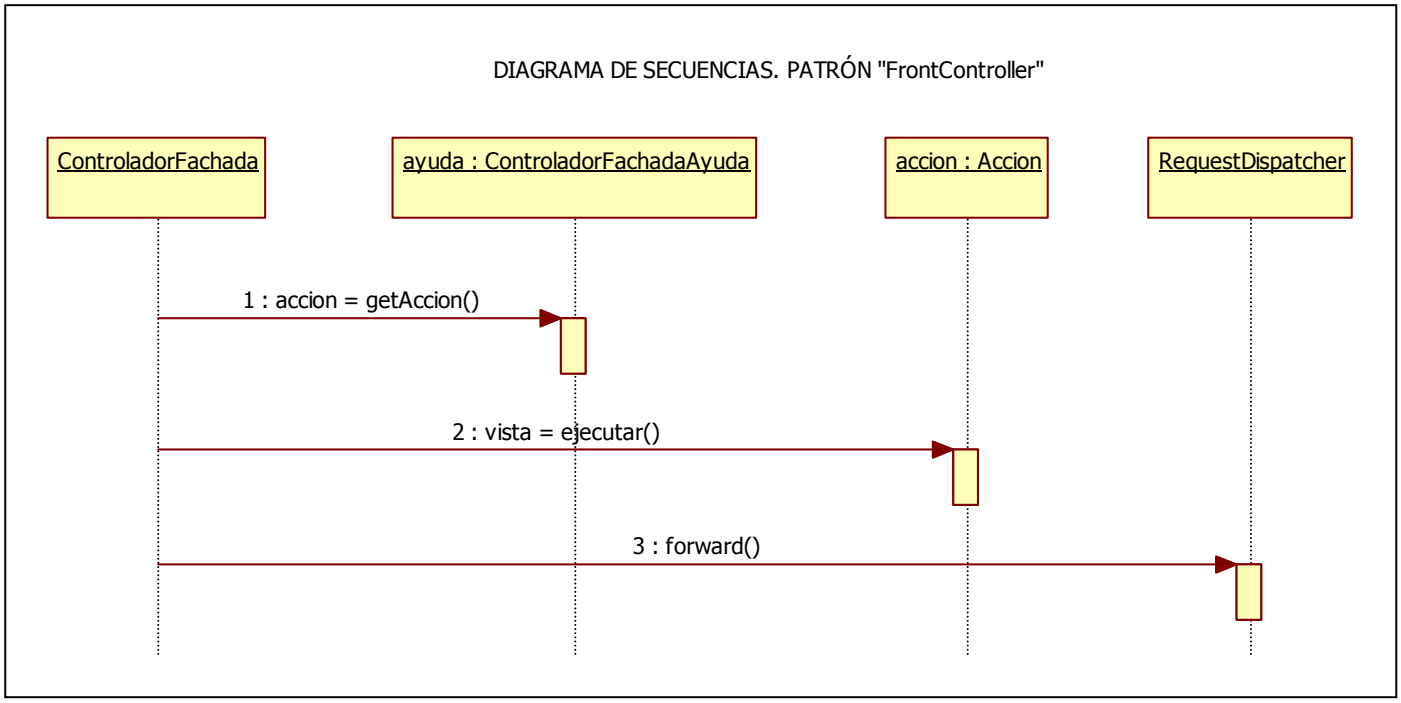


Figura 10. Diagrama de Secuencias. Patrón “Front Controller”

MODELO

Como hemos mencionado previamente el sistema puede subdividirse en 2 subsistemas. Aunque la vista y el controlador tengan la misma estructura en ambos subsistemas el modelo es diferente. Ambos modelos hacen uso de la API JENA, sin embargo para el acceso a las ontologías en el subsistema "Integración de información" se hace uso del patrón DAO y para el subsistema "Recuperación de información" no.

La aplicación hace uso de la API Jena para el acceso a las ontologías, pero podría hacer uso de otra API diferente. Para conseguir esto se debe aislar la aplicación de la tecnología de persistencia haciendo que el modelo no tenga conocimiento directo del destino final de la información que manipula. El patrón DAO "Data Access Object" ha sido usado en el subsistema "Integración de Información" para este propósito. La flexibilidad tiene un precio y al hacer uso del patrón DAO en la aplicación añade una capa adicional de persistencia incrementando la cantidad de código ejecutado en tiempo de ejecución. En el subsistema "Recuperación de información" la eficiencia debe ser tenida en cuenta puesto que deben de darse de alta miles de páginas HTML. Es por ello que no ha sido utilizado el patrón DAO.

Modelo del Subsistema "Recuperación de Información".

El modelo del subsistema "Recuperación de Información" está definido en el paquete "mapeo". Incluye la lógica necesaria para realizar el análisis y la correspondencia de los documentos HTML ofrecidos por los portales (Ono, Digital Plus, IMDb) a las ontologías asociadas a ellos. Se hace uso de la API Jericho-Html-2.4 para el análisis de las páginas HTML y de la API Jena-2.5.4 para el acceso a la ontologías.

Modelo del Subsistema "Integración de Información".

En el modelo del subsistema "Integración de Información" se hace uso del patrón DAO. Las principales clases y paquetes de clases involucrados en el patrón DAO son:

1) “dao.FactoriaDAO”. Clase abstracta que define una Factoría Abstracta. En esta clase hay definido un método que devuelve una factoría concreta según el parámetro que se le pase. En nuestro proyecto solo hay disponible un tipo de factoría que define una factoría de acceso a Ontologías que hará uso de la API Jena-2.5.4.

2) “dao.WebSemantica.FactoriaDAO”. Representa a la factoría DAO concreta. Define los métodos abstractos de “Factoria DAO”.

3) “dao.interfaces”. Se disponen de tantas interfaces como elementos DAO existen, es decir, ocho (Canal, Contenido, Emisión, Operador, Persona, Premio, Subtitulo, Versión). Cada una de estas interfaces define los métodos asociados al elemento DAO en cuestión. Cualquier factoría concreta puede implementar estas interfaces y disponer de los mismos métodos. El objetivo es que el código usado sea reutilizable y solo cambie la forma de acceder a los datos, es decir, solo cambie la forma de implementar los métodos de estas interfaces.

A continuación (ver figura 11) se muestra el diagrama de clases del patrón DAO usado en nuestra aplicación.

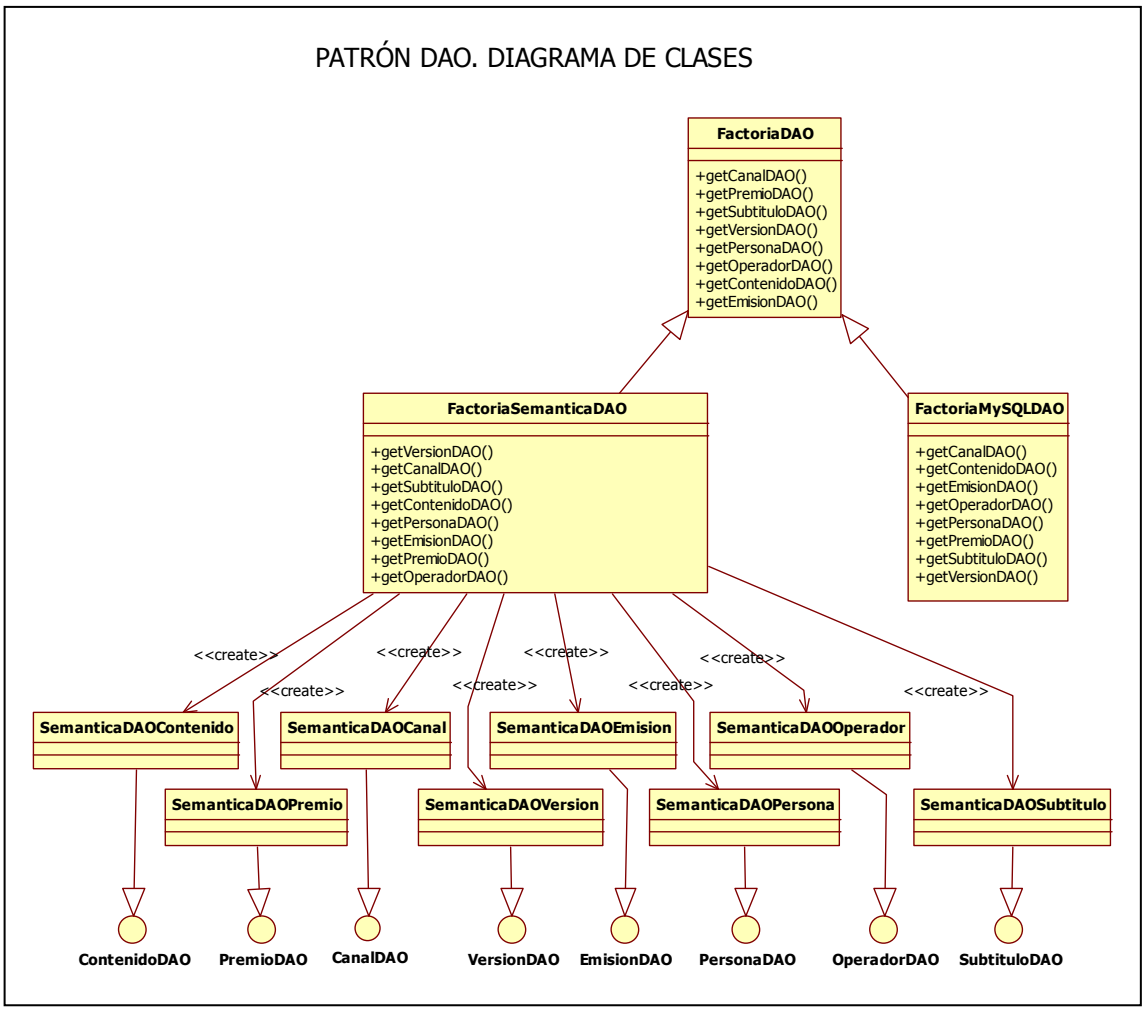


Figura 11. Diagrama de Clases. Patrón DAO

VISTA

La vista es la encargada de presentar los datos al usuario mediante el despliegue de una interfaz de usuario. Los datos son proporcionados por el modelo. Está formada por un conjunto de páginas JSP y etiquetas JSP. Todas las etiquetas están disponibles en el paquete “etiquetas” y están definidas en el mismo fichero “contenido.tld”. Todas ellas tienen los/el mismo/s objetivo/s:

1) Imprimir la información ofrecida por una ontología.
2) Imprimir la información asociada a un individuo de una ontología.

La vista es parte de la interfaz de usuario. Sin embargo, la interfaz de usuario es un concepto más amplio que además de la vista abarca páginas

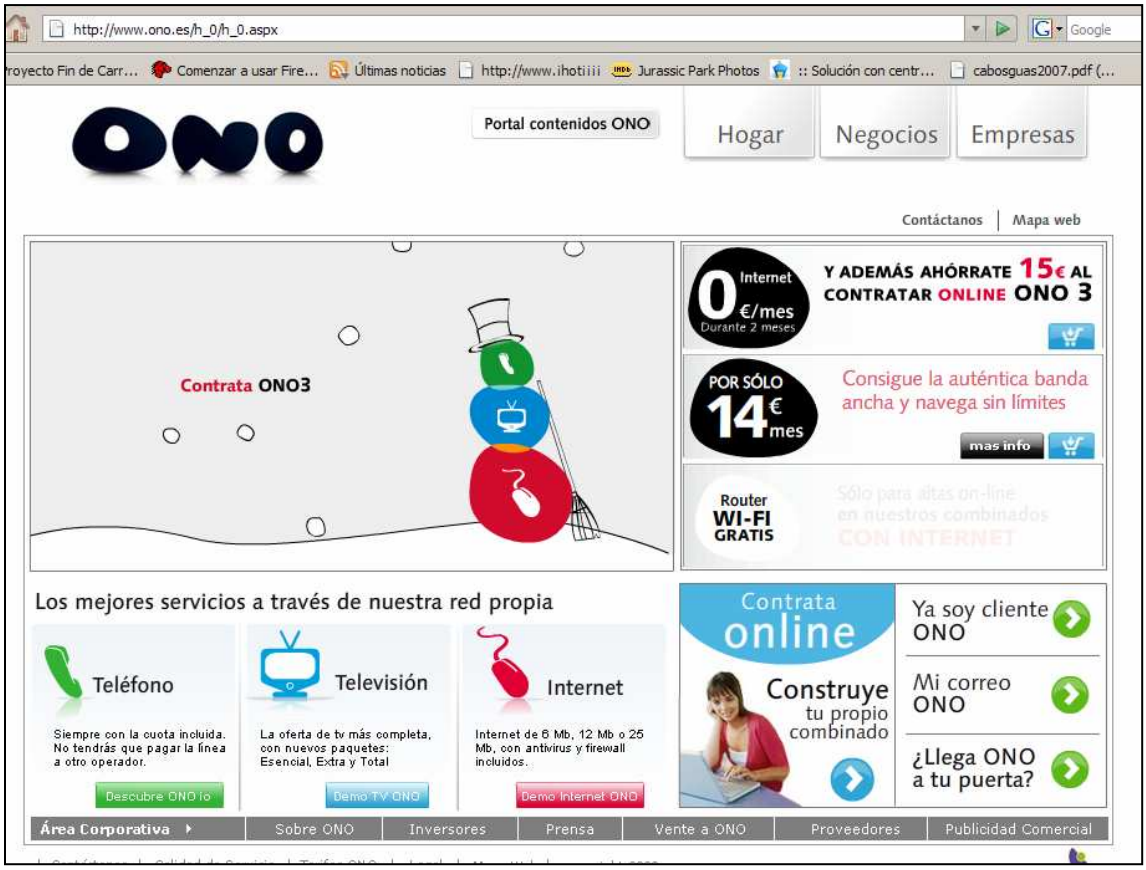
HTML, ficheros JavaScript y ficheros de Cascading Style Sheets (CSS). El uso de JavaScript es para evitar el envío de peticiones incorrectas por parte del usuario, que impide que un formulario sea enviado si los campo son incorrectos y el uso de ficheros CSS para el diseño visual de la interfaz.

3.2 MODELADO DE RECURSOS: E/R Y ONTOLOGÍA DE RECURSO

El proyecto se ha basado en la información ofrecida por tres portales de Internet, dos de ellos ofrecen información acerca de la programación de televisión ofrecida por el operador asociado al portal, y otro ofrece información relativa a películas y series de cine y televisión.

1) ONO [10]. Ofrece información acerca de la programación de televisión de los canales ofrecidos por el operador ONO. La información ofrecida es relativa a la comunidad autónoma del operador, es decir, la programación de Ono en Murcia no tiene por qué ser la misma que la de Madrid.

Figura 12. Portal de contenidos audiovisuales ONO



2) Digital Plus [2]. Información acerca de la programación de televisión de los canales ofrecidos por el operador Digital Plus.

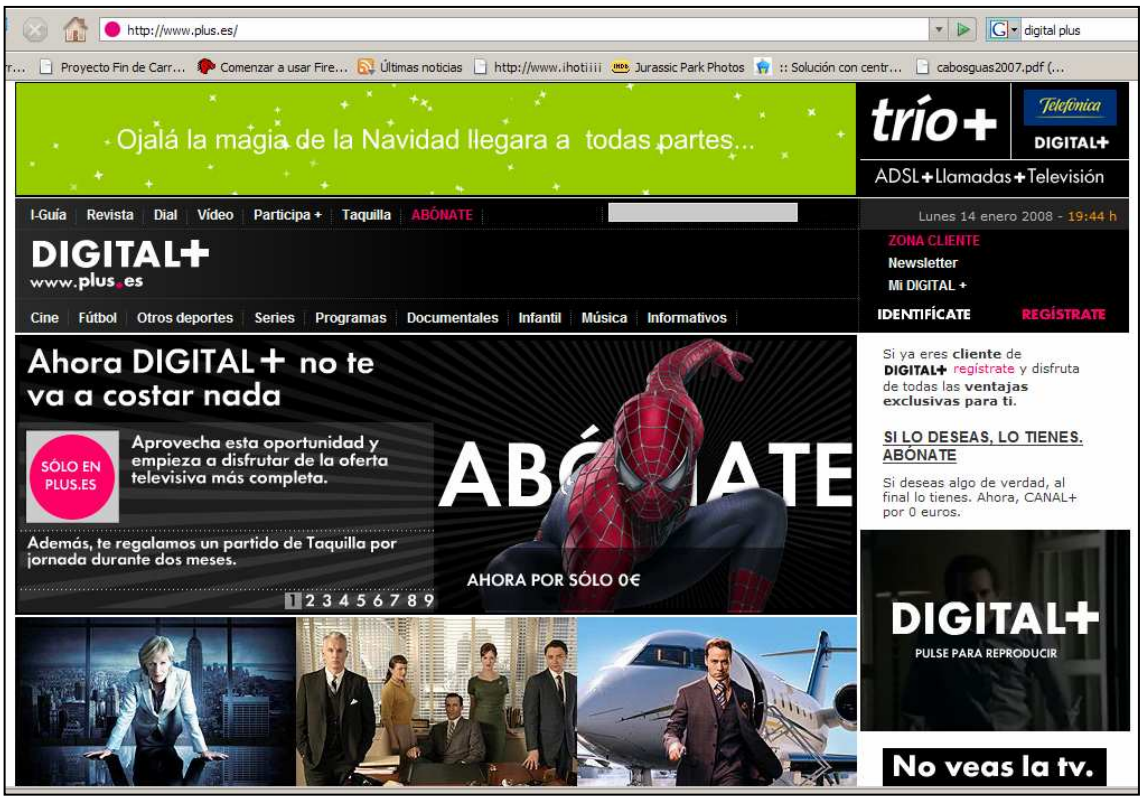


Figura 13. Portal de contenidos audiovisuales Digital Plus

3) The Internet Movie Database (IMDb) [5]. Portal que ofrece abundante información sobre contenidos audiovisuales de todas las nacionalidades.

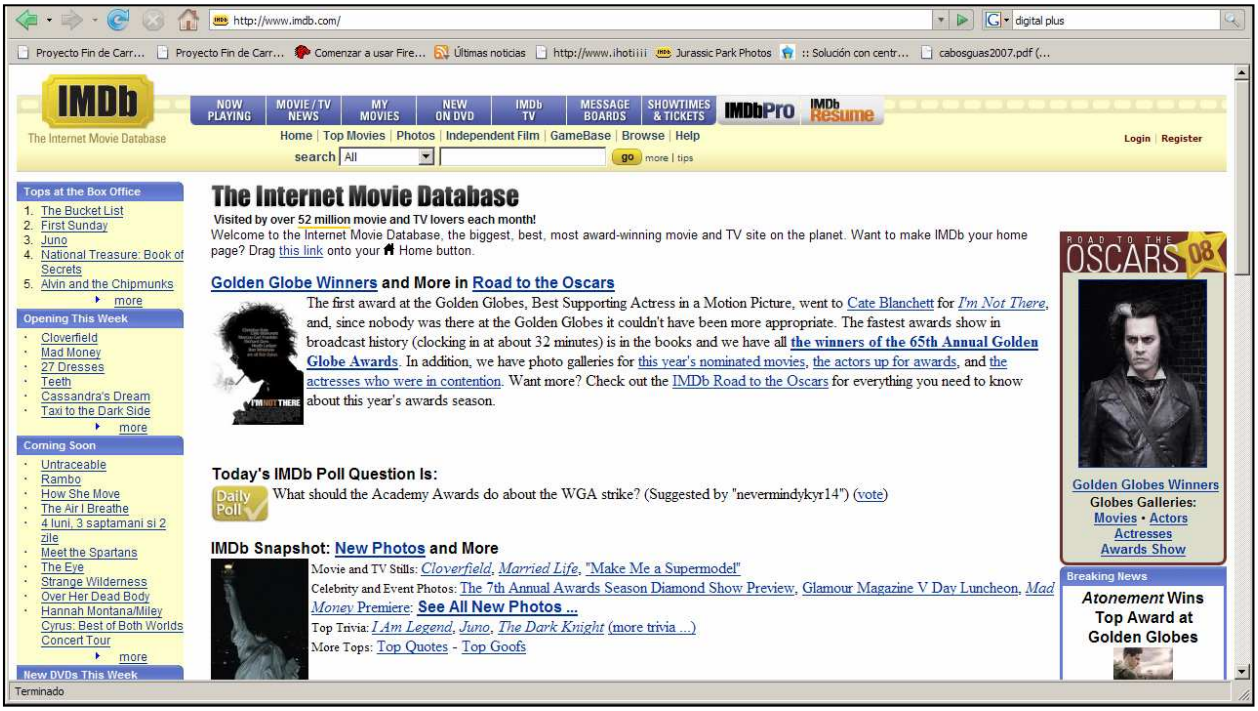


Figura 14. Portal de contenidos audiovisuales IMDb

Para cada uno de estos portales en los que se ha basado el proyecto ha sido necesaria la elaboración de un modelo conceptual. Para su desarrollo han sido utilizados modelos entidad-relación con la notación MPM 1999 [15] [27].

Los nombres de los atributos y entidades han sido nombradas tal y como aparecen en los portales, de este modo se da el caso de que un atributo con el mismo significado semántico en los portales sea nombrado de forma diferente en los modelos conceptuales. En otros casos el nombre no se especificaba, siendo necesaria la creación de nombres intuitivos.

Estos modelos conceptuales han sido utilizados como base para la construcción de cada una de las ontologías asociadas a los recursos.

MODELO ENTIDAD RELACIÓN
NOTACIÓN [MPM1999]
PORTAL ONO

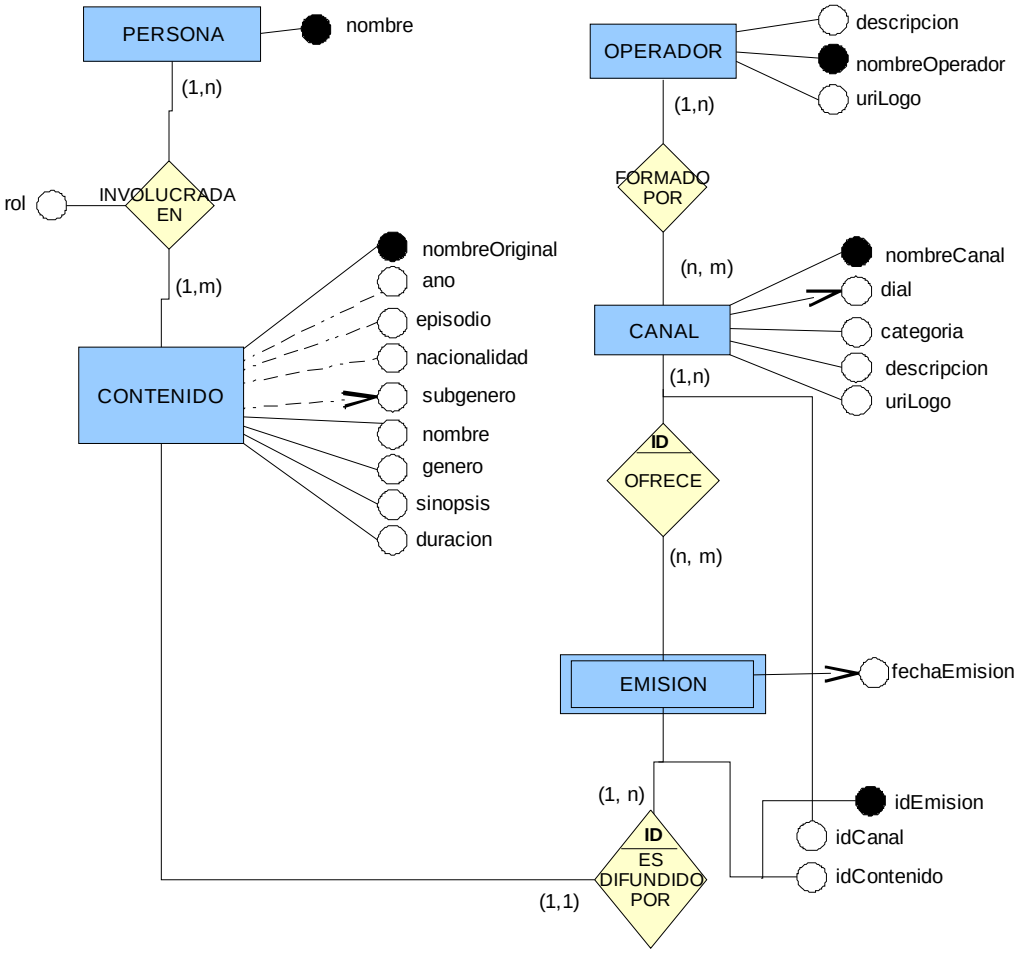


Figura 15. Modelo Entidad Relación. Portal ONO

MODELO ENTIDAD RELACIÓN
NOTACIÓN [MPM1999]
PORTAL DIGITAL PLUS

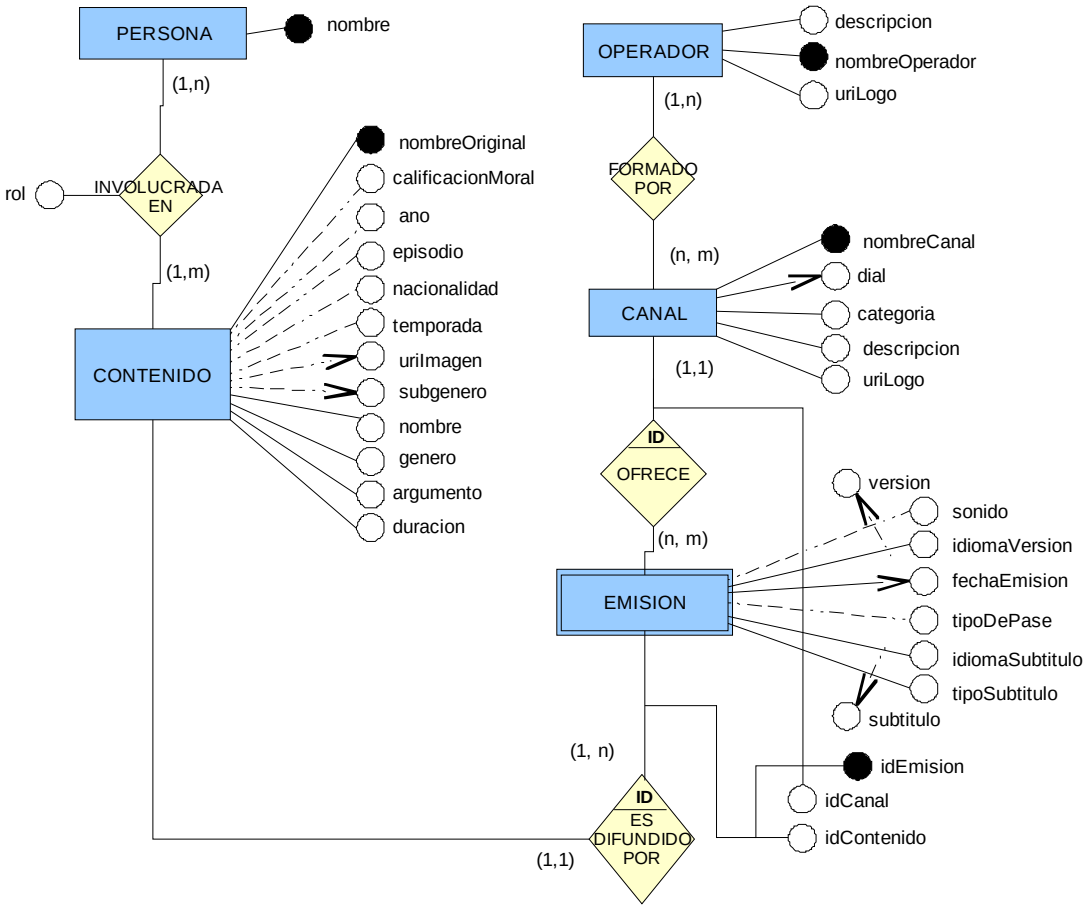


Figura 16. Modelo Entidad Relación. Portal Digital Plus

MODELO ENTIDAD RELACIÓN
NOTACIÓN [MPM1999]
PORTAL IMDb

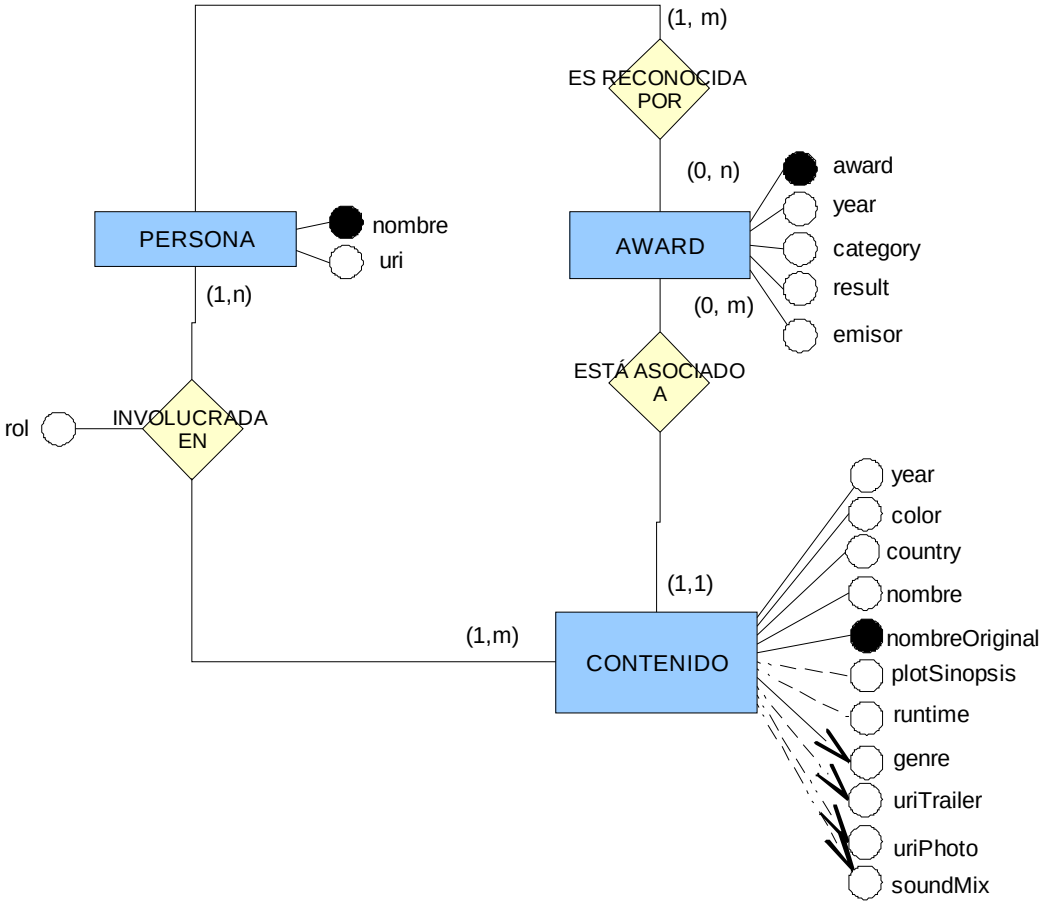


Figura 17. Modelo Entidad Relación. Portal IMDb

A partir de los tres modelos creados anteriormente se ha construido un nuevo modelo conceptual integrador a partir del cual se construirá la ontología integradora.

MODELO ENTIDAD RELACIÓN
NOTACIÓN [MPM1999]
INTEGRACIÓN DE MODELOS

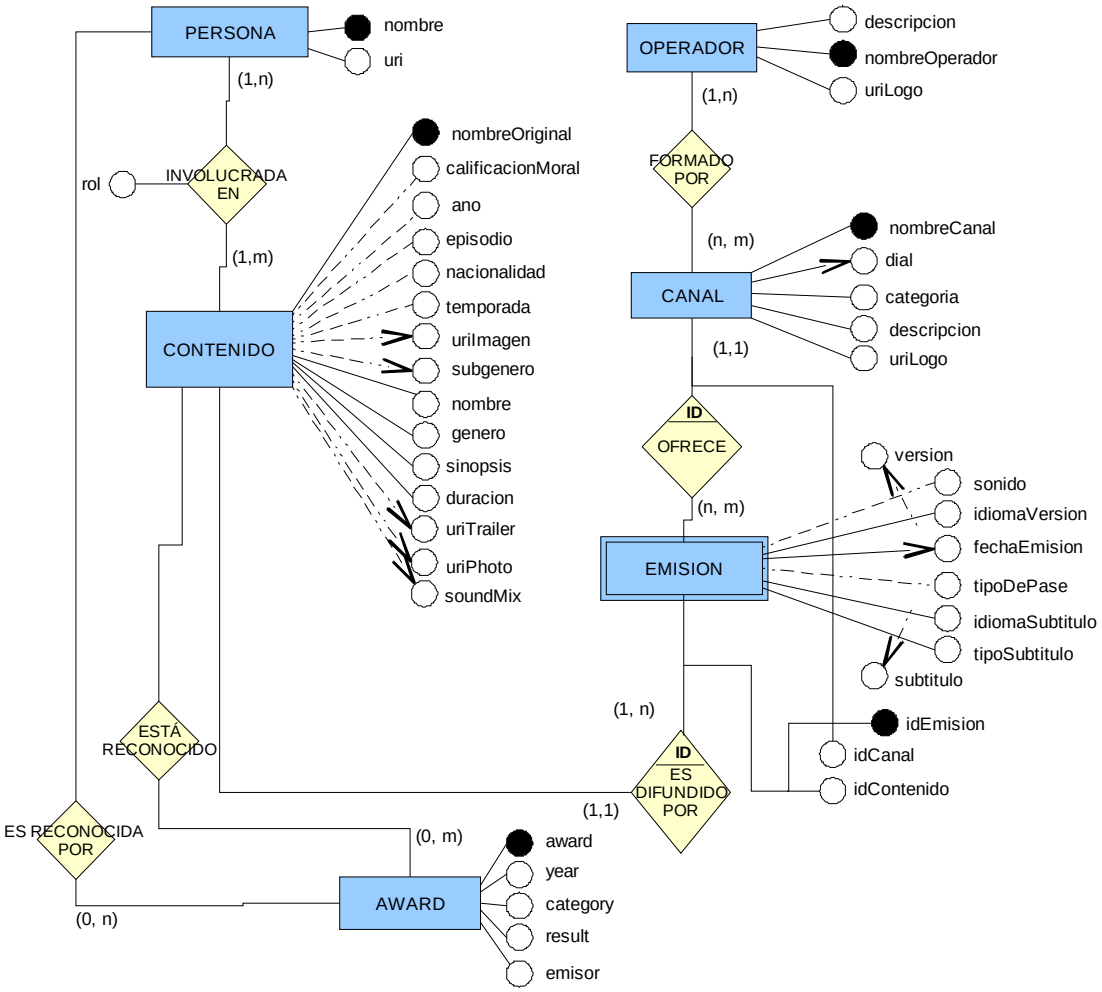


Figura 18. Modelo Entidad Relación Integrador

Se han implementado cuatro ontologías, una para cada uno de los portales de Internet que estamos utilizando, que llamaremos ontologías parciales, y otra resultado de la unión de las otras tres, que llamaremos ontología global. El contenido almacenado en las ontologías parciales será el resultado del almacenamiento previo de información realizado por el subsistema “Recuperación de Información” cuyo objetivo era analizar la información de los documentos HTML proporcionados por los portales y almacenarla en las ontologías. La ontología global es utilizada para la creación de instancias resultado de la integración de la información proporcionada por cada una de las ontologías parciales.

Las ontologías se han implementado haciendo uso del editor de ontologías “Protégé” y basándose en los modelos entidad-relación vistos anteriormente. Los atributos, relaciones, entidades y restricciones de cardinalidad proporcionados por los modelos entidad-relación han sido incluidos en las ontologías mediante sus correspondientes datatype properties, object properties, clases y restricciones de cardinalidad. En las siguientes figuras (figura 20, 21, 22 y 23) podemos ver las ontologías correspondientes a los modelos entidad-relación vistos anteriormente. Los diagramas han sido creados con el plugin “Ontoviz” de Protégé. Se muestran las clases, atributos y relaciones, faltarían las restricciones de cardinalidad que han sido omitidas para no hacer demasiado grandes y complejos los diagramas.

Los diagramas están formados por tablas (ver figura 19) estructuradas de la siguiente forma. La primera fila indica el nombre de la clase y las siguientes muestran los atributos y las relaciones. Por ejemplo, la tabla de la figura 19 representa a la clase “EMISION” y tiene un atributo llamado “fechaEmisión” y dos relaciones, una llamada “emision_ofrecidaPor_canal” que relaciona una instancia de “EMISION” con “CANAL” y otra llamada “emision_difunde_contenido” que relaciones una instancia de tipo “EMISION” con otra de tipo CONTENIDO.

EMISION		
fechaEmision	String*	
emision_ofrecidaPor_canal	Instance*	CANAL
emision_difunde_contenido	Instance*	CONTENIDO

Figura 19. Clase Emision

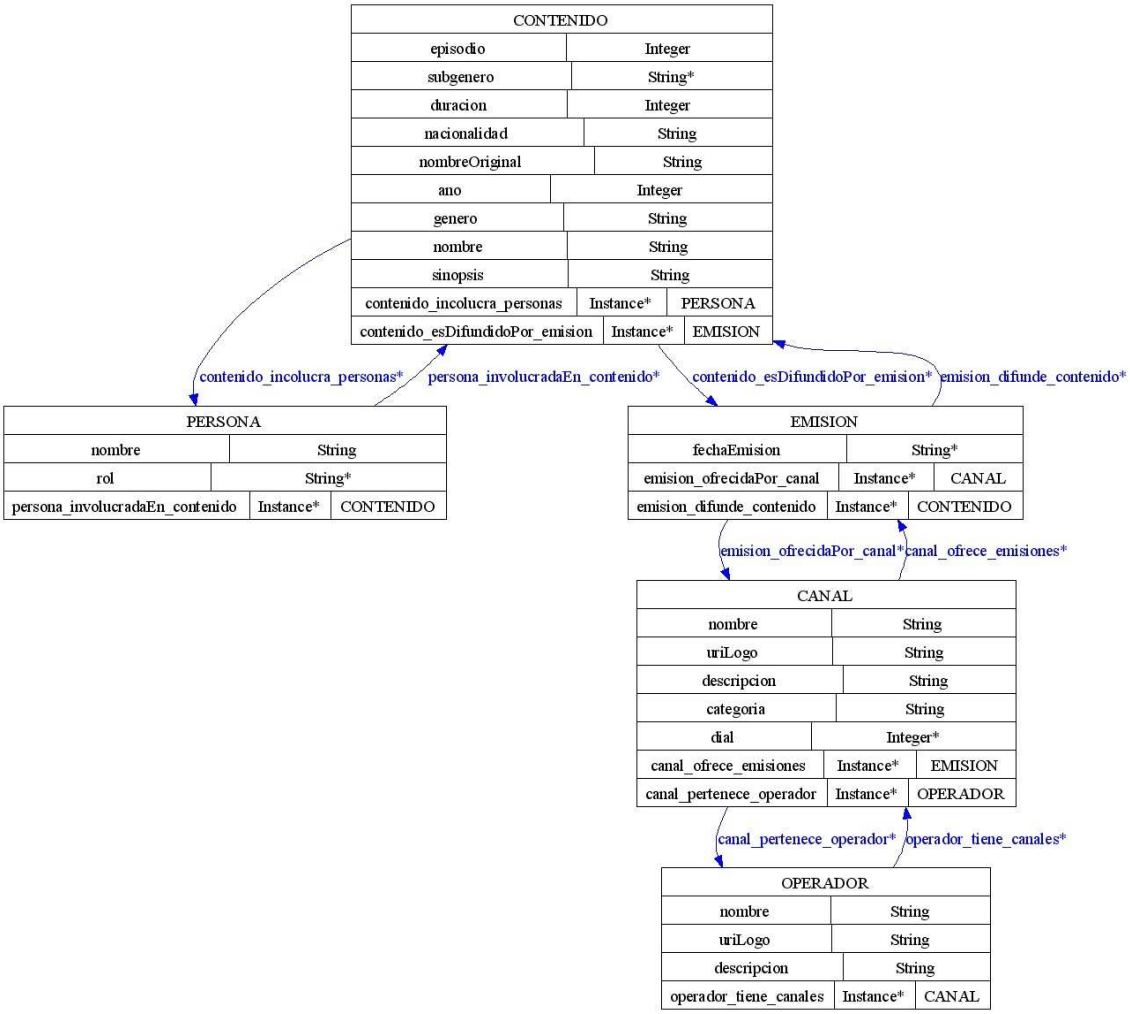


Figura 20. Ontología ONO

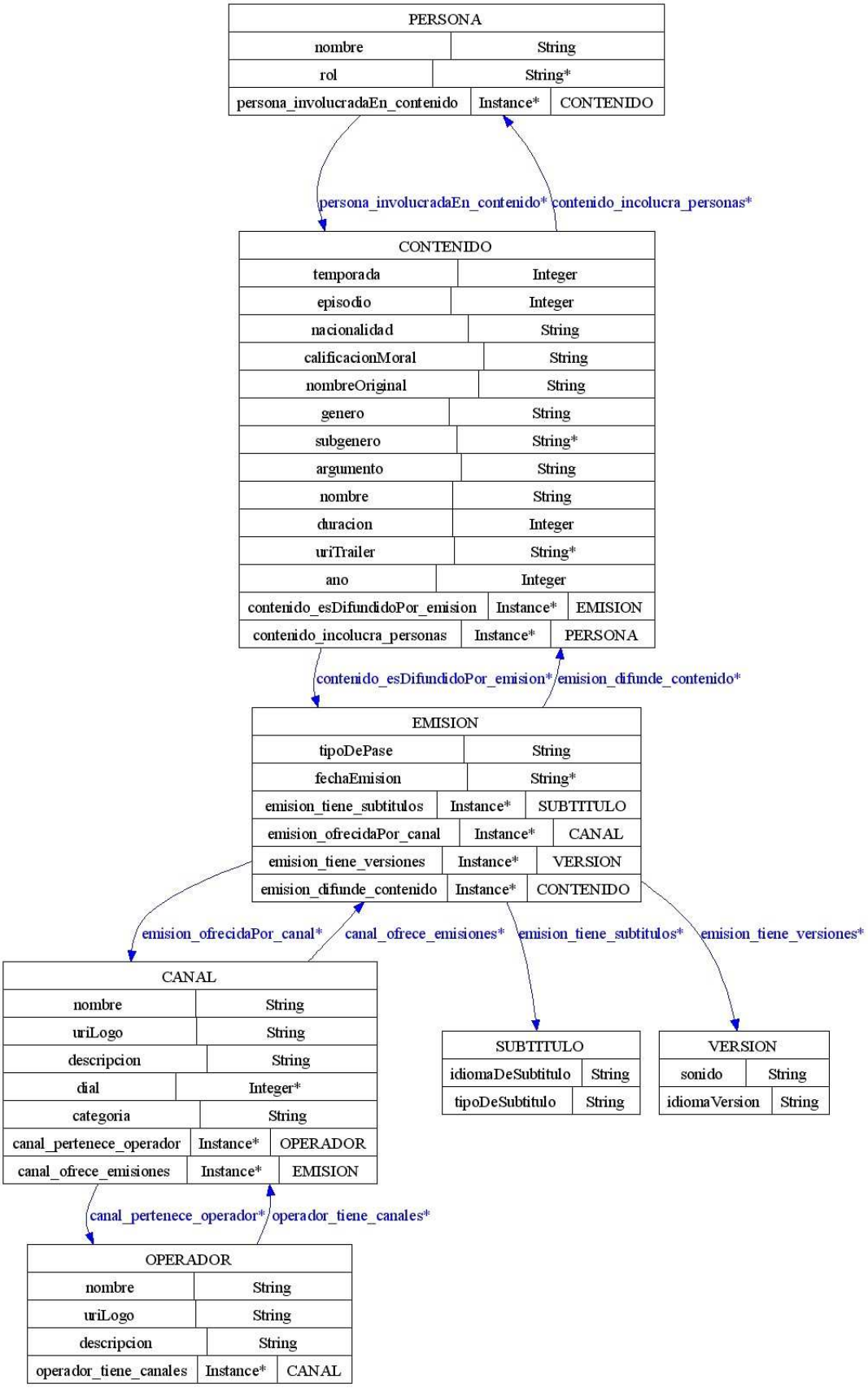


Figura 21. Ontología Digital Plus

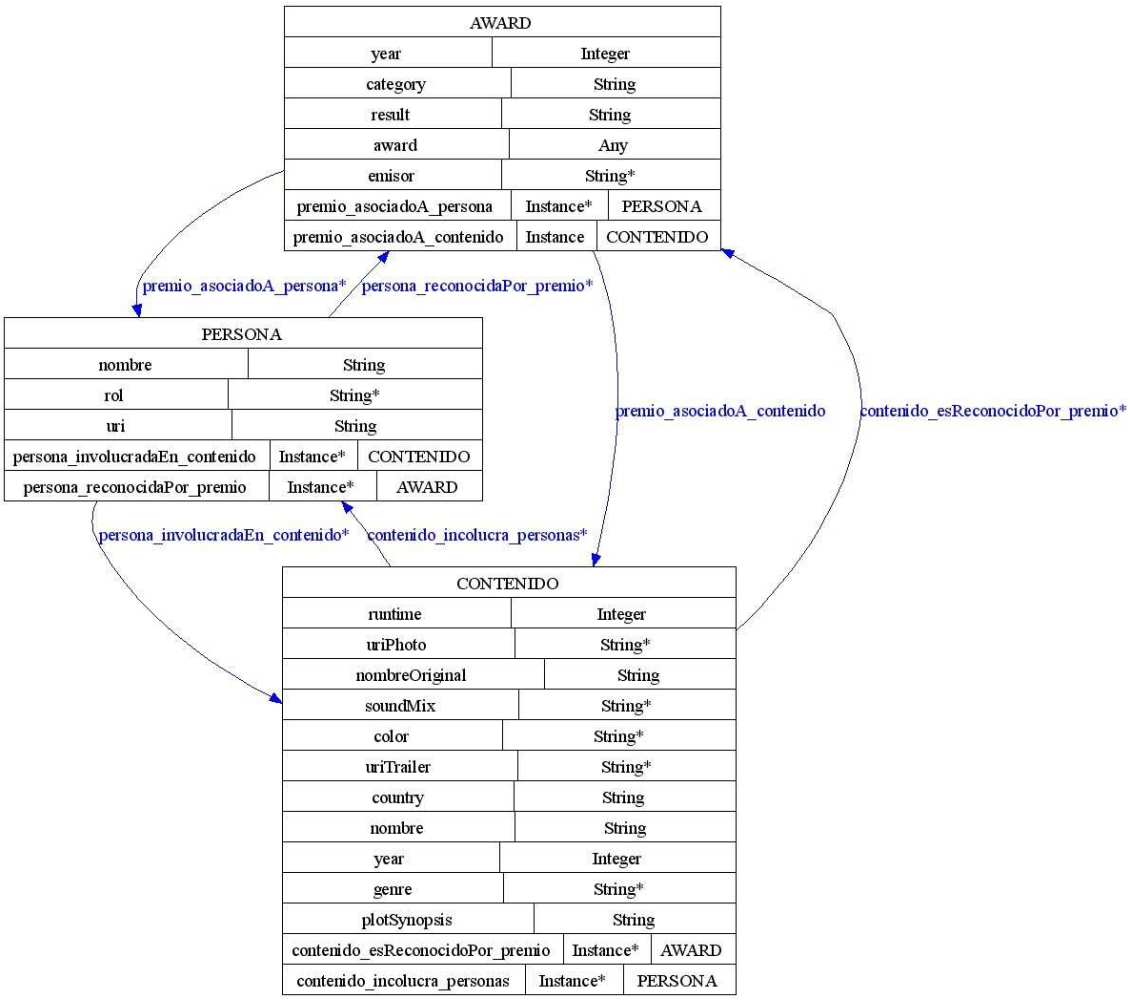


Figura 22. Ontología IMDb

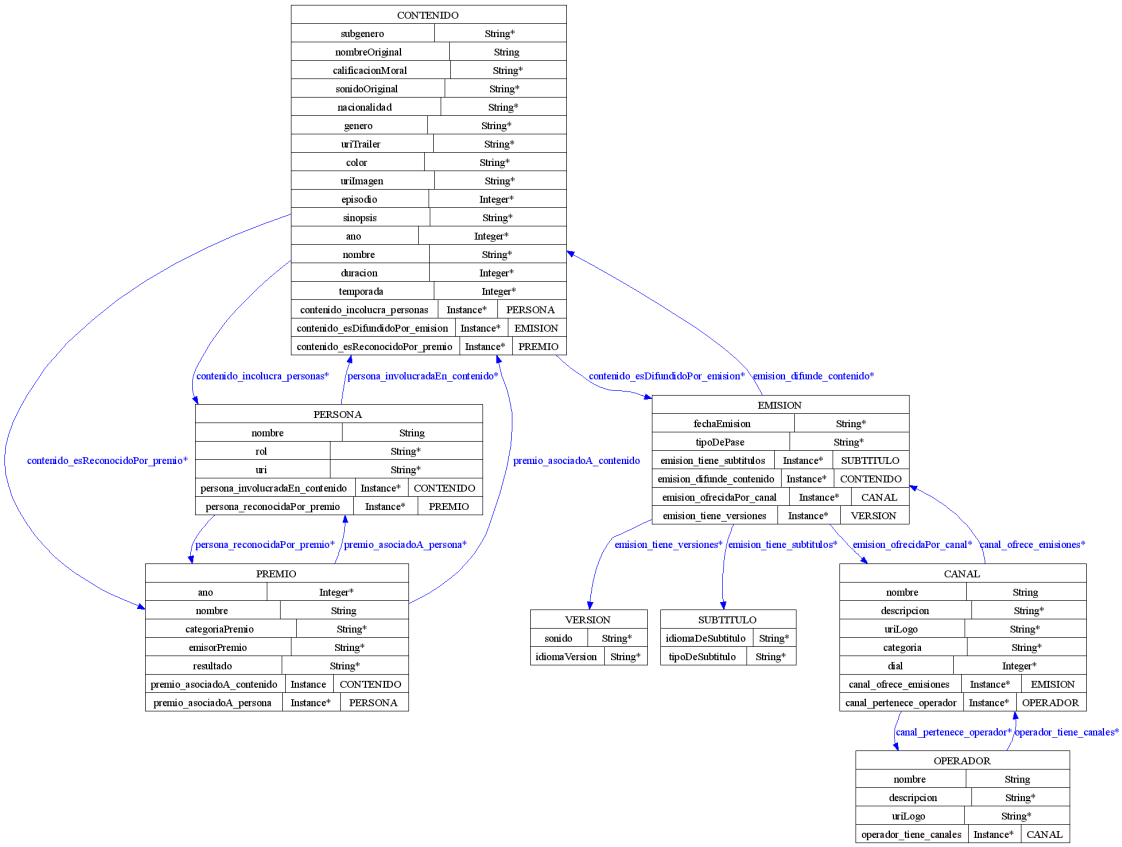


Figura 23. Ontología Global

La correspondencia entre las entidades de los modelos entidad-relación y las clases se muestran a continuación. La correspondencia entre las entidades del modelo entidad-relación de ONO y Digital Plus con sus correspondientes ontologías son las mismas:

ENTIDADES	CLASES
Persona	Persona
Contenido	Contenido
Operador	Operador
Canal	Canal
Emision	Emision
	Version
	Subtitulo

La correspondencia entre las entidades del modelo entidad-relación de IMDb con su correspondiente ontologías aparecen en la siguiente tabla.

ENTIDADES	CLASES
Persona	Persona
Award	Award
Contenido	Contenido

Como se puede observar en las tablas anteriores hay una correspondencia 1 a 1 entre las entidades y las clases pero no al revés, por ejemplo las clases “Subtitulo” y “Version” no se corresponden con entidades del modelo entidad-relación.

Cuando se realiza una consulta en el subsistema “Integración de Información” internamente se llevan a cabo tres consultas SPARQL, una para cada una de las ontologías parciales, y se integra la información en una instancia de la ontología global. Si incluyéramos todas las restricciones de cardinalidad en la ontología global la instancia generada podría tener un estado inconsistente puesto que la consultas realizadas ofrecen información parcial de

la ontología, por este motivo la en la Ontología global implementada se han omitido las restricciones de cardinalidad en la mayoría de los atributos y relaciones.

3.3 GENERACIÓN DE INSTANCIAS DE CONTENIDOS A PARTIR DE DOCUMENTOS HTML

La forma de analizar la información ofrecida por los portales utilizados ha estado determinada por el formato de la información. En los diferentes portales encontramos código HTML de diferentes versiones.

En el portal Digital Plus encontramos en la cabecera de los documentos una referencia al DTD (Definición de Tipo de Documentos) de XTHML 1.O Transitional. Este tipo de documentos se utiliza cuando no se describe la presentación de los documentos por medio de hojas de estilo en cascada (CSS). Este no es el caso puesto que en este portal se hace uso de CSS. En teoría este tipo de documentos podría ser procesado y analizado con la API JAXP (Java API for XML Processing). Sin embargo, al hacer uso de esta tecnología se observó que se daban excepciones en el análisis. Para comprobar la consistencia de los ficheros ofrecidos por el portal con XTHML 1.0 se hizo uso del validador “Markup Validation Service” [31] ofrecido por la W3C, encontrándose una media de 75 errores por página. Esto significa que aunque haya sido incluido el DTD en sus documentos, éstos no han sido validados correctamente por el DTD incluido en la cabecera.

En los portales de ONO e IMDb se hace uso de HTML convencional sin indicar DTD en su cabecera. Hemos realizado pruebas incluyendo en las cabeceras una referencia al DTD de XTHML para comprobar si se puede analizar con JAXP encontrando de nuevo errores en ambos portales. Puesto que en los documentos de los portales los documentos o bien, no hacen uso de XHTML, o bien tienen errores, se ha optado por usar un analizador HTML convencional. El analizador seleccionado ha sido “Jericho” ya que es una librería de código abierto que tiene la gran ventaja de poder ignorar el código incorrecto HTML a diferencia de los analizadores basados en la construcción de árboles, como DOM (Document Object Model) de JAXP, que deben identificar cada nodo en el documento de arriba abajo. Jericho, comparado con el analizador basado en eventos SAX (Simple API for XML) de JAXP, ofrece una interfaz más intuitiva y de más alto nivel adaptada a documentos HTML.

El análisis de los documentos se ha hecho en base a las etiquetas encontradas, en muchos casos de las etiquetas “div” ofrecidas por el uso de CSS. Se ha realizado una implementación genérica basada en el patrón estructural de los documentos ofrecidos por los portales. Esto significa que, en caso de que el portal decida cambiar la estructura de sus documentos, el subsistema “Recuperación de Información” debería ser modificado para su correcto funcionamiento. Es importante tener en cuenta que el subsistema “Recuperación de Información” ha sido diseñado para realizar un mapeo o transición de la información HTML ofrecida por los portales a las ontologías, dejando de ser útil una vez llevado a cabo su objetivo.

La información obtenida a partir del análisis de los documentos ha sido almacenada en las ontologías haciendo uso de la API Jena [6]. Para ello es necesario conocer la ubicación de las ontologías y el nombre de las clases, de las propiedades de tipo de datos y de las propiedades de tipo objeto. La ubicación y los prefijos de todas las ontologías están indicados/definidos en el fichero “urisOntologias.properties”. El nombre de las clases, las propiedades de tipo de datos y de las propiedades de objeto están definidos como variables constantes en la clase correspondiente, de forma que si modifica algún nombre en la ontología solo sea necesario cambiar el valor de la constante.

3.4 DEFINICIÓN DE LAS CORRESPONDENCIAS ENTRE LOS RECURSOS

La ontología global es la ontología resultante de la unión de las ontologías parciales. Semánticamente, las ontologías parciales comparten conceptos, atributos y relaciones equivalentes pero cuyo nombre difiere entre ellas. Para poder integrar la información de las tres ontologías parciales es necesario saber la correspondencia entre ellas. Para ello se realiza un mapeo entre la ontología global y cada ontología parcial, es decir, saber qué clase, atributo o relación de la ontología global se corresponde con qué clase, atributo o relación de cada una de las ontologías parciales.

Para el mapeo de ontologías se ha hecho uso de la herramienta “Ontology Mapping” [21], herramienta desarrollada en el grupo de investigación con el que hemos colaborado en el proyecto. Haciendo uso de esta herramienta hemos generado un fichero XML para cada una de las correspondencias Global/Ono, Global/Digital_Plus y Global/IMDb. Para poder crear los ficheros de correspondencia ha sido necesaria la creación de otra ontología que defina el concepto de mapeo entre ontologías.

Esta ontología está formada por seis clases. Tres clases son utilizadas para definir el concepto de mapeo entre atributo, concepto y relación y otras tres para definir la correspondencia. La jerarquía se clases se muestra en el siguiente diagrama, obtenido haciendo uso del plugin “Jambalaya” disponible en la versión de 3.4 beta de “Protegé”.

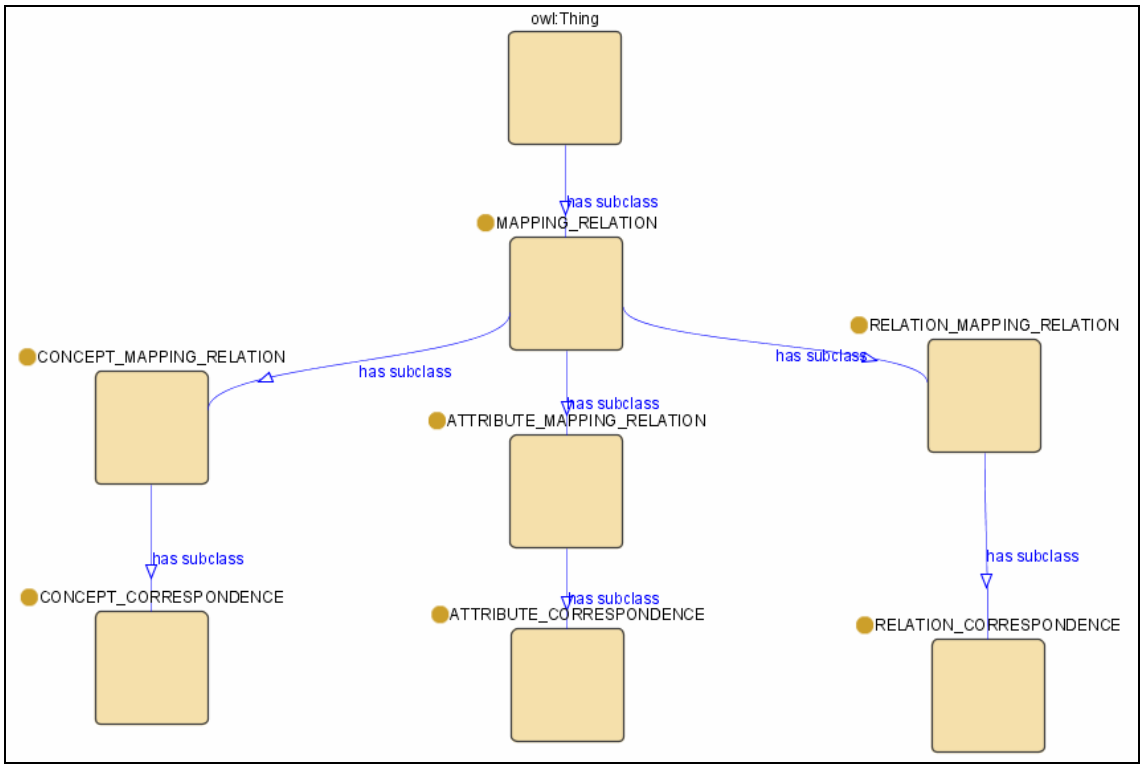


Figura 24. Ontología de definición de correspondencia entre ontologías. Jerarquía de clases

A continuación presentamos parte del fichero de correspondencia entre la ontología Global y la ontología asociada al portal de Ono, obtenido con la herramienta “Ontology Mapping”.

```
…
<map>
 <Cell>
   <entity1 resource="file:///C:/proyecto/OWL/global.owl#PREMIO"/>
   <entity2 resource="file:///C:/proyecto/OWL/imdb.owl#AWARD"/>
   <measure datatype="xsd;float">1.0</measure>
   <relation>CONCEPT_CORRESPONDENCE</relation>
 </Cell>
</map>
<map>
<map>
 <Cell>
   <entity1 resource="file:///C:/proyecto/OWL/global.owl#CONTENIDO.nacionalidad"/>
   <entity2 resource="file:///C:/proyecto/OWL/imdb.owl#CONTENIDO.country"/>
   <measure datatype="xsd;float">1.0</measure>
   <relation>ATTRIBUTE_CORRESPONDENCE</relation>
 </Cell>
</map>
</map>
```

```
<map>
<map>
 <Cell>
   <entity1 resource="file:///C:/proyecto/OWL/global.owl#CONTENIDO contenido_esReconocidoPor_premio PREMIO"/>
   <entity2 resource="file:///C:/proyecto/OWL/imdb.owl#CONTENIDO contenido_esReconocidoPor_premio AWARD"/>
   <measure datatype="xsd;float">1.0</measure>
   <relation>RELATION_CORRESPONDENCE</relation>
 </Cell>
</map>
</map>
…
```

Vemos como con la herramienta ha sido posible hacer uso de las correspondencias definidas en nuestra “ontología de definición de correspondencias” e indicar si la correspondencia en cuestión es entre atributos, conceptos o clases. Por ejemplo, en la primera correspondencia del ejemplo, indica que el recurso file:///C:/proyecto/OWL/global.owl#PREMIO, se corresponde con el recurso file:///C:/proyecto/OWL/imdb.owl#AWARD, que la correspondencia tiene un nivel de confianza 1 (el máximo), y que los recursos en cuestión son conceptos (clases). La segunda correspondencia es entre dos atributos y la tercera entre dos relaciones.

La ubicación de cada uno de estos ficheros XML generados con la herramienta “Ontology Mapping” está definida en el fichero de propiedades “mapeo.properties” del cual dispone la aplicación.

La correspondencia entre la ontología global y las parciales se lleva a cabo, una vez desplegada la aplicación web, en la primera búsqueda realizada del subsistema “Integración de Información”. Una vez realizado el mapeo no se vuelve a llevar a cabo, de esta forma conseguimos un sistema más eficiente. Para el análisis del fichero de mapeo XML se ha hecho uso de JAXB 2.0 (Java Architecture for XML Binding). El XML Schema usado lo hemos definido en función de la estructura de los ficheros de mapeo obtenidos a partir de la herramienta “Ontology Mapping” usada.

En la siguiente tabla mostramos la correspondencia entre la ontología global y cada una de las ontologías parciales. El significado de la tabla se describe a continuación:

o Clases. Se muestra la correspondencia entre los nombres de las clases de la ontología global y cada una de las parciales.

o Atributos o Datatype Properties. Para cada uno de los atributos de la ontología global se muestra su correspondencia con cada una de las ontologías parciales. A su vez, para cada uno de los atributos se muestra la clase a la que pertenece.

o Relaciones u Object Properties. Para cada uno de las relaciones de la ontología global se muestra su correspondencia con cada una de las ontologías parciales. A su vez, para cada uno de las relaciones se muestra la clase a la que pertenece.

Ontología “Global”		Ontología “Digital Plus”		Ontología “ONO”		Ontología “IMDb”	
CLASSES							
CONTENIDO		CONTENIDO		CONTENIDO		CONTENIDO	
PERSONA		PERSONA		PERSONA		PERSONA	
CANAL		CANAL		CANAL			
OPERADOR		OPERADOR		OPERADOR			
EMISION		EMISION		EMISION			
SUBTITULO		SUBTITULO					
VERSION		VERSION					
PREMIO						AWARD	
DATATYPE PROPERTIES							
Nombre de la Propiedad	Clase	Nombre de la Propiedad	Clase	Nombre de la Propiedad	Clase	Nombre de la Propiedad	Clase
subgenero	CONTENIDO	subgenero	CONTENIDO	subgenero	CONTENIDO	genre	CONTENIDO
nombreOriginal	CONTENIDO	nombreOriginal	CONTENIDO	nombreOriginal	CONTENIDO	nombreOriginal	CONTENIDO
calificacionMoral	CONTENIDO	calificacionMoral	CONTENIDO				
sonidoOriginal	CONTENIDO					soundMix	CONTENIDO
nacionalidad	CONTENIDO	nacionalidad	CONTENIDO	nacionalidad	CONTENIDO	country	CONTENIDO
genero	CONTENIDO	genero	CONTENIDO	genero	CONTENIDO		
uriTrailer	CONTENIDO	uriTrailer	CONTENIDO			uriTrailer	CONTENIDO
color	CONTENIDO					color	CONTENIDO
uriImagen	CONTENIDO					uriPhoto	CONTENIDO
episodio	CONTENIDO	episodio	CONTENIDO	episodio	CONTENIDO		
sinopsis	CONTENIDO	argumento	CONTENIDO	sinopsis	CONTENIDO	plotSynopsis	CONTENIDO
duracion	CONTENIDO	duracion	CONTENIDO	duracion	CONTENIDO	runtime	CONTENIDO
temporada	CONTENIDO	temporada	CONTENIDO				
ano	CONTENIDO	ano	CONTENIDO	ano	CONTENIDO	year	CONTENIDO
nombre	CONTENIDO	nombre	CONTENIDO	nombre	CONTENIDO	nombre	CONTENIDO
rol	PERSONA	rol	PERSONA	rol	PERSONA	rol	PERSONA
nombre	PERSONA	nombre	PERSONA	nombre	PERSONA	nombre	PERSONA
uri	PERSONA					uri	PERSONA
nombre	OPERADOR	nombre	OPERADOR	nombre	OPERADOR		
descripcion	OPERADOR	descripcion	OPERADOR	descripcion	OPERADOR		
uriLogo	OPERADOR	uriLogo	OPERADOR	uriLogo	OPERADOR		

descripcion	CANAL	descripcion	CANAL	descripcion	CANAL		
uriLogo	CANAL	uriLogo	CANAL				
nombre	CANAL	nombre	CANAL	nombre	CANAL		
categoria	CANAL	categoria	CANAL	categoria	CANAL		
dial	CANAL	dial	CANAL	dial	CANAL		
fechaEmision	EMISION	fechaEmision	EMISION	fechaEmision	EMISION		
tipoDePase	EMISION	tipoDePase	EMISION				
idiomaDeSubtitulo	SUBTITULO	idiomaDeSubtitulo	SUBTITULO				
tipoDeSubtitulo	SUBTITULO	tipoDeSubtitulo	SUBTITULO				
sonido	VERSION	sonido	VERSION				
idiomaVersion	VERSION	idiomaVersion	VERSION				
categoriaPremio	PREMIO					category	AWARD
emisorPremio	PREMIO					emisor	AWARD
resultado	PREMIO					result	AWARD
ano	PREMIO					year	AWARD
nombre	PREMIO					award	AWARD
OBJECT PROPERTIES							
Nombre de la Propiedad	Clase	Nombre de la Propiedad	Clase	Nombre de la Propiedad	Clase	Nombre de la Propiedad	Clase
emision_tiene_subtitulos	EMISION	emision_tiene_subtitulos	EMISION				
emision_difunde_contenido	EMISION	emision_difunde_contenido	EMISION	emision_difunde_contenido	EMISION		
emision_ofrecidaPor_canal	EMISION	emision_ofrecidaPor_canal	EMISION	emision_ofrecidaPor_canal	EMISION		
emision_tiene_versiones	EMISION	emision_tiene_versiones	EMISION				
contenido_incolucra_personas	CONTENIDO	contenido_incolucra_personas	CONTENIDO	contenido_incolucra_personas	CONTENIDO	contenido_incolucra_personas	CONTENIDO
contenido_esDifundidoPor_emision	CONTENIDO	contenido_esDifundidoPor_emision	CONTENIDO	contenido_esDifundidoPor_emision	CONTENIDO		
contenido_esReconocidoPor_premio	CONTENIDO					contenido_esReconocidoPor_premio	CONTENIDO
persona_involucradaEn_contenido	PERSONA	persona_involucradaEn_contenido	PERSONA	persona_involucradaEn_contenido	PERSONA	persona_involucradaEn_contenido	PERSONA
persona_reconocidaPor_premio	PERSONA					persona_reconocidaPor_premio	PERSONA
operador_tiene_canales	OPERADOR	operador_tiene_canales	OPERADOR	operador_tiene_canales	OPERADOR		
canal_ofrece_emisiones	CANAL	canal_ofrece_emisiones	CANAL	canal_ofrece_emisiones	CANAL		
canal_pertenece_operador	CANAL	canal_pertenece_operador	CANAL	canal_pertenece_operador	CANAL		
premio_asociadoA_contenido	PREMIO					premio_asociadoA_contenido	AWARD
premio_asociadoA_persona	PREMIO					premio_asociadoA_persona	AWARD

3.5 PROCESAMIENTO DE CONSULTAS

La interfaz de usuario de la aplicación permite al usuario realizar consultas de contenidos. El usuario desconoce la existencia de los recursos de forma independiente, la interfaz es genérica y engloba a los recursos independientes. El usuario podrá introducir un patrón de búsqueda, una expresión regular [34], para los atributos que desea que se tengan en cuenta en la búsqueda.

En la búsqueda se pueden tener en cuenta dos tipos de atributos, de cadena y numéricos, en los atributos de cadena el usuario podrá indicar si quiera que la búsqueda sea sensitiva[4] o no y, en caso de atributos numéricos podrá indicar si quiera que los resultados tengan un valor mayor, menor o igual del indicado. En la figura 25 mostramos el formulario de búsqueda que utiliza el usuario.

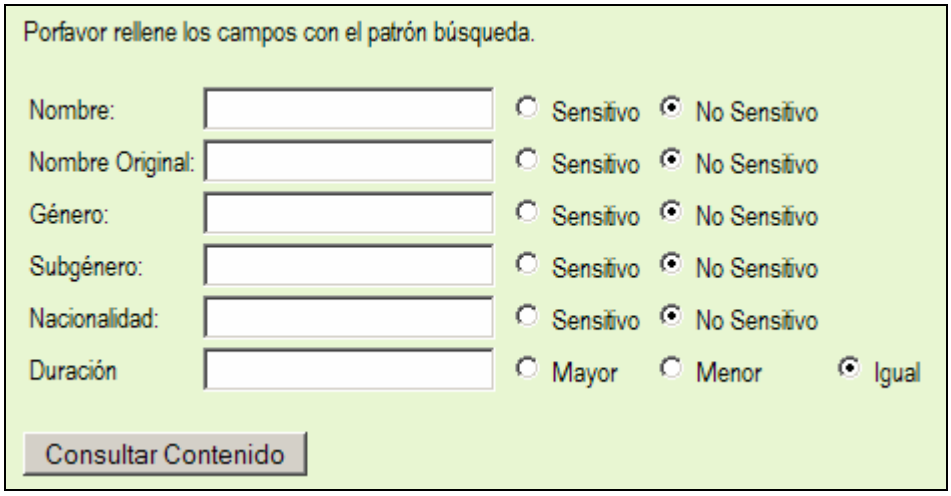


Figura 25. Formulario de búsqueda

Las peticiones son enviadas desde la interfaz hacia el controlador de la aplicación, el cual delegará en la clase responsable de realizar la búsqueda global solicitada. Hay una única clase encargada de realizar las búsquedas globales por contenido.

[4] Tendrá en cuenta las mayúsculas/minúsculas introducidas.

En base al formulario de búsqueda de la interfaz de usuario (ver figura 25) y haciendo uso de JAXB 2.0 el sistema genera un documento XML que indica qué atributos tienen que tenerse en cuenta en la búsqueda, qué valor tienen, de qué tipo son (numérico o de cadena) y qué tipo de búsqueda se desea realizar, sensitivo o no en caso de que sea un atributo de cadena y mayor/menor/igual en caso de que sea un atributo numérico. Este documento XML es utilizado para construir de forma automática las tres consultas SPARQL [32] que van a ejecutarse hacia las ontologías parciales. Los nombres de atributos indicados en el documento XML hacen referencia a atributos de la ontología global siendo necesario saber cuál es la correspondencia de estos atributos en las ontologías parciales. La correspondencia entre los atributos, clases y/o relaciones de la ontología global con cada una de las ontologías parciales se realiza en la primera consulta que se efectúe, de modo que cada vez que se realice una consulta no se tenga que realizar la correspondencia. En la correspondencia se hacen uso de los documentos XML generados con la herramienta Ontology Mapping.

Los resultados obtenidos [33] para cada una de las consultas SPARQL realizadas son documentos XML con un determinado XML Schema asociado. A lo largo de la evolución de SPARQL se han hecho uso de varios XML Schema para los resultados. Como se ha utilizado la API Jena-2.5.4 hemos consultado de qué XML Schema se hace uso encontrando que la versión usada es la versión del 24-09-07. A partir de este XML Schema usamos JAXB 2.0 para la generación de clases a partir de él, utilizándolas para analizar e integrar los resultados obtenidos de las tres consultas en una instancia de la ontología global. La estructura simplificada del formato de los resultados la mostramos a continuación.

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">

 <head>
  <variable name="x"/>
  <variable name="y"/>
  .....
 </head>
```

```
...
  <results>
    <result>
      <binding name="x"> ... </binding>
      <binding name="hpage"> ... </binding>
    </result>
    <result>
      <binding name="x"> ... </binding>
      <binding name="hpage"> ... </binding>
    </result>
    <result>...
    </result>
    ...
  </results>
</sparql>
```

Está formada por una cabecera que contiene una secuencia de elementos que describen las variables de la consulta, seguida de cada uno de los resultados obtenidos. Cada uno de estos resultados esta formado por una secuencia ordenada de etiquetas llamadas “binding” que contienen el nombre de la variable y su valor. Para saber si hemos construido correctamente las consultas y si los resultados son los esperados nos hemos apoyado en el panel de consultas SPARQL que ofrece Protégé.

La integración de los resultados se ha hecho analizando cada uno de los documentos XML de respuesta de las tres consultas. Finalmente después de analizar los resultados de las tres consultas SPARQL e integrar la información en una instancia de la ontología global las vistas imprimirán el contenido de la instancia.

Ejemplo de Uso:

Paso 1. El usuario, a través de la interfaz, solicita una búsqueda introduciendo en el formulario como “nombre original” el valor “snake” y como “duración” un valor menor de 400.

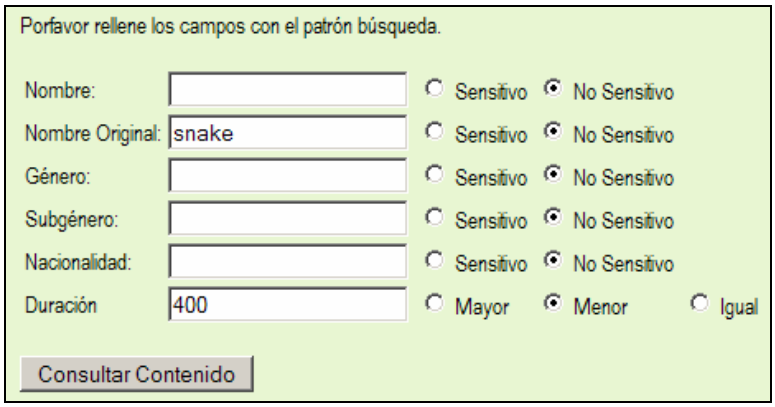


Figura 26. Formulario de búsqueda. Ejemplo

Paso 2. El controlador recibe la petición y crea un documento XML con los atributos solicitados. El documento XML generado se muestra a continuación:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<parametros>
  <parametroCadena>
    <nombre>8</nombre>
    <valor>snake</valor>
    <tipoComparacion>1</tipoComparacion>
  </parametroCadena>
  <parametroNumerico>
    <nombre>3</nombre>
    <valor>400</valor>
    <tipoComparacion>1</tipoComparacion>
  </parametroNumerico>
</parametros>
```

El nombre del atributo que hace referencia a la ontología global y los tipos de comparación, en nuestro caso “CASO_SENSITIVO” y “MENOR”, están definidos como variables estáticas finales de tipo entero en una clase disponible por el resto de clases de la aplicación. El documento XML es enviado al modelo para que realice la consulta a cada una de las ontologías e integre el resultado en una instancia de la ontología global.

Paso 3. El fichero es utilizado para realizar las 3 consultas SPARQL hacia cada una de las ontologías parciales. Las consultas son construidas automáticamente. A continuación se muestran las tres consultas realizadas. La consulta realizada a la ontología asociada al portal del operador Digital Plus es:

```
P1. PREFIX pre: <file:///C:/proyecto/OWL/proyecto.owl#>
P2. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
P3. SELECT ?ano ?calificacionMoral ?duracion ?episodio ?genero ?nacionalidad ?nombre ?nombreOriginal ?sinopsis
?subgenero ?temporada ?uriTrailer WHERE {
OPTIONAL {?y pre:ano ?ano}
OPTIONAL {?y pre:calificacionMoral ?calificacionMoral}
OPTIONAL {?y pre:duracion ?duracion}
OPTIONAL {?y pre:episodio ?episodio}
OPTIONAL {?y pre:genero ?genero}
OPTIONAL {?y pre:nacionalidad ?nacionalidad}
OPTIONAL {?y pre:nombre ?nombre}
OPTIONAL {?y pre:nombreOriginal ?nombreOriginal}
OPTIONAL {?y pre:argumento ?sinopsis}
OPTIONAL {?y pre:subgenero ?subgenero}
OPTIONAL {?y pre:temporada ?temporada}
OPTIONAL {?y pre:uriTrailer ?uriTrailer}
P5. ?y rdf:type pre:CONTENIDO .
P6. FILTER regex(?nombreOriginal, "snake","i")
P7. FILTER (?duracion<400)
}
```

La consulta realizada a la ontología asociada al portal de Ono es:

```
P1. PREFIX pre: <file:///C:/proyecto/OWL/proyecto.owl#>
P2. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
P3. SELECT ?ano ?duracion ?episodio ?genero ?nacionalidad ?nombre ?nombreOriginal ?sinopsis ?subgenero
WHERE {
OPTIONAL {?y pre:ano ?ano}
OPTIONAL {?y pre:duracion ?duracion}
OPTIONAL {?y pre:episodio ?episodio}
OPTIONAL {?y pre:genero ?genero}
OPTIONAL {?y pre:nacionalidad ?nacionalidad}
OPTIONAL {?y pre:nombre ?nombre}
OPTIONAL {?y pre:nombreOriginal ?nombreOriginal}
OPTIONAL {?y pre:sinopsis ?sinopsis}
OPTIONAL {?y pre:subgenero ?subgenero}
P5. ?y rdf:type pre:CONTENIDO .
P6. FILTER regex(?nombreOriginal, "snake","i")
P7. FILTER (?duracion<400)
}
```

La consulta realizada a la ontología asociada al portal de IMDb es:

```
P1. PREFIX pre: <file:///C:/proyecto/OWL/proyecto.owl#>
P2. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
P3. SELECT ?ano ?color ?duracion ?nacionalidad ?nombre ?nombreOriginal ?sinopsis ?subgenero ?uriImagen
```

```
?uriTrailer ?sonidoOriginal
WHERE {
OPTIONAL {?y pre:year ?ano}
OPTIONAL {?y pre:color ?color}
OPTIONAL {?y pre:runtime ?duracion}
OPTIONAL {?y pre:country ?nacionalidad}
OPTIONAL {?y pre:nombre ?nombre}
OPTIONAL {?y pre:nombreOriginal ?nombreOriginal}
OPTIONAL {?y pre:plotSynopsis ?sinopsis}
OPTIONAL {?y pre:genre ?subgenero}
OPTIONAL {?y pre:uriPhoto ?uriImagen}
OPTIONAL {?y pre:uriTrailer ?uriTrailer}
OPTIONAL {?y pre:soundMix ?sonidoOriginal}
P4. ?y rdf:type pre:CONTENIDO .
P5. FILTER regex(?nombreOriginal, "snake","i")
P6. FILTER (?duracion<400)
}
```

El modo de construir las consultas es el mismo para los tres recursos. A continuación vamos a explicar en detalle cómo hemos construido cada una de las tres consultas anteriores, hay que tener en cuenta que la correspondencia entre los atributos, clases y relaciones de la ontología global a cada una de las parciales se establecen en la primera consulta que se efectúe.

P1. Se construye en base al prefijo de la ontología. Los prefijos de las ontologías están indicados en fichero de propiedades “urisOntologias.properties”.

P2. Hace referencia al vocabulario RDF. Este valor es constante.

P3. La correspondencia entre los atributos, clases y relaciones entre la ontología global y parcial están almacenados en memoria. Se preguntan por todos los atributos de la clase “Contenido” ya que queremos obtener toda la información posible para después imprimirla. El nombre de los atributos depende de la ontología en cuestión, además es posible que haya atributos que no existen en la ontología parcial y si en la global, es decir, que no tengan correspondencia.

P4. Se indica que se pregunta por instancias de la clase “Contenido”.

P5. Como en el formulario se ha indicado que el nombre original del contenido deber ser “snake”, se hace un filtro en base a esta cadena de texto. Hay que tener en cuenta que el usuario introduce en el formulario una expresión regular.

P6. Se vuelve a hacer un filtro para el segundo atributo por el que se preguntado, el atributo “duracion”.

Paso 4. Para cada una de las consultas anteriores se obtiene un documento XML con los resultados de la consulta. Estos documentos, en ocasiones tienen un tamaño muy grande y pueden ocasionar problemas de rendimiento. Se muestra a continuación un pequeño trozo del documento resultado de la última consulta.

```
<?xml version="1.0"?>
<sparql
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema#"
   xmlns="http://www.w3.org/2005/sparql-results#" >
 <head>
   <variable name="ano"/>
   <variable name="color"/>
   <variable name="duracion"/>
   <variable name="nacionalidad"/>
   <variable name="nombre"/>
   <variable name="nombreOriginal"/>
   <variable name="sinopsis"/>
   <variable name="subgenero"/>
   <variable name="uriImagen"/>
   <variable name="uriTrailer"/>
   <variable name="sonidoOriginal"/>
 </head>
 <results>
  <result>
    <binding name="ano">
     <literal datatype="http://www.w3.org/2001/XMLSchema#int">1998</literal>
    </binding>
    <binding name="color">
     <literal datatype="http://www.w3.org/2001/XMLSchema#string">Color (DeLuxe)</literal>
    </binding>
    <binding name="duracion">
     <literal datatype="http://www.w3.org/2001/XMLSchema#int">98</literal>
    </binding>
    <binding name="nacionalidad">
     <literal datatype="http://www.w3.org/2001/XMLSchema#string">USA</literal>
    </binding>
    <binding name="nombre">
     <literal datatype="http://www.w3.org/2001/XMLSchema#string">Snake Eyes</literal>
    </binding>
    <binding name="nombreOriginal">
```

```
      <literal datatype="http://www.w3.org/2001/XMLSchema#string">Snake Eyes</literal>
    </binding>
    <binding name="sinopsis">
      <literal datatype="http://www.w3.org/2001/XMLSchema#string"></literal>
    </binding>
    <binding name="subgenero">
      <literal datatype="http://www.w3.org/2001/XMLSchema#string">Crime</literal>
    </binding>
    <binding name="uriImagen">
      <literal datatype="http://www.w3.org/2001/XMLSchema#string">http://www.imdb.com/gallery/granitz/0431-
sna/Events/0431-sna/cagenico.las.html?path=gallery&amp;path_key=0120832</literal>
    </binding>
    <binding name="uriTrailer">
      <literal
datatype="http://www.w3.org/2001/XMLSchema#string">http://www.trailerfan.com/movie/snake_eyes/trailer</literal>
    </binding>
    <binding name="sonidoOriginal">
      <literal datatype="http://www.w3.org/2001/XMLSchema#string">SDDS</literal>
    </binding>
  </result>
…
```

Paso 5. Por último los documentos XML de respuesta son analizados e integrados en una instancia de la ontología global. Para el análisis, como hemos explicado anteriormente, se hacen uso de las clases generadas con JAXB 2.0 a partir del XML Schema asociado. Como sabemos cual es la estructura de los resultados, la analizamos sin ningún problema. El único aspecto a tener en cuenta es qué campos tenemos en cuenta para considerar que 2 resultados hacen referencia al mismo objeto. En el caso de los contenidos hemos hecho uso del campo “nombreOriginal” para llegar a esta conclusión, se podría haber utilizado adicionalmente el campo que hace referencia a la nacionalidad y al año, pero hay que tener en cuenta el año es un campo opcional y distintos valores para la nacionalidad pueden hacer referencia a la misma, por ejemplo, USA, EEUU, E.E.U.U, United Status, serían sinónimos. Esto lo dejamos como vía futura y posible mejora. En la siguiente tabla mostramos qué campos se han utilizado a la hora de decidir qué dos resultados hacen referencia al mismo objeto.

Contenido → Nombre del contenido.
Emisión → Contenido asociado a la emisión y fecha de emisión.

Operador → Nombre del operador.
Persona → Nombre de la persona.
Premio → Categoría del premio, nombre del premio y nombre del contenido al que hace referencia.

Finalmente los resultados integrados son mostrados al usuario a través de la interfaz de usuario.

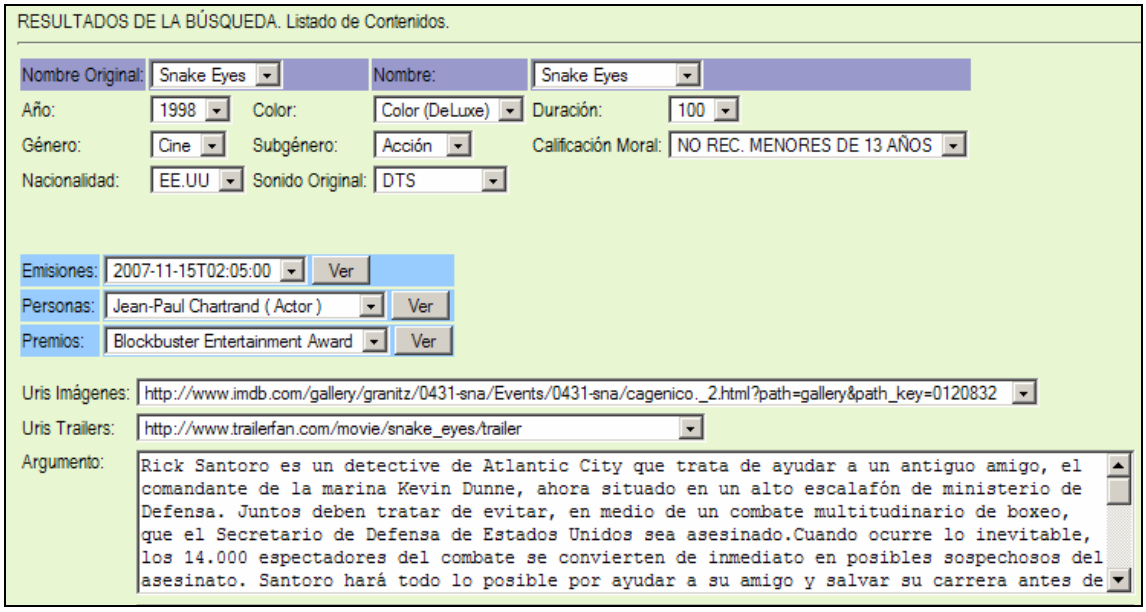


Figura 27. Resultados de una búsqueda

Paso 6. La vista de la figura anterior solo muestra la información relativa a los Contenidos y un listado de Premios, Personas, y Emisiones. Para ver información relativa a los Premios, las Personas, las Emisiones, los Canales y los Operadores se deberá navegar por la interfaz de usuario. El acceso a los Premios, Personas y Emisiones está directamente accesible desde la página de Contenidos (ver figura 27), sin embargo, los canales están accesibles desde las Emisiones, y los Operadores desde los canales. Esta es la forma que hemos escogido para navegar por la interfaz, se podría haber hecho de otra forma ya que, en las ontologías desarrolladas, todas las relaciones entre las clases, excepto las relaciones de Emisión con Versión y Subtítulo, se han establecido de forma bidireccional. En el siguiente diagrama (ver figura número

28) se muestra la forma de navegar por la interfaz. Cada vez que el usuario realiza una acción de navegar por la interfaz se repiten los pasos 2, 3, 4 y 5. El documento XML de parámetros generado será diferente para cada caso con el fin de realizar la búsqueda solicitada. Por ejemplo si a partir del resultado obtenido en la figura 27 se desea navegar a la primera emisión listada, el resultado generado sería el de la figura 29.

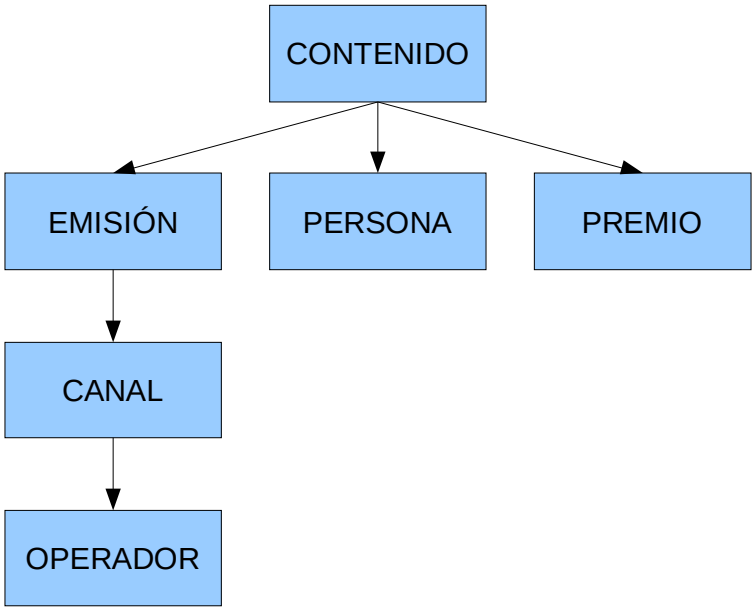


Figura 28. Navegabilidad en la Interfaz de Usuario

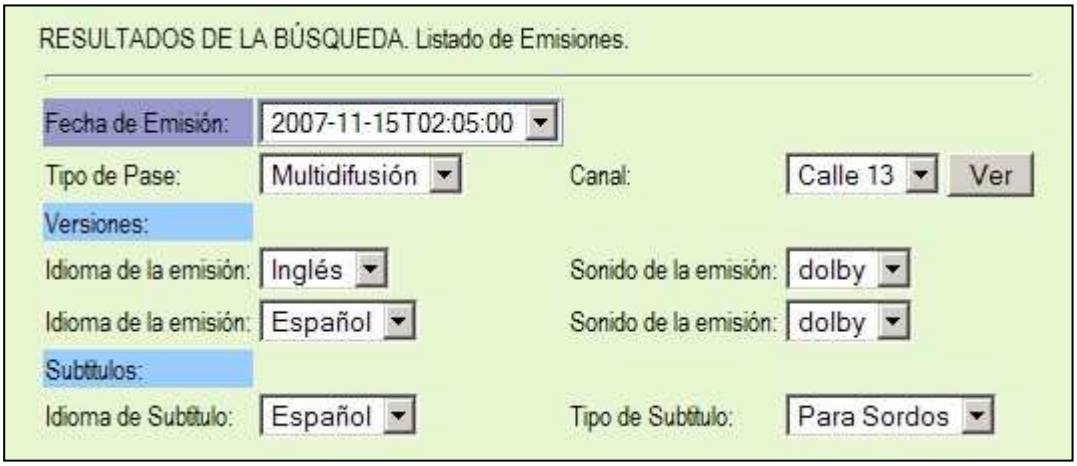


Figura 29. Características de una Emisión asociada de un Contenido

3.6 VISUALIZACIÓN DE RESULTADOS

Para las vistas se ha hecho de Java Server Pages (JSP). JSP es una tecnología java que permite generar código dinámico para web permitiendo la utilización de código de programación java. Todas las etiquetas del sistema están definidas en el fichero “contenido.tld” dentro de la carpeta “WEB-INF/etiquetas” de la aplicación web “ModuloWeb.war” e implementadas en el paquete “etiquetas”.

En el subsistema “Recuperación de Información” se hace uso de una única página JSP encargada de imprimir un mensaje que indica el resultado de la acción solicitada indicando si la acción se ha efectuado con éxito o no.

En el subsistema “Integración de Información” las vistas son más complejas, utilizándose etiquetas JSP. Las etiquetas JSP del subsistema “Integración de Información” se pueden clasificar, según su finalidad, en 2 grupos, un primer grupo encargado de imprimir todos los individuos de una clase de una instancia de la ontología global o los individuos asociados a una propiedad de tipo objeto asociado a un individuo de una instancia de la ontología global, y un segundo grupo encargado de imprimir los valores de una propiedad de tipo de datos asociado a un individuo. Con las etiquetas del primer grupo se podrán llevar a cabo búsquedas recursivas a partir de un individuo, por ejemplo, si el usuario realizara una búsqueda sobre un nombre de contenido y se obtienen como resultado 3 contenidos, se invocaría a una etiqueta del primer grupo para imprimir los 3 contenidos, y a continuación se volvería a invocar a otra etiqueta del primer grupo para imprimir los individuos asociados a sus propiedades de tipo objeto.

3.7 ARQUITECTURA LÓGICA

La arquitectura lógica del sistema está totalmente estructurada y adaptada al patrón Modelo-Vista-Controlador [3], al patrón FrontController [12] y al patrón Data Access Object (DAO) [13]. Haciendo uso de estos patrones se ha conseguido un sistema con una estructura lógica totalmente modular separando la lógica de negocio de la lógica de diseño.

El diagrama de paquetes se muestra a continuación. La interfaz de usuario es una interfaz web definida por páginas HTML y JSP que hacen uso de una serie de etiquetas definidas en el paquete “etiquetas”. El controlador está definido en el paquete “controlador” y subpaquetes correspondientes (“acciones”, “acciones.digital”, “acciones.ono”, “acciones.digital” y “acciones.imdb”), se hace uso de un servlet que recibe todas las peticiones e invoca y delega al modelo o capa de datos el acceso a las ontologías correspondientes. El modelo o capa de datos está formado por el paquete “dao” y el paquete “mapeo” que son los paquetes que incluyen las clases de acceso a las ontologías.

El paquete que implementa la capa de presentación se comunica con el que implementa la capa de negocio para que este realice las operaciones que ha solicitado el usuario. Para ello obtiene los datos necesarios comunicándose con el paquete que implementa la capa de datos. De este modo las tres capas realizan su función de forma independiente, existiendo un interfaz controlado en los accesos de una a otra, lo que simplifica mucho el mantenimiento de la aplicación al poderse identificar fácilmente el módulo a modificar y las dependencias de este con los demás.

El resto de paquetes que se muestran en el diagrama corresponden a las APIs auxiliares que utiliza nuestra plataforma, y también quedan totalmente englobados en alguna de las tres capas mencionadas. Por tanto, podemos realizar una descripción somera de la estructura lógica del sistema comentando una por una las características de estas capas.

DIAGRAMA DE PAQUETES. SISTEMA “RICAO v1.0”

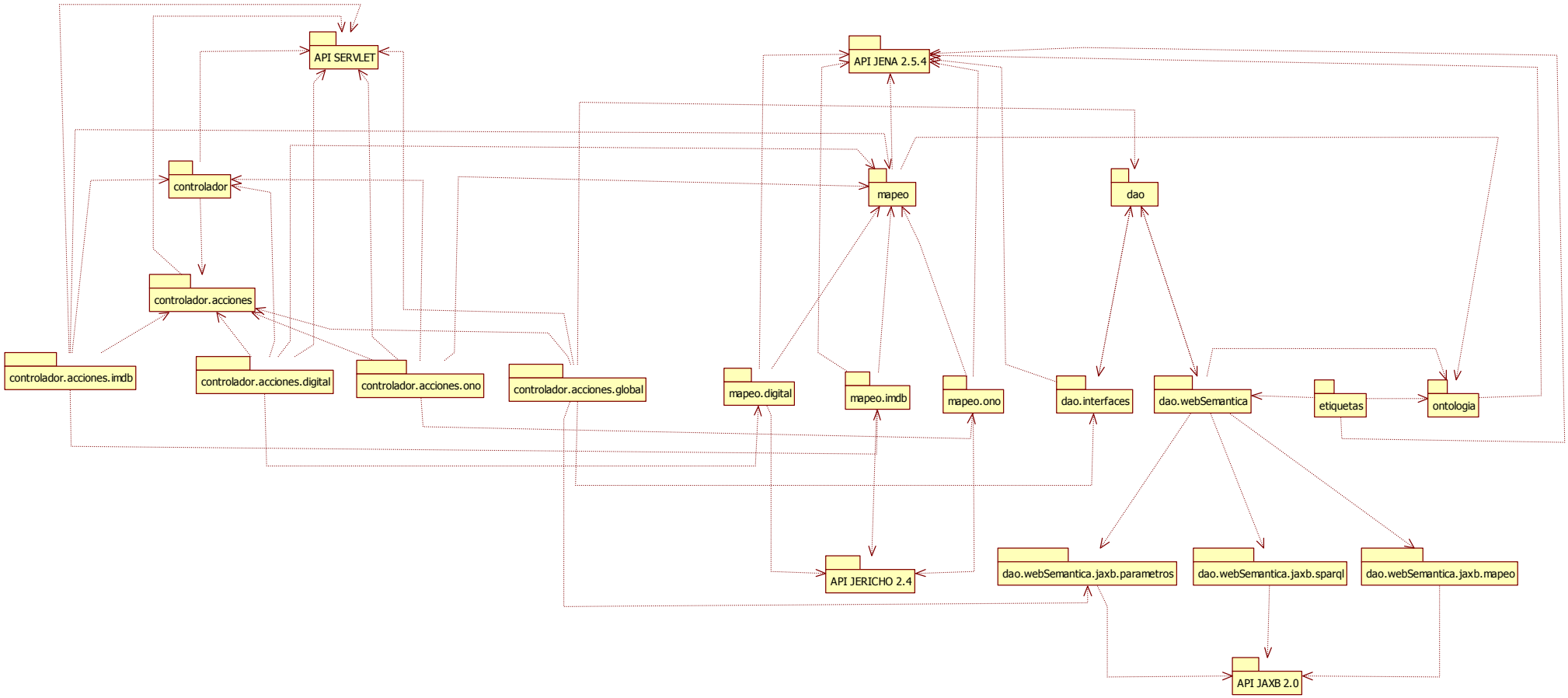


Figura 30. Diagrama de paquetes. Sistema RICAO v1.0

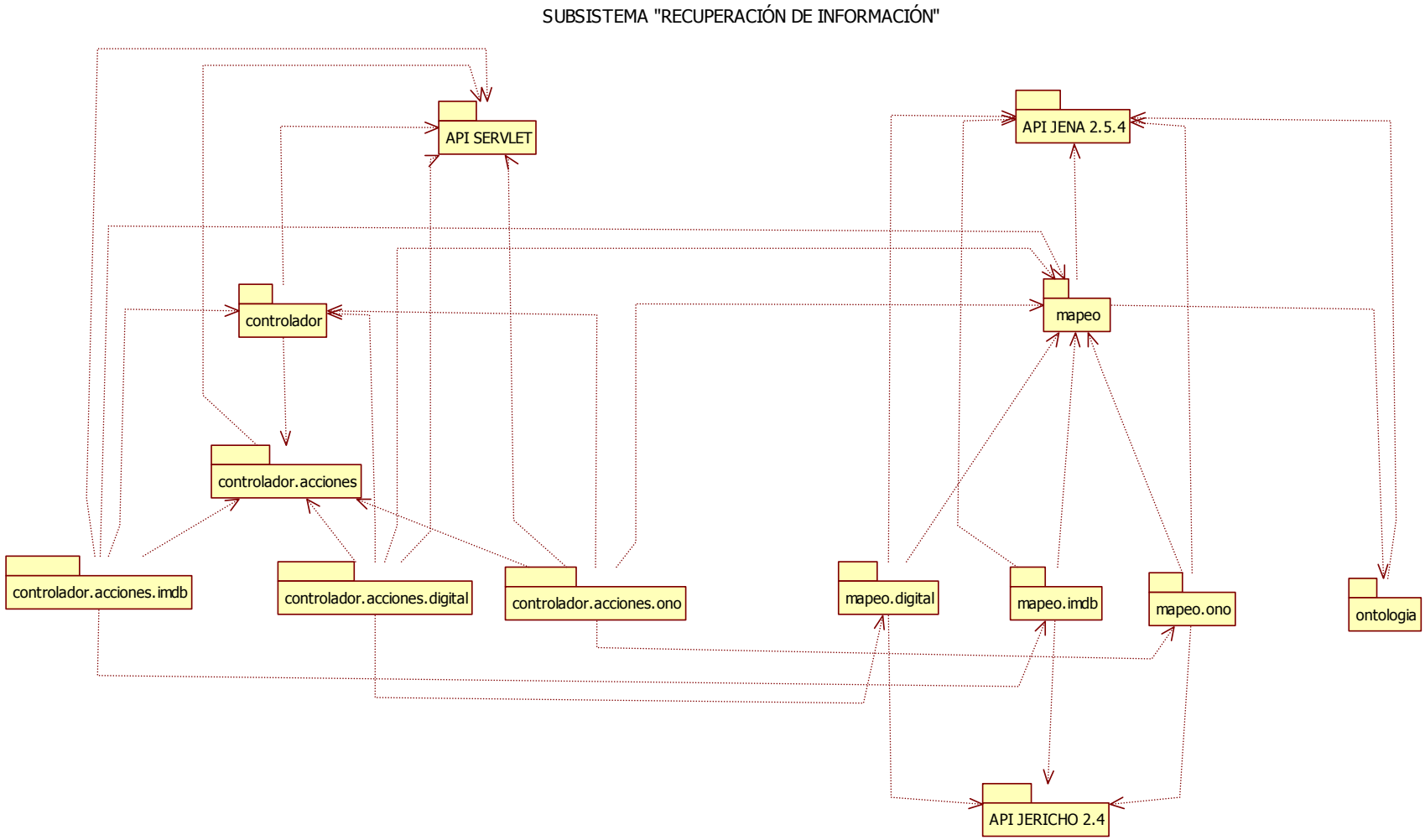


Figura 31. Diagrama de paquetes. Subsistema “Recuperación de Información”

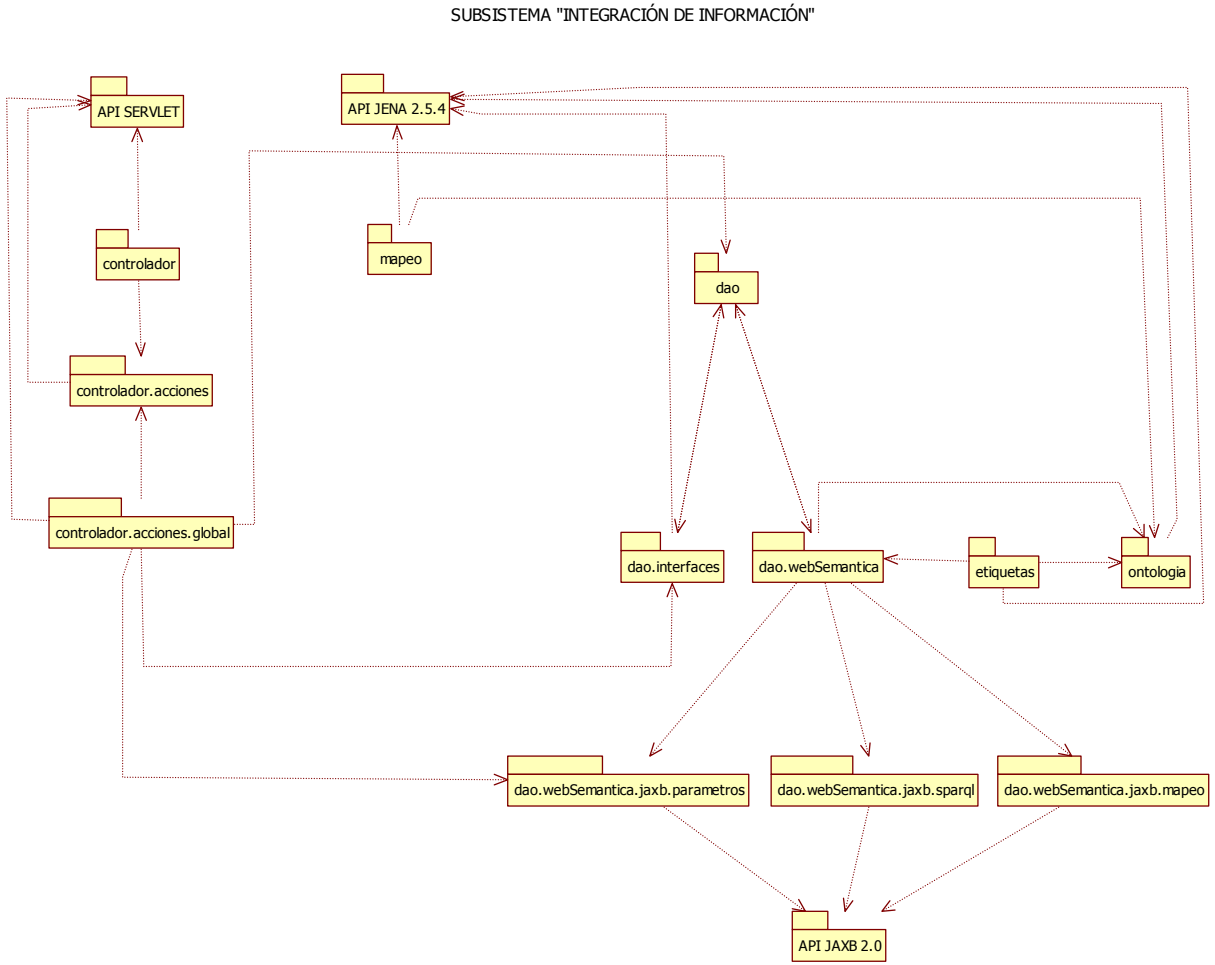


Figura 32. Diagrama de paquetes. Subsistema "Integración de Información"

El uso del patrón Modelo-Vista-Controlador implica que la estructura de la implementación del sistema esté dividida en tres capas, la vista encargada de presentar el modelo, el controlador, encargado de recibir y servir las peticiones de la capa de presentación y el modelo que define la lógica de negocio. A continuación mostramos un diagrama de secuencias donde se muestran las interacciones entre los objetos organizadas en una secuencia temporal.

1) El usuario interactúa con la interfaz de usuario mostrada en su navegador web y solicita la ejecución de una acción. El navegador envía la solicitud al controlador y espera la respuesta. Según el rol que tenga el usuario la solicitud será distinta. Si el rol es “cliente administrador”, dar de alta información de uno de los portales (Ono, Digital Plus, o IMDb) en una ontología. Si el rol es “cliente consultor”, obtener información resultante de la integración de las ontologías asociadas a los diferentes recursos.

2, 3 y 4) El controlador representado por la clase “ControladorFachada” recibe (por parte de los objetos de la vista) la notificación de la acción solicitada por el usuario y, haciendo uso de la clase “ControladorFachadaAyuda”, identifica la clase de la acción a ejecutar. Una vez ejecutada la acción el objeto respuesta será una página JSP.

5, 6, 7 y 8) El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. Haciendo uso de las etiquetas JSP la página JSP tendrá una estructura e información determinada.

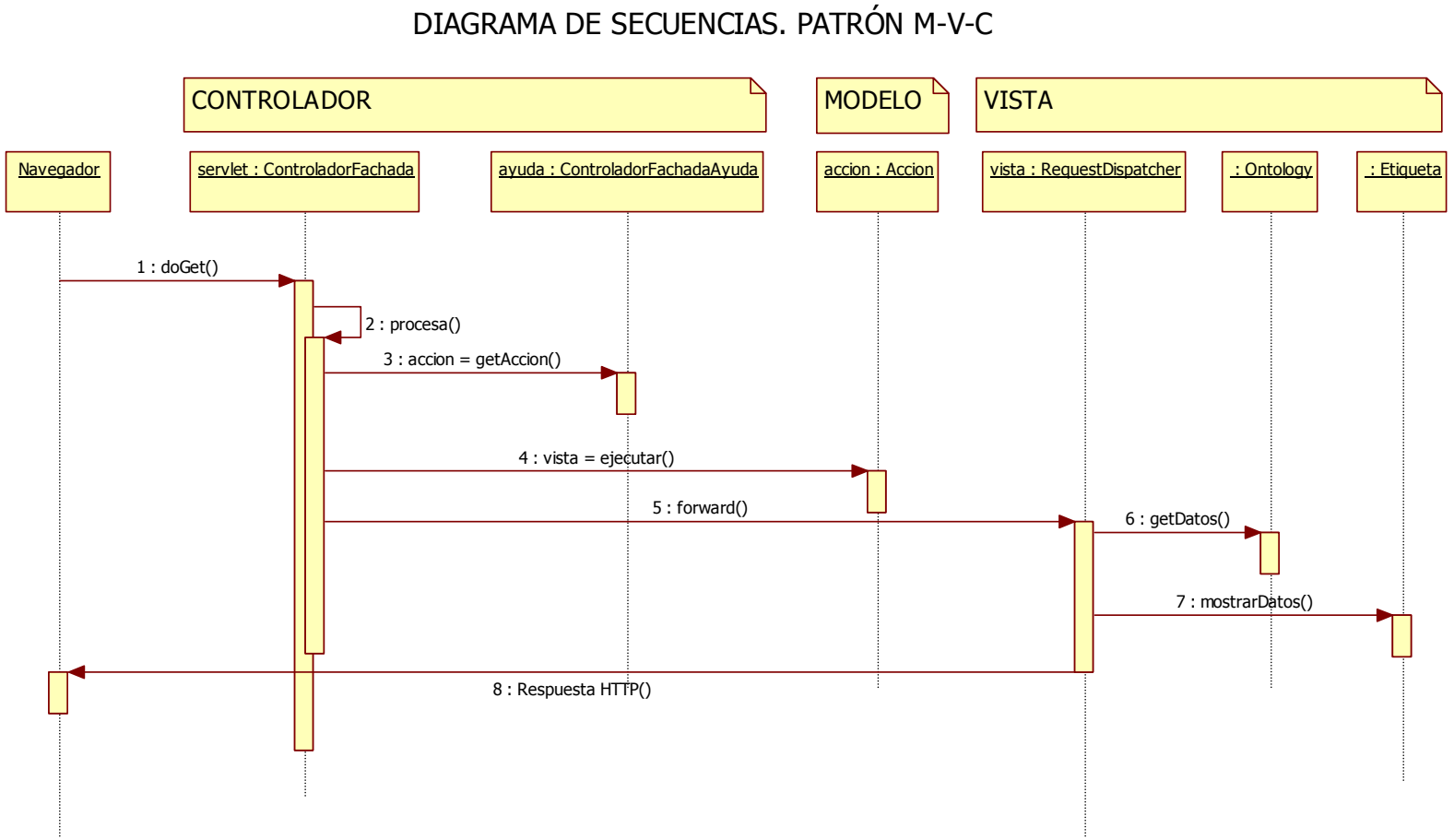


Figura 33. Diagrama de Secuencias. Patrón Modelo Vista Controlador.

3.8 COMPARACIÓN CON OTRAS HERRAMIENTAS

AVATAR “AdVAnced Telematic search of Audiovisual contents by semantic Reasoning” [29].

Avatar es un sistema de recomendación personalizada de contenidos televisivos. El Sistema Avatar surge por la necesidad de herramientas para buscar contenidos de televisión personalizados según las preferencias de cada usuario ante la aparición de la televisión digital y la cantidad de programas irrelevantes para el usuario.

Avatar asocia a cada nuevo contenido un perfil, de forma que el propio sistema, basándose en las preferencias del usuario es capaz de recomendar programas de la ontología de televisión relacionados semánticamente. Avatar a diferencia de RICAO, se centra más en el uso y creación de agentes software que procesen información sobre los contenidos audiovisuales y generen recomendaciones al usuario. La arquitectura del Sistema se muestra a continuación.

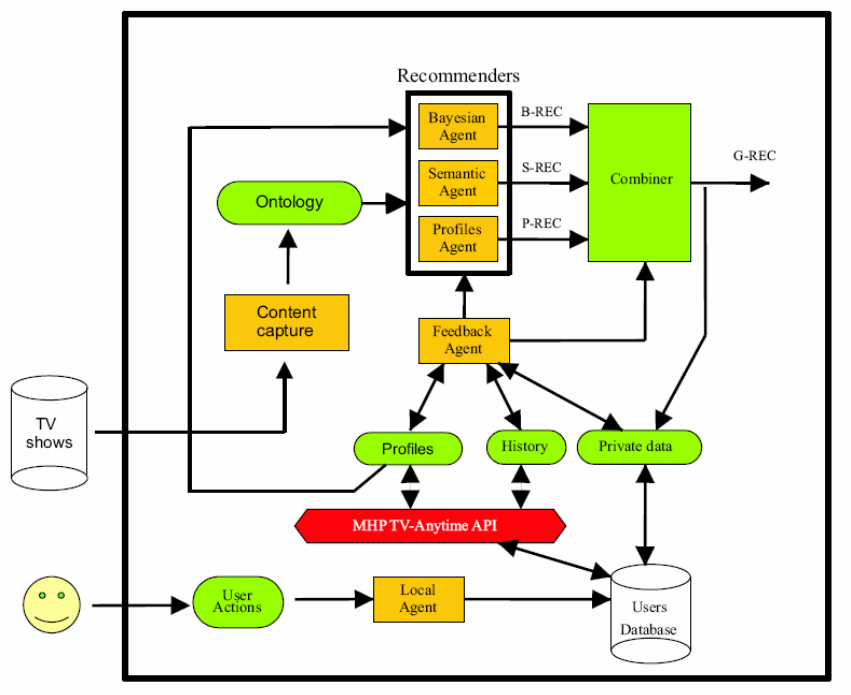


Figura 34. Avatar

4. CONCLUSIONES Y VÍAS FUTURAS

4.1 CONCLUSIONES

Dotar a los contenidos audiovisuales de metainformación es ya una realidad. La adopción de diversas normas propiciarán que el usuario pueda acceder a un sin fin de contenidos audiovisuales y servicios, causando una recepción masiva de información hacia el usuario. Se requerirán métodos de búsqueda e integración de información eficientes que ofrezcan una información fiable y completa adaptada a las necesidades del usuario. Un razonamiento sintáctico basado en comparación de cadenas no podrá abordar el problema, siendo necesario un razonamiento semántico.

Al comienzo del proyecto nos planteábamos un gran objetivo, *"el diseño e implementación de una arquitectura genérica, modular y multiplataforma capaz de:*

1) Analizar información ofrecida por determinados recursos de contenidos audiovisuales disponibles en Internet y almacenarla en ontologías.

2) Ofrecer un mecanismo de búsqueda de contenidos, cuyos resultados sean la integración de información ofrecida por diferentes ontologías. "

Ahora que el proyecto llega a su fin, creemos haberlo alcanzado en la medida de lo posible, teniendo en cuenta que es posible mejorar y ampliar la funcionalidad del sistema.

Hemos construido una aplicación web capaz de almacenar información ofrecida por documentos HTML en ontologías y su posterior integración en base a búsquedas personalizadas de los usuarios.

El primer paso en el proyecto fue seleccionar los portales de contenidos audiovisuales que iban a ser caso de estudio. Se decidió elegir tres portales de Internet que ofrecían contenidos audiovisuales, ONO, Digital Plus e IMDB.

ONO y Digital Plus fueron seleccionados por su gran repercusión e implantación a nivel nacional, e IMBd por ser un portal que ofrecía una amplia información de contenidos audiovisuales de carácter general y ser conocido internacionalmente.

Una vez decididos los portales que iban a ser utilizados, se decidió implementar la arquitectura en dos fases, una para el subsistema "Recuperación de Información" y otra para el subsistema "Integración de Información". La arquitectura del sistema se ha diseñado, fundamentalmente, basándose en tres patrones de diseño; el patrón "Data Access Object", el patrón "Modelo Vista Controlador" y el patrón "Front Controller". De esta forma hemos conseguido una arquitectura modular y altamente reutilizable. Hemos hecho uso de multitud de herramientas y tecnologías, todas ellas con licencia GNU GPL o compatible [8].

Para el subsistema "Recuperación de Información" hemos diseñado un mecanismo eficiente que permite al usuario indicar mediante URL los documentos HTML a analizar para su posterior inclusión en la ontología correspondiente. Se decidió usar un analizador HTML por dos motivos. El primero era que ofrecía una interfaz de alto nivel para manipular documentos HTML y el segundo porque algunos documentos HTML contenían errores y no se adoptaban a la sintaxis XML.

Para el subsistema "Integración de Información" hemos diseñado un mecanismo mediante el cual el usuario podrá realizar una búsqueda de contenidos audiovisuales, obteniendo como resultado la integración de la información disponible en cada una de las ontologías del sistema.

4.2 VÍAS FUTURAS

Existe gran variedad de posibles vías futuras que continúen este proyecto, entre ellas podríamos citar las siguientes:

Mejorar la gestión de ontologías:

- Hacer uso de un gestor de ontologías que tenga mecanismos de seguridad para mantener la integridad, confidencialidad, disponibilidad e irrefutabilidad, y escalabilidad.
- Definición dinámica de las ontologías en la aplicación. Uso de algún mecanismo que defina las clases, atributos y relaciones de las ontologías externo a la propia aplicación.

Mejorar el subsistema “Recuperación de Información”:

- Automatización de la inserción de instancias a partir de un patrón de URLs. Uso de expresiones regulares para indicar qué documentos HTML analizar.
- Incluir algún mecanismo de seguridad en la aplicación de forma que solo los usuarios con privilegios podrán dar de alta a instancias en el sistema.

Mejorar el subsistema “Integración de Información”:

- Introducción de lenguaje natural a la hora de realizar búsquedas.
- Formularios de búsqueda más avanzados y completos.
- Asociación de perfiles a los contenidos y poder realizar búsquedas en base a esos perfiles.
- Control de consistencia entre los recursos que se ha establecido la correspondencia.
- Introducir una caché que almacene las últimas consultas realizadas.
- Incorporar un diccionario terminológico de sinónimos.

5. BIBLIOGRAFÍA

[**1**] ATELLA, S; GENTA, N; GONZÁLEZ, D. *Tutorial de JENA* [en línea]. <www.fing.edu.uy/~lsilva/JenaTutorial.ppt> [Consulta: 25 de Enero de 2008].

[**2**] DIGITAL PLUS. Portal de contenidos audiovisuales. *Catálogo de Contenidos* [en línea]. <http://www.plus.es/guiatv/buscadorbasico.html> [Consulta: 25 de Enero de 2008].

[**3**] FROUFE, A. La Arquitectura MVC [en línea]. <http://www.cica.es/formacion/JavaTut/Apendice/mvc.html> [Consulta: 25 de Enero de 2008].

[**4**] GRUBER T.R. *"Toward Principles for the Design of Ontologies Used for Knowledge Sharing*" Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University, CA, 1993.

[**5**] IMDB. The Internet Movie Database. *Catálogo de Contenidos* [en línea]. <http://www.imdb.com/> [Consulta: 25 de Enero de 2008].

[**6**] JENA DEVELOPERS. *The Jena Javadoc* [en línea]. <http://jena.sourceforge.net/javadoc/index.html> [Consulta: 25 de Enero de 2008]

[**7**] JERICHOHTML. *Jericho HTML Parser [en línea]*. <http://jerichohtml.sourceforge.net/doc/index.html> [Consulta: 25 de Enero de 2008].

[**8**] LABRADOR, R.M.G. *Tipos de Licencias Software* [en línea]. Septiembre de 2005. <http://www.informatica.us.es/~ramon/articulos/LicenciasSoftware.pdf> [Consulta: 25 de Enero de 2008].

[**9**] OMG. *OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2* [en línea]. <http://www.omg.org/docs/formal/07-11-02.pdf> [Consulta: 26 de Enero de 2008].

[**10**] ONO. Portal de contenidos audiovisuales. *Catálogo de Contenidos* [en línea]. <https://servicios.ono.es/programacionTV/> [Consulta: 25 de Enero de 2008].

[**11**] OPEN SOURCE SOFTWARE IN JAVA. Open Source HTML Parsers in Java. <http://java-source.net/open-source/html-parsers/> [Consulta: 25 de Enero de 2008].

[**12**] PALOS, J.A. *Catálogo de Patrones de Diseño J2EE. I.- Capa de Presentación*. <http://www.programacion.com/java/tutorial/patrones/4/> [Consulta: 25 de Enero de 2008].

[**13**] PALOS, J.A. *Catálogo de Patrones de Diseño J2EE. Y II: Capas de Negocio y de Integración.* <http://www.programacion.com/java/tutorial/patrones2/8#patrones28_solucion/> [Consulta: 25 de Enero de 2008].

[**14**] MCBRIDE, B. *An Introduction to RDF and the Jena RDF API* [en línea]. Stanford University. <http://jena.sourceforge.net/tutorial/RDF_API/index.html> [Consulta: 25 de Enero de 2008]

[**15**] MIGUEL, A.; PIATTINI, M.; MARCOS, E. *Notación [MPM 1999]. Diseño de bases de datos relacionales*. Ra-Ma. Capítulo 2.

[**16**] NOY, F.N; DEBORAH, L.M. *Desarrollo de Ontologías-101: Guía Para Crear Tu Primera Ontología* [en línea]. Stanford University. Traducido del inglés por Erick Antezana. Septiembre de 2005. <http://protege.stanford.edu/publications/ontology_development/ontology101-es.pdf> [Consulta: 25 de Enero de 2008].

[**17**] SERVICIO DE COORDINACIÓN DE BIBLIOTECAS. *Cómo citar una bibliografía* [en línea]. <http://www.biblioteca.upm.es/manuales/citabibliografia.htm#textoselectr%F3nicos> [Consulta: 28 de Enero de 2008].

[**18**] SPIVACK. *How the WebOS Evolves?.* <http://novaspivack.typepad.com/nova_spivacks_weblog/2007/02/steps_towards_a.html> [Consulta: 25 de Enero de 2008].

[**19**] STARUML. StarUML 5.0 *Developer Guide.* <http://staruml.sourceforge.net/docs/StarUML_5.0_Developer_Guide.pdf> [Consulta: 25 de Enero de 2008].

[**20**] STUDER S; BENJAMINS R.; FENSEL D. "*Knowledge Engineering: Principles andMethods*", Data and Knowledge Engineering, 25, 161-197, 1998.

[**21**] TECNOMOD (Tecnologías de Modelado, Procesamiento y Gestión del conocimiento). <http://klt.inf.um.es/> [Consulta: 25 de Enero de 2008].

[**22**] TELLO, A.L. *Ontologías en la Web Semántica* [en línea]. <http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf> [Consulta: 25 de Enero de 2008].

[**23**] THE APACHE SOFTWARE FOUNDATION. *Apache Tomcat* [en línea]. <http://tomcat.apache.org/> [Consulta: 25 de Enero de 2008].
DISEÑO DEL SISTEMA

[**24**] UNIVERSIDAD DE MURCIA. Apuntes de la asignatura Arquitectura del Software de 5º de Ingeniería Informática [en línea]. <http://dis.um.es/~joseberm/> [Consulta: 25 de Enero de 2008].

[**25**] UNIVERSIDAD DE MURCIA. Apuntes de la asignatura Desarrollo de Aplicaciones Distribuidas de 5º de Ingeniería Informática [en línea]. <http://dis.um.es/~jbermudez/> [Consulta: 25 de Enero de 2008].

[**26**] UNIVERSIDAD DE MURCIA. *Evaluation in E-learning based on Semantic Web and NLP technologies*. <http://klt.inf.um.es/~oele/> [Consulta: 25 de Enero de 2008].

[**27**] UNIVERSIDAD DE MURCIA. *Modelo Entidad-Relación.* Apuntes de la asignatura Bases de Datos de 3º de Ingeniería Informática. Tema 3.

[**28**] UNIVERSIDAD DE STANFORD. *Ontology editor and knowledge-base framework* [en línea]. <http://protege.stanford.edu/> [Consulta: 25 de Enero de 2008].

[**29**] UNIVERSIDAD DE VIGO. *Proyecto Avatar* [en línea]. <http://avatar.det.uvigo.es/> [Consulta: 25 de Enero de 2008].

[**30**] W3C. Cascading Style Sheets [en línea]. <http://www.w3.org/Style/CSS/> [Consulta: 25 de Enero de 2008].

[**31**] W3C. Markup Validation Service [en línea]. <http://validator.w3.org/> [Consulta: 25 de Enero de 2008].

[**32**] W3C. *SPARQL Query Language for RDF* [en línea]. Noviembre de 2007. <http://www.w3.org/TR/2007/PR-rdf-sparql-query-20071112/> [Consulta: 25 de Enero de 2008].

[**33**] W3C. *SPARQL Query Results XML Format* [en línea]. Noviembre de 2007. <http://www.w3.org/TR/2007/PR-rdf-sparql-XMLres-20071112/> [Consulta: 25 de Enero de 2008].

[**34**] W3C. *XML Schema Part 2: Datatypes Second Edition* [en línea]. Octubre de 2004. Apéndice F. <http://www.w3.org/TR/xmlschema-2/#regexs/> [Consulta: 25 de Enero de 2008].

[**35**] WIKIPEDIA. *DVB* [en línea]. <http://es.wikipedia.org/wiki/DVB> [Consulta: 26 de Enero de 2008].

[**36**] WIKIPEDIA. *European Telecommunications Standards Institute* [en línea]. <http://es.wikipedia.org/wiki/ETSI> [Consulta: 26 de Enero de 2008].

[**37**] WIKIPEDIA. *Guía Electrónica de Programas* [en línea]. <http://es.wikipedia.org/wiki/EPG> [Consulta: 26 de Enero de 2008].

[**38**] WIKIPEDIA. *Lenguaje de Definición de Descripción* [en línea]. <http://es.wikipedia.org/wiki/Lenguaje_de_Definici%C3%B3n_de_Descripci%C3%B3n_%28DDL%29> [Consulta: 26 de Enero de 2008].

[**39**] WIKIPEDIA. *Moving Picture Experts Group* [en línea]. <http://es.wikipedia.org/wiki/MPEG> [Consulta: 26 de Enero de 2008].

[**40**] WIKIPEDIA. *Multimedia Home Platform* [en línea]. <http://es.wikipedia.org/wiki/MHP> [Consulta: 26 de Enero de 2008].

[**41**] WIKIPEDIA. *Organización Internacional para la Estandarización* [en línea]. <http://es.wikipedia.org/wiki/Organizaci%C3%B3n_Internacional_para_la_Estandarizaci%C3%B3n> [Consulta: 26 de Enero de 2008].

[**42**] WIKIPEDIA. *Tim Berners-Lee* [en línea]. <http://es.wikipedia.org/wiki/Tim_Berners-Lee> [Consulta: 26 de Enero de 2008].

[**43**] WIKIPEDIA. *TV-Anytime* [en línea]. <http://es.wikipedia.org/wiki/TV-Anytime> [Consulta: 26 de Enero de 2008].

[**44**] WIKIPEDIA. *XML Schema* [en línea]. <http://es.wikipedia.org/wiki/XML_Schema> [Consulta: 26 de Enero de 2008].

6. ANEXO

A1. MANUAL DE USUARIO

Despliegue de la aplicación

La aplicación web está incluida en un fichero WAR (*Web Archive, Archivo Web*). En este fichero están incluidas todas las librerías necesarias para la aplicación excepto la librería asociada a los servlet que será dependiente del servidor que vayamos a utilizar.

En el fichero WAR están incluidos dos ficheros de propiedades que deberán configurarse adecuadamente para que la aplicación funcione correctamente. Estos ficheros se llaman “mapeo.properties” y “urisOntologias.properties” ubicados en el directorio WEB-INF. Un fichero con extensión “.properties” tiene el formato “clave = valor”. Los valores son los datos que deben ser modificados para desplegar la aplicación.

Fichero “mapeo.properties”. En este fichero deben indicarse las ubicaciones de los tres ficheros de mapeos generados por la herramienta “Ontology Mapping”. A continuación se muestra, a modo de ejemplo, una posible configuración:

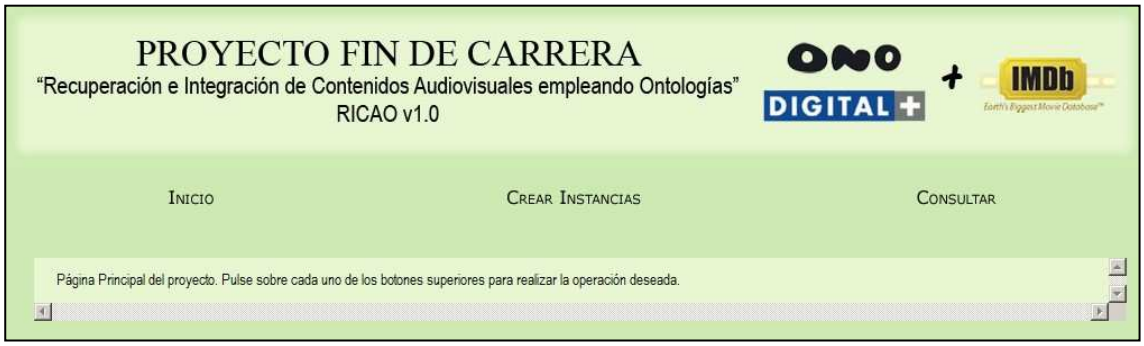
```
uriMapeoGlobalDigital=file:///C:/proyecto/PROGRAMACION/ontologias/mapeo_global_digital.xml
uriMapeoGlobalOno=file:///C:/proyecto/PROGRAMACION/ontologias/mapeo_global_ono.xml
uriMapeoGlobalIMDB=file:///C:/proyecto/PROGRAMACION/ontologias/mapeo_global_imdb.xml
```

Fichero “urisOntologias.properties”. En este fichero deben indicarse las ubicaciones de los cuatro ficheros OWL correspondientes a las tres ontologías asociadas a los portales de contenidos audiovisuales y a la ontología integradora. Además en este fichero deben indicarse los prefijos de cada una de las ontologías. A continuación se muestra, a modo de ejemplo, una posible configuración:

```
uriOntologiaDigitalPlus=C:/proyecto/PROGRAMACION/ontologias/digital.ow
l
uriOntologiaOno=C:/proyecto/PROGRAMACION/ontologias/ono.owl
uriOntologiaIMDB=C:/proyecto/PROGRAMACION/ontologias/imdb.owl
uriOntologiaGlobal=C:/proyecto/PROGRAMACION/ontologias/global.owl
uriOntologiaGlobalVacia=C:/proyecto/PROGRAMACION/ontologias/globalVaci
o.owl
prefijoOno=file:///C:/proyecto/OWL/proyecto.owl#
prefijoDigital=file:///C:/proyecto/OWL/proyecto.owl#
prefijoIMDB=file:///C:/proyecto/OWL/proyecto.owl#
prefijoGlobal=file:///C:/proyecto/OWL/proyecto.owl#
```

Interfaz de Usuario del Sistema RICAO v1.0

Como se explicó en el apartado “ARQUITECTURA DEL SISTEMA”, el sistema se puede ver como la unión de dos subsistemas, el sistema “Recuperación de Información” y el sistema “Integración de Información”. Un usuario tendrá el rol “cliente administrador” o el rol “cliente consultor” según acceda al primero o al segundo. La interfaz de usuario es la misma para ambos subsistemas, sin embargo, cada uno tiene su propio apartado en la interfaz.

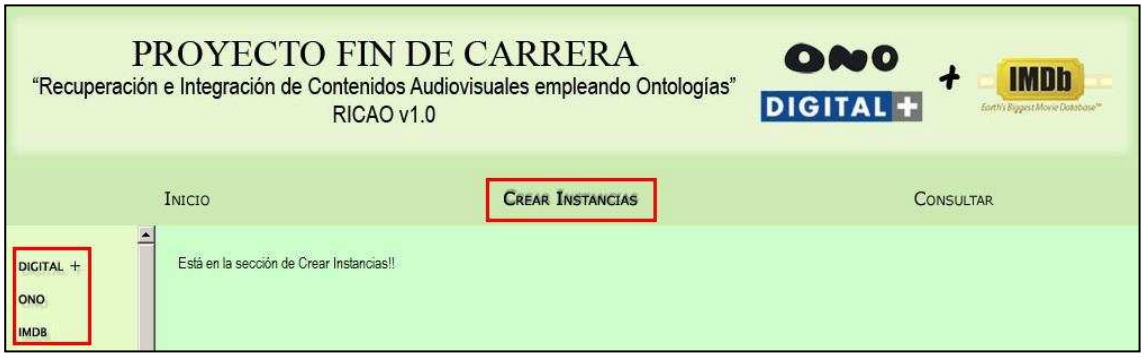


Captura 1. Página Inicial de la aplicación

En la captura anterior se muestra la página inicial de la aplicación. Se disponen de tres botones, uno para acceder a la página inicial llamado “INICIO”, otro para acceder a la funcionalidad ofrecida por el subsistema “Recuperación de Información” llamado “CREAR INSTANCIAS”, y otro para acceder a la funcionalidad ofrecida por el subsistema “Integración de Información” llamado “CONSULTAR”. Según se acceda a uno u otro el cliente tomará un rol determinado.

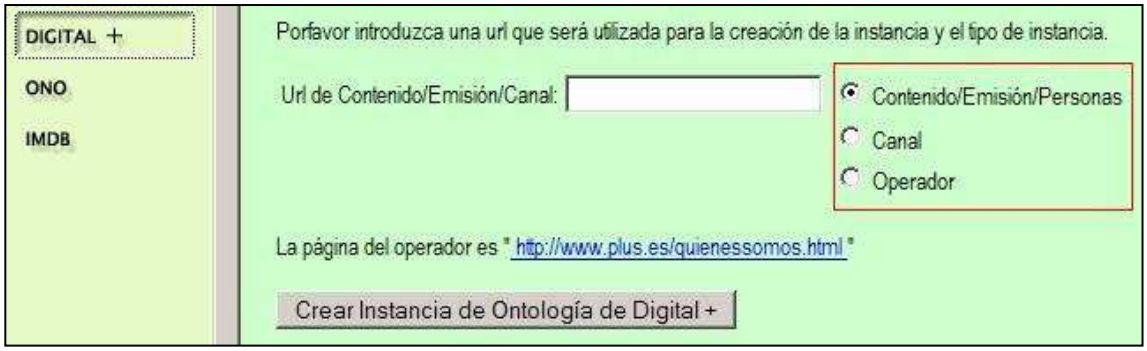
Subsistema “Recuperación de Información”

Una vez accedido al subsistema hay tres posibles opciones, recuperar información del portal de contenidos audiovisuales de Ono, de Digital Plus o de IMDb. Para acceder a cada una de las opciones se deberá hacer clic izquierdo de ratón sobre los botones situados en el recuadro rojo a la izquierda de la captura 2.



Captura 2. Subsistema “Recuperación de Información”

Opción 1) Digital Plus (http://www.plus.es). Recordemos que el objetivo del subsistema “Recuperación de Información” es obtener la información de los diferentes portales y almacenarla en su correspondiente ontología. El portal de Digital plus está organizado en páginas que muestran información acerca del Contenido, las Emisiones y las Personas involucradas, páginas que muestran información acerca de un Canal concreto y la página del Operador. En la interfaz de usuario se podrán indicar URLs de este tipo de páginas (ver recuadro rojo de la Captura número 3).



Captura 3. Creación de Instancias en Digital Plus

A este tipo de páginas se puede acceder de diferentes formas en el portal. Toda la información está disponible en la guía de contenidos que el operador la llama “I-GUÍA DIGITAL+: BUSCADOR DE PROGRAMACIÓN”. Para acceder a está guía deberá hacerse clic izquierdo de ratón sobre el botón que aparece en la página principal del portal llamado “I-Guía” (ver recuadro rojo de la captura número 4).

Captura 4. Guía de Programación del portal de Digital Plus

Al acceder a la guía nos aparece un simple buscador (ver captura número 5).

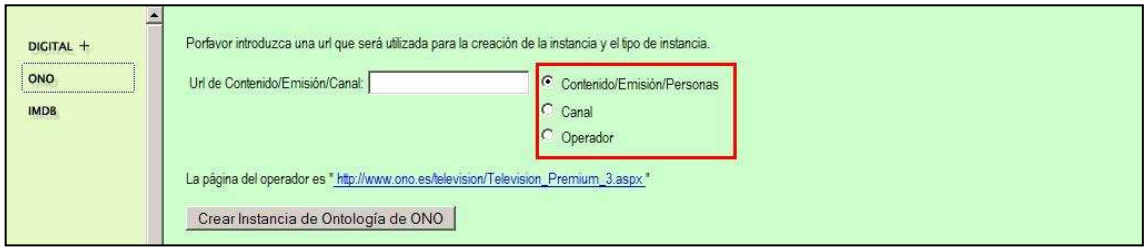
Captura 5. Buscador de Digital Plus

Para cada uno de los resultados obtenidos (ver captura número 6) se puede obtener la URL del Canal asociado y la URL de la página de Contenidos/Emisiones/Personas.

Captura 6. Resultados de una Búsqueda en Digital Plus

La página del operador, en principio, no cambia y es http://www.plus.es/quienessomos.html.

Opción 2) ONO (http://www.ono.es). ONO está estructurado de la misma forma que el portal de Digital Plus, es decir, en páginas que muestran información acerca del Contenido, las Emisiones y las Personas involucradas, páginas que muestra información acerca de un Canal concreto y la página del Operador. En la interfaz de usuario se podrán indicar URLs de este tipo de páginas (ver recuadro rojo de la Captura número 7).

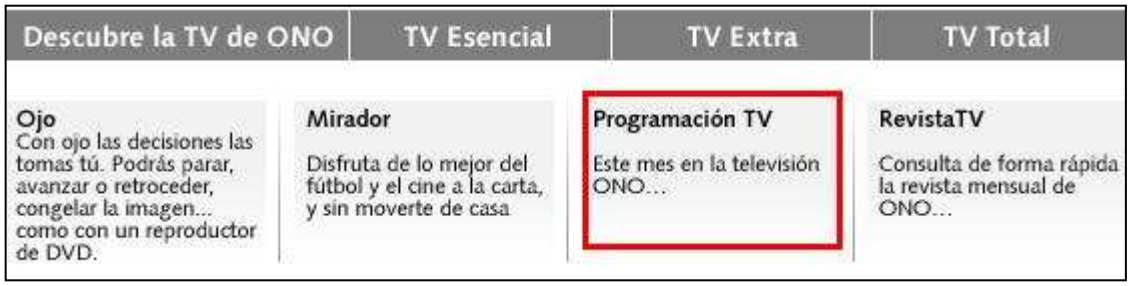


Captura 7. Creación de Instancias es ONO

Para acceder a los contenidos audiovisuales deberá hacer clic izquierdo de ratón sobre el botón que hace referencia a la televisión que aparece en la página principal del portal (ver recuadro rojo de la captura número 8).

Captura 8. Acceso a los contenidos audiovisuales. Paso 1

A continuación deberá hacer clic izquierdo de ratón sobre el botón llamado "Programación TV" (ver recuadro rojo de la captura número 9).



Captura 9. Acceso a los contenidos audiovisuales. Paso 2

ONO tiene en cada provincia canales diferentes. Por este motivo, a continuación se debe seleccionar la provincia de la que desea obtener información de contenidos audiovisuales.

Captura 10. Acceso a los contenidos audiovisuales. Paso 3

Una vez realizados los pasos anteriores, aparecerá un simple buscador, donde se pueden buscar contenidos por canal y categoría (ver captura número 11).

Captura 11. Buscador de ONO

Para cada uno de los resultados obtenidos (ver captura número 6) se puede obtener la URL de la página de Contenidos/Emisiones/Personas. La página de resultados contiene la información del canal que se va a almacenar en la ontología.

Captura 12. Resultados de una Búsqueda en ONO

Opción 3) IMDb (http://www.imdb.com). IMDb es el portal que mejor estructurado está de los tres. Como veremos a partir de una URL de un

contenido podremos acceder a toda la información necesaria (contenido, premios y personas involucradas). Para acceder a las URLs de los contenidos debemos hacer una búsqueda del mismo en la página principal del portal (ver captura número 13).

Captura 13. Buscador IMDb

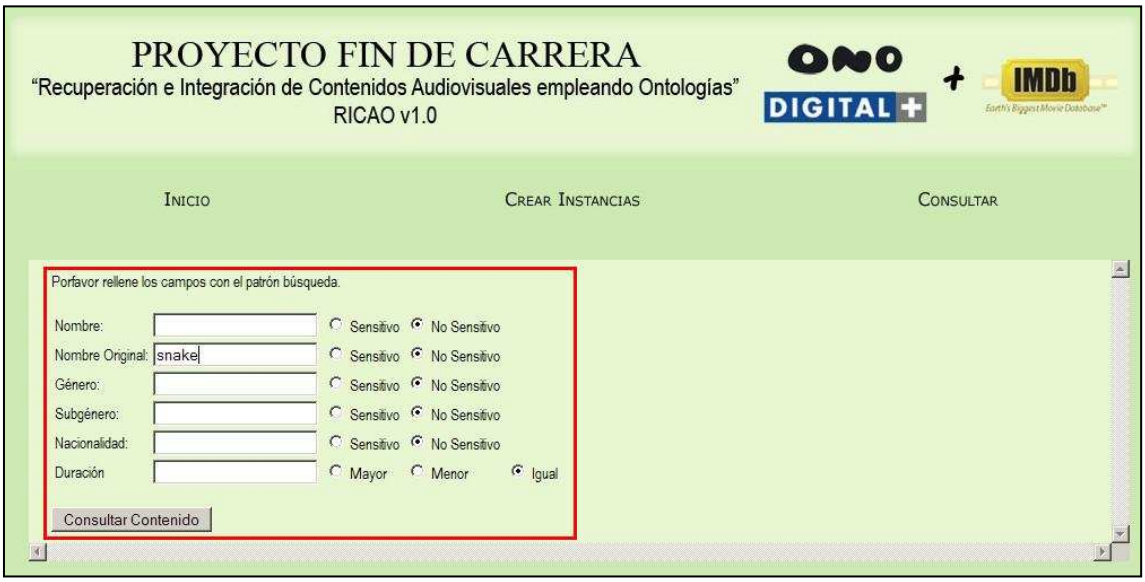
En la captura anterior hemos realizado una búsqueda con el nombre “LA HUELLA DEL SILENCIO”, los resultados obtenidos son URLs (ver en captura 14) que podemos utilizar en nuestro sistema.

Captura 14. Resultados de una Búsqueda en IMDb

Del documento HTML referenciado por un enlace de uno de los resultados obtenidos, se obtiene parte de la información utilizada. La sinopsis, los trailers, las personas involucradas, y los premios se obtienen añadiéndole a este enlace la ruta relativa “/synopsis”, “/trailers”, “/fullcredits#cast”, y “/awards” respectivamente.

Subsistema “Integración de Información”

La página inicial del subsistema muestra un simple formulario donde el usuario podrá indicar el patrón de búsqueda a utilizar. En los campos donde el valor sea una cadena se podrá indicar si la búsqueda quiere que se haga de forma sensitiva o no (se tendrán en cuenta o no las mayúsculas y minúsculas). Y en el campo duración, único campo numérico, si desea que los contenidos obtenidos como resultado tengan un valor mayor, menor o igual.



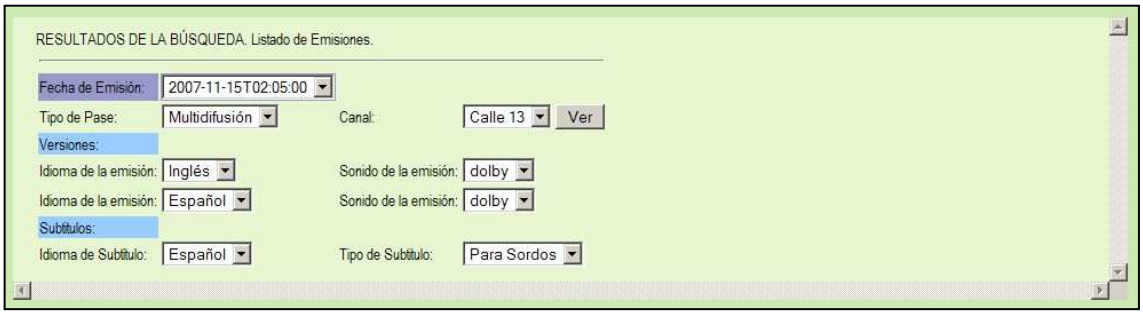
Captura 15. Subsistema “Integración de Información”

Una vez introducidos los valores de los campos a comparar, la búsqueda se lleva a cabo haciendo clic izquierdo de ratón sobre el botón “Consultar Contenido”.

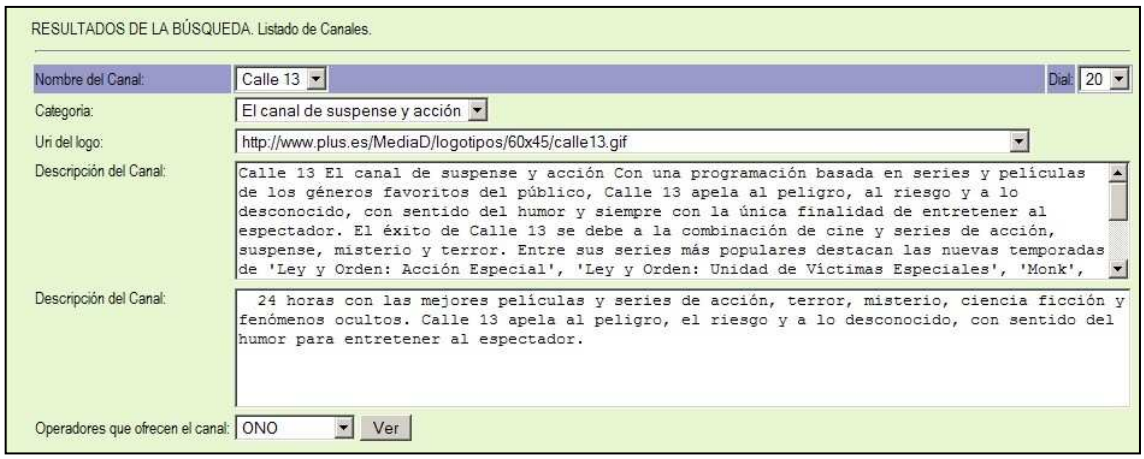
Los resultados obtenidos (ver captura número 16) son el resultado de la integración de la información de las diferentes ontologías previamente introducida en el subsistema “Recuperación de Información”.

Captura 16. Resultados de una consulta en el sistema RICAO v1.0

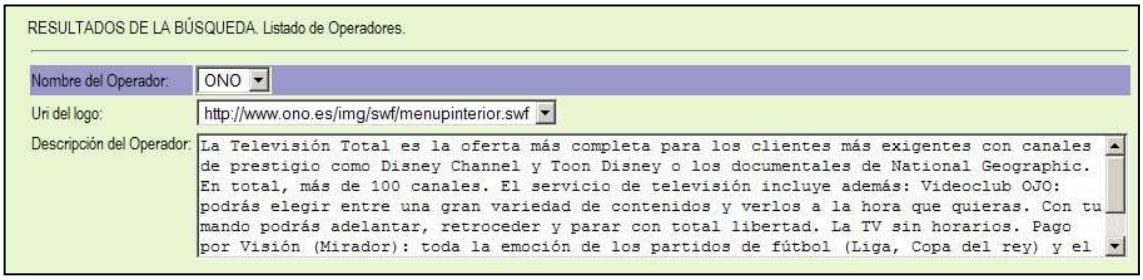
Para cada uno de los contenidos resultado se puede navegar a cada una de las personas involucradas (ver captura 20), a los premios (ver captura 21) y a las emisiones (ver captura 17). A su vez, a través de una emisión se podrá navegar hacia los canales (ver captura 18), a partir de los cuales se podrá navegar al operador (ver captura 21).



Captura 17. Listado de Emisiones de un contenido determinado



Captura 18. Canal asociado a una Emisión



Captura 19. Operador asociado a un Canal.

Captura 20. Persona asociada a un Contenido

Captura 21. Premios asociados a un Contenido