UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

SISTEMA PARA LA COMPILACIÓN AUTOMATIZADA DE CONTENIDO SINDICADO EN INTERNET

REPORTE DE TRABAJO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE
LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

SONIA BEATRIZ IZQUIERDO VILLAMIL

TUTORA
DRA. AMPARO LÓPEZ GAONA

2008

A mamá por enseñarme a vivir

A Amparo por enseñarme a hacer computación

ÍNDICE

INTRODUCCIÓN

El documento que aquí da inicio es un reporte del trabajo que realicé como Programador Analista de la Biblioteca Digital de la Universidad Nacional Autónoma de México desde Agosto del 2006 y hasta Enero del 2008.

Para construir este reporte -que constituye mi Proyecto de Titulación por Trabajo Profesional- fue necesario seleccionar un proyecto o solución específica que hubiera trabajado durante mi estancia en la Biblioteca Digital. Ésta no fue una tarea fácil debido a que mis labores dentro de dicha institución involucraron diversas y múltiples responsabilidades.

Mi área de especialización durante este periodo de trabajo profesional fue el desarrollo de Aplicaciones Web y Sistemas con Arquitectura Cliente-Servidor. Durante mi estadía en la Biblioteca Digital trabajé en 6 diferentes proyectos relacionados con el área.

De los 6 proyectos que desarrollé, seleccioné el primero con el que tuve el gusto de trabajar al que llamé *Agregador Web BiDi*.

Este documento dará entonces un detalle extensivo acerca de los orígenes de este proyecto, su planeación, desarrollo y construcción con el objetivo de documentar la experiencia que obtuve de mi trabajo profesional dentro de esta importante institución de la UNAM.

Problemática

Como parte de una renovación general de los Servicios Web de la Biblioteca Digital de la UNAM se decide agregar nuevas aplicaciones a la Página Web de la Institución.

Uno de estos aplicativos era un cuadro de noticias en el que se desplegara información de diferentes fuentes como periódicos nacionales e internacionales, proveedores de la UNAM, noticias internas de la UNAM e incluso noticias internas de la Biblioteca Digital (BiDi). Los usuarios de la Página Web de BiDi deberían de poder ver un resumen de toda la información en la página y además tener la opción de consultar la noticia completa visitando la página web que la había publicado.

Objetivo del Sistema

Es entonces que se decide programar un Sistema que permitiera agregar un cuadro de noticias a la Página Web de BiDi que cumpliera con la funcionalidad mencionada anteriormente.

Este Sistema, al que se le bautizó como *Agregador Web BiDi*, debería de poder recolectar noticias e información de diferentes fuentes definidas por el usuario, ordenarlas, almacenarlas y posteriormente desplegarlas en la página web.

Agregador Web BiDi debería de ser fácil de usar y susceptible a actualizaciones futuras.

Herramientas

Como se mostrará en secciones posteriores, el intercambio y la recolección de contenido noticioso a través de Internet no es una necesidad reciente por lo que existen muchas herramientas que facilitan enormemente esta tarea.

La primera herramienta es la completa estandarización que existe para publicar contenido noticioso a través de Internet. El estándar RSS –que será explicado con mayor detalle posteriormente– define esquemas basados en XML que permiten dar una estructura unificada a los documentos de difusión de contenido noticioso.

El lenguaje de programación seleccionado para construir la aplicación fue *Java*. La principal razón es que este lenguaje cuenta con muchas herramientas para resolver problemas que tiene que enfrentar el aplicativo como conexiones remotas, recepción y envío de peticiones *http* y análisis de documentos XML.

Estructura

Este trabajo comprende la documentación del proceso de creación del *Agregador Web BiDi* y está estructurado como sigue:

Capítulo 1. Introducción a la Sindicación de Contenido.

Este capítulo provee las bases teóricas de la Sindicación de Contenido dando la definición del concepto así como su historia y orígenes resaltando cuáles fueron las necesidades que dieron como resultado la concepción del mismo.

Capítulo 2. Estándares para la Sindicación de Contenido.

Después de que en el capítulo 1 se define lo que es la Sindicación de Contenido, este capítulo describe los resultados que se presentaron a raíz de la concepción de la Sindicación de Contenido en el mundo de Internet. A lo largo de este capítulo se documenta ampliamente la especificación de los principales estándares RSS que unifican la Sindicación de Contenido actualmente.

Capítulo 3. Especificación del Sistema.

Haciendo uso del marco teórico provisto por los capítulos 1 y 2, el capítulo 3 da inicio a la documentación de la creación del Sistema. Para la construcción de *Agregador Web BiDi* se siguió la metodología de Ingeniería de Software de cuatro fases: especificación, análisis, diseño e implementación. El capítulo 3 constituye la Especificación del Sistema.

Capítulo 4. Análisis del Sistema.

El capítulo 4 comprende el Análisis del Sistema lo que incluye una versión más computacional del contenido de la Especificación descrita en el capítulo 3. Dentro del Análisis se define la estructura general de paquetes y clases que tendrá el Sistema así como los casos de uso que se resolverán.

Capítulo 5. Diseño del Sistema.

Dentro del Diseño del Sistema se aumenta el detalle que el Análisis provee. En este capítulo se definen específicamente todas las clases que conforman la aplicación así como los métodos y atributos de cada una de ellas. Para documentar esta información, este capítulo contiene los diagramas de clases de todos los paquetes. En éstos se explica la forma en la que cada módulo logrará cumplir con su función.

CAPITULO 1

Introducción a la Sindicación de Contenido

Este capítulo introduce el concepto de Sindicación de Contenido. Debido principalmente a que la palabra “sindicación” no es una palabra oficial del español sino un anglicismo, el término Sindicación de Contenido suele ser confundido o mal interpretado.

Sin embargo, y a pesar de lo áspero que pueda parecer el tema de primera instancia, la Sindicación de Contenido promete ser, entre otras cosas, la base para la evolución de Internet y la creación de una Web Semántica, por lo que hoy en día su estudio y desarrollo ocupan un lugar primordial en la tecnología de la información.

¿Qué es la Sindicación de Contenido?

El término sindicación es un anglicismo que proviene de “syndication”, habitual terminología anglosajona de los medios de comunicación. De manera estricta, se desaconseja su uso en español, sin embargo es un término que se ha hecho muy popular dentro de la jerga técnica a partir del auge del “Content-Syndication” o Sindicación de Contenido como normalmente se le conoce en español.

Se le llama entonces Sindicación de Contenido a la redifusión de contenidos informativos o lúdicos de un emisor original por otro, que adquiere los derechos gracias a un contrato o licencia. La Sindicación de Contenido, o en términos lingüísticamente más recomendables “Redifusión de Contenido”, es un concepto aplicado a varias situaciones: Sindicación Televisiva, Sindicación de Prensa, Sindicación de Radio y Sindicación de Contenido Web.

Sindicación de Contenido Web

A lo largo de este documento, al hacer referencia a Sindicación de Contenido, se estará hablando específicamente de Sindicación de Contenidos Web.

La Sindicación de Contenido comprende todo el contenido de un sitio web que se encuentra disponible para ser usado por otros servicios. El Contenido Sindicado, también conocido como fuente (o “feed” en inglés), puede estar compuesto tanto por el contenido en sí como por metadatos (es decir, información acerca del contenido mismo).

El Contenido Sindicado puede estar compuesto por elementos de diferentes características que pueden ser desde muy sencillos hasta muy complejos –de acuerdo a la elección del sitio web que lo publica. Podemos tener contenido formado por enlaces a documentos y encabezados de noticias e historias y tener contenidos mucho más complejos que abarquen toda la información que un sitio web ofrece.

La Sindicación de Contenido permite a los usuarios acceder a un sitio web desde múltiples dispositivos y ser notificado de actualizaciones y novedades de una gran variedad de servicios. Por otro lado, resulta una herramienta muy poderosa para los desarrolladores de Servicios Web al ser económica, sencilla, rápida y versátil.

Un poco de historia

La Sindicación de Contenido Web está estandarizada desde hace más de una década. La gran mayoría de estos estándares están basados en XML (Extensible Markup Language).

XML es una especificación de propósito general para crear lenguajes de marcado personalizados. Es clasificado como un lenguaje extensible debido a que brinda a sus usuarios la posibilidad de definir sus propios elementos. Su principal propósito es facilitar la difusión de datos estructurados a través de diferentes sistemas de información.

A pesar de ser varios los estándares de Sindicación de Contenido basados en XML, el estándar conocido como RSS es hasta la fecha y desde sus orígenes el más popular. A grandes rasgos, RSS es una familia de estándares de Sindicación de Contenido usada comúnmente para aplicaciones Web como *"blogs"*, noticias, y cualquier contenido que tienda a actualizarse constantemente. Hablaremos un poco de los orígenes de RSS para entender sus dimensiones al día de hoy.

Antecesores de RSS: MCF y RDF

RSS fue creado formalmente en 1999 y es un amplio conjunto de estándares que ha sido renovado y modificado constantemente desde su nacimiento.

Los primeros orígenes de RSS se remontan a 1995, con el trabajo de Ramanathan V. Guha. Guha desarrolló un sistema llamado *Meta Content Framework* (MCF). El objetivo general de este sistema era el de describir objetos, sus atributos y las relaciones entre éstos.

MCF fue un proyecto de investigación experimental fundado por Apple© y arrojó como resultado una aplicación llamada ProjectX que posteriormente fue renombrada como HotSauce. A finales de 1996, algunos cientos de sitios estaban

creando archivos MCF que contenían la descripción general de sus servicios, y HotSauce permitía a los usuarios navegar a través de estas descripciones.

HotSauce fue popular pero meramente experimental, y cuando Steve Jobs' regresó a la gerencia de Apple© en 1997 terminó con mucha de la investigación que esta corporación llevaba a cabo y Guha abandonó sus tareas en la empresa para unirse a Netscape©.

Estando en Netscape©, Guha conoció a Tim Bray, uno de los pioneros de XML, e inició la exportación de MCF hacia un nuevo formato basado en XML. Este proyecto se convirtió más tarde en el *Resource Description Format* (RDF).

RDF es un lenguaje de propósito general para representar información en Internet. Está diseñado específicamente para la representación de metadatos y relaciones entre objetos. En su forma más extendida, RDF es la base para el concepto conocido como Web Semántica, la visión del World Wide Web Consortium (W3C) para una red de información que las computadoras puedan entender.

Channel Definition Format

Para finales de 1997, Microsoft© no había pasado por alto la experiencia de HotSauce y continuó el desarrollo de MCF para la descripción de Sitios Web y creó el *Channel Definition Format* (CDF).

CDF está basado en XML y puede describir la popularidad, agenda, logotipo y metadatos acerca de un sitio web. Fue introducido en Internet Explorer 4.0 y posteriormente en el escritorio de Windows XP donde proporcionó las bases de lo que posteriormente se llamó Active Desktop.

Para 1999, MCF estaba fuertemente posicionado en XML y RDF estaba tomando fuerza. Microsoft© y Netscape© pasaban por un periodo de competencia muy fuerte y ambas compañías estaban a punto de lanzar las nuevas versiones de sus navegadores.

Primera aparición de RSS

Netscape© lanzó entonces un sitio web llamado "*My Netscape Network"*, y con este sitio, lanzó también RSS.

RSS como abreviación para *RDF Site Summary*, permitía a este sitio web desplegar información proveniente de otros Sitios Web en una sola página. El usuario podía entonces personalizar "*My Netscape Page"* -una página individual y personalizable dentro del sitio "*My Netscape Network"*- para que ésta contuviera noticias de cualquier sitio en el que estuviera interesado y que tuviera un archivo RSS disponible. Este servicio era, básicamente, un sitio web fruto de lo que HotSauce y CDF habían dejado como enseñanza. "*My Netscape Network"* fue un gran éxito a nivel internacional.

"My Netscape Network" trajo múltiples beneficios debidos principalmente a que, de un momento a otro, Netscape© contaba con cantidades enormes de contenido que resultaban de interés para los usuarios y que además habían adquirido de manera gratuita. El usuario también era beneficiado, podía tener la información de sus sitios preferidos resumidos en una sola página y este era un servicio novedoso y atractivo.

Todas estas cualidades aunadas a la relativa sencillez del mismo estándar RSS demostraron ser tan útiles y prometedoras que RSS no fue único de Netscape© por mucho tiempo. Convertir RSS a HTML es simple, por lo que muchos sitios basados en RSS empezaron a surgir en la red. Sitios Web como *slashdot.com* incorporaron fuentes RSS como remplazos de sus propios formatos internos y desarrolladores de los principales lenguajes de *scripting* crearon rápidamente herramientas sencillas para desplegar fuentes RSS.

RSS se convirtió entonces en una tecnología sumamente prometedora.

Evolución de los estándares

El primer estándar RSS fue el estándar RSS 0.9 desarrollado por Dan Libby, investigador de Netscape©. Este estándar, completamente basado en RDF, resultó ser muy completo, pero demasiado complejo para los usuarios que se esperaba, tendría en ese entonces.

Fue entonces que surgió, casi de manera inmediata, RSS 0.91 que incorporaba nomenclatura familiar para los usuarios obtenida de los más conocidos lenguajes de scripting de ese momento y que, además, era completamente independiente de RDF. Debido a este cambio drástico, Dan Libby cambió el significado del término RSS –originalmente las siglas de *RDF Site Summary*- y lo definió como una palabra en sí.

Sin embargo, Netscape© no lanzó más versiones de RSS. Luego de que "*My Netscape Network"* fue cerrada, el equipo de investigación de RSS dentro de la corporación también fue disuelto. Cuando hubo la necesidad de trabajar en nuevas versiones de RSS, fue la comunidad de desarrolladores independientes la que tomó la tarea y pronto, este estándar se dividió en dos bandos.

El primer bando, encabezado por Rael Dornfest –integrante de O'Reilly's-, quería producir una extensión para el estándar que permitiera agregar nuevas funcionalidades. Su plan era reintroducir RDF y hacer uso de espacios de nombres de XML para lograrlo.

El otro bando se encontraba liderado por Dave Winer -CEO de Userland Software©- quien difería de la propuesta de O'Reilly's argumentando que ésta solamente complicaría el estándar. Su propuesta personal era que, a toda costa, se debería de luchar por mantener la simplicidad del estándar tal y como la versión 0.91 lo había planteado.

La primera ramificación

A pesar de las diferencias, en Diciembre de 2000 se liberó RSS 1.0. Éste incluía el uso de espacios de nombres XML, y regresaba al modelo basado en RDF. Sin embargo, dos semanas después del lanzamiento de RSS 1.0, Dave Winer publicó

RSS 0.92 que, como era de esperarse, era una opción mucho más sencilla que además, prescindía de RDF.

RSS estuvo entonces dividido en dos estándares vigentes por casi dos años. Se tenía la versión simple (RSS 0.92) y la versión compleja (RSS 1.0). Para los usuarios finales de las fuentes RSS, sin embargo, esto no fue un inconveniente debido a que ambos estándares eran compatibles

La segunda ramificación

A mediados del 2002, la comunidad RSS volvió a sufrir una ramificación. Irónicamente, esta ramificación se debió a los esfuerzos realizados para unir las versiones 0.9x y 1.0 en una nueva versión que se llamaría RSS 2.0.

Una vez más, el equipo de desarrolladores se dividió en dos corrientes. Por un lado, Dave Winer y algunos otros seguían creyendo en la importancia de la simplicidad del estándar y por otro lado, había un gran grupo de desarrolladores que seguían impulsando la idea de volver a introducir RDF como base de RSS.

Finalmente, después del lanzamiento de RDF 2.0 el 16 de Septiembre del 2002 por el grupo de Dave Winer, el grupo de seguidores de RDF 1.0 decidió desconocer el nuevo estándar y pensaron en publicar su propia versión que se llamaría RSS 1.0-based specification. Sin embargo, este nunca fue publicado.

Hasta la fecha, RSS 2.0 es el estándar más usado de RSS. Sin embargo, aún existen aplicaciones que trabajan sobre RSS 1.0. Por ahora, la solución ha sido que todas las aplicaciones construidas para trabajar con estos estándares, sean capaces de soportar ambos.

Beneficios de la Sindicación de Contenido

La Sindicación de Contenido hoy en día es algo que se ve en cualquier sitio web más o menos reconocido. Esta es la razón por la cual es un tema tan importante y controvertido.

Pero, ¿Por qué la Sindicación de Contenido se ha vuelto tan popular?, ¿Qué ventajas trae al sitio web que la proporciona?, ¿Por qué este tema merece estudios tan especializados? Estos son algunos de los beneficios inmediatos que brinda la Sindicación de Contenido:

1. Incrementa la cantidad de tráfico al sitio web.
2. Incrementa la presencia del sitio web en la conciencia de los usuarios.
3. Incrementa el posicionamiento del sitio web en los resultados arrojados por los motores de búsqueda en Internet.
4. Permite establecer relaciones más sólidas entre comunidades de Sitios Web.
5. Mejora la relación entre el sitio web y el usuario.

Conclusiones

Como parte de la explosión en la evolución de Internet y los Servicios Web han surgido muchas nuevas necesidades que han obligado a compañías de Software y desarrolladores a establecer nuevos estándares y construir nuevas bases de desarrollo. Los estándares RSS surgen como una solución a una necesidad imperante que fue la propagación ordenada y eficiente de información cambiante a través de los medios de Internet.

En los capítulos subsecuentes se desglosa con detalle el proceso de Ingeniería de Software de una aplicación para el manejo y control de Contenido Sindicado; el contenido teórico presentado en este capítulo provee las bases necesarias para seguir detallando los orígenes y resultados de este aplicativo.

CAPITULO 2

Estándares para la Sindicación de Contenido

Como se mencionó en la sección anterior, se han creado varios estándares para la Sindicación de Contenido. A lo largo de este capítulo, se ahondará más en la especificación de 4 de los principales estándares basados en XML:

- RSS 0.91
- RSS 0.92
- RSS 1.0
- RSS 2.0

RSS 0.91

RSS 0.91 es el estándar más viejo de Sindicación de Contenido basado en XML y a pesar de que su creación data de 1999, actualmente todavía existen fuentes que usan este estándar para definir su Contenido Sindicado.

La principal aportación de RSS 0.91 fue la eliminación del uso de elementos RDF para construir archivos RSS, esto permitió que usuarios y desarrolladores menos experimentados pudieran hacer uso de esta nueva tecnología.

Características Generales

Las características que introdujo RSS 0.91 y que lo hicieron diferente a todos los estándares que existían en el momento de su lanzamiento se resumen en 5 puntos importantes:

1. Validación XML completa.

   Los archivos RSS 0.91 son archivos XML 100% válidos. Para lograr esto, se obligó el uso de la etiqueta DOCTYPE dentro de los archivos, la cual valida el documento con el DTD de RSS 0.91 correspondiente.
   Con esta característica, los archivos RSS 0.91 no solamente deben estar *bien formados*, sino también ser válidos con respecto a su DTD.

2. Eliminación de etiquetas con mezcla de contenido.

   Las etiquetas podrán tener contenido de un solo tipo. Es decir, cada etiqueta podrá o bien contener sub etiquetas o bien contener texto, pero no ambos.

3. Incorporación de nuevas etiquetas.

   Se incorporó a la biblioteca de etiquetas una considerable cantidad de nuevos elementos que permiten a los sitios que hacen uso del estándar, poder dar mayor especificación a su Contenido Sindicado.

4. Eliminación de referencias RDF.

   RDF presenta dos inconvenientes. El primero es que está más enfocado a la definición de meta datos que a la definición de Contenido Sindicado y el segundo es que es muy difícil para el usuario o desarrollador promedio

entenderlo y usarlo adecuadamente. Por estas razones, se eliminó por completo el uso de RDF y se apegó el estándar hacia el enfoque general de XML.

Especificación

A continuación se presenta un listado de las etiquetas de RSS 0.91 en orden alfabético.

Etiqueta `<channel>`

Descripción	Contiene la información acerca de un canal en particular. Todo lo concerniente a un canal individual debe de estar contenido dentro de esta etiqueta.
Atributos	Ninguno

Sub elementos requeridos	`<description>`
	`<language>`
	`<link>`
	`<title>`

Sub elementos opcionales	`<copyright>`
	`<docs>`
	`<image>`
	`<item>`
	`<lastBuildDate>`
	`<managingEditor>`
	`<pubDate>`
	`<rating>`
	`<skipDays>`
	`<skipHours>`
	`<textInput>`
	`<webMaster>`

Etiqueta `<copyright>`

Descripción	Cadena que contiene la leyenda de los derechos de autor del contenido.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta <day>

Descripción	El día de la semana escrito en inglés.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta <description>

Descripción	Texto plano que contiene la descripción de un ítem <item>, un canal <channel> o una imagen <image>.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta <docs>

Descripción	Esta etiqueta debe de contener una URL que haga referencia a documentación que describa con más detalle el canal.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta <!DOCTYPE>

Descripción	Constituye el identificador del tipo de documento. Esta etiqueta XML identifica donde encontrar la definición para este formato.
Atributos	Es necesario incluir la ubicación del archivo DTD que validará el archivo. Preferentemente, debe de ser alguno de los siguientes valores: 1. rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN" "" 2. rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN" "http://my.netscape.com/publish/formats/rss-0.91.dtd"
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta <height>

Descripción	Especifica la altura de una imagen <image>. Debe de ser un valor entero.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta <hour>

Descripción	Especifica una hora del día. Debe de ser un entero entre 0 y 23.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta <image>

Descripción	Especifica una imagen asociada con un canal <channel>.
Atributos	Ninguno
Sub elementos requeridos	<url> <link> <title>
Sub elementos opcionales	<description> <width> <height>

Etiqueta <item>

Descripción	Describe un elemento asociado a un canal <channel>. El elemento debe de representar una Página Web o una sección dentro de una Página Web. Debe de tener un URL único asociado a él. Cada elemento <item> debe de contener un título <title> y una liga o referencia <link>. Una descripción <description> es opcional.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<language>`

Descripción	Especifica el idioma del canal `<channel>`.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<lastBuildDate>`

Descripción	Especifica la última fecha en la que fue modificado el canal `<channel>`.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<link>`

Descripción	Etiqueta que almacena una liga o URL a un sitio web o a una sección dentro de un sitio web.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<managingEditor>`

Descripción	Contiene la dirección de correo (e-mail) del editor del sitio web o de la persona que debe de ser contactada para asuntos relacionados con la edición del Contenido Sindicado.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<name>`

Descripción	Contiene el nombre de un objeto correspondiendo al atributo *name* de una etiqueta `<input>` de HTML. Esta etiqueta aplica solamente a dentro de la etiqueta `<textinput>`.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<pubDate>`

Descripción	Fecha de publicación del canal.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<rating>`

Descripción	Especifica la audiencia del canal, es decir, que tanta gente consulta el canal cada cierto periodo de tiempo. La cantidad especificada dentro de esta etiqueta debe de estar respaldada por algún sitio web dedicado a auditar la audiencia de Servicios Web.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<rss>`

Descripción	Identifica el inicio y el fin del contenido RSS.
Atributos	• `version` – En este caso, debe de ser 0.91.
Sub elementos requeridos	• `channel` – Cada archivo debe de contener, por lo menos, un canal.
Sub elementos opcionales	Ninguno

Etiqueta `<skipDays>`

Descripción	Contiene una lista de días de la semana (representados por la etiqueta `<day>`) que indican los días de la semana en los que el canal no será actualizado.
Atributos	Ninguno
Sub elementos requeridos	• `day` – Es necesario especificar por lo menos un día de la semana en caso de incluir la etiqueta `<skipDays>`.
Sub elementos opcionales	Ninguno

Etiqueta `<skipHours>`

Descripción	Contiene una lista de horas (representadas por la etiqueta `<hour>`) que indica las horas del día en las que es muy poco probable o imposible que el canal sea actualizado. Si esta etiqueta no es usada, se asume que el canal será actualizado cada hora.
Atributos	Ninguno
Sub elementos requeridos	• `hour` – Es necesario especificar por lo menos una hora del día en caso de incluir la etiqueta `<skipHours>`.
Sub elementos opcionales	Ninguno

Etiqueta `<textinput>`

Descripción	Representa un campo editable de texto con el propósito de que los usuarios puedan enviar información que resulte útil para el editor del Contenido Sindicado. Este elemento debe de tener un título, una liga (a la aplicación que trabajará con la información recaudada), una descripción y un nombre que será usado como el atributo *name* de una etiqueta `<input>` de HTML.
Atributos	Ninguno
Sub elementos requeridos	• `title` – Cadena que identifique a este elemento. • `link` – Liga o URL a la aplicación que trabajará con la información recaudada. • `description` – Descripción que contenga breves instrucciones acerca de qué es lo que se espera que el usuario haga con este componente. • `name` – Nombre que será usado como el atributo *name* de una etiqueta `<input>` de HTML.

Etiqueta `<title>`

Descripción	Cadena de texto que sirve como identificador para un recurso determinado. Cuando esta etiqueta es usada dentro de `<item>`, será el nombre de la liga `<link>` del elemento `<item>`. Cuando esta etiqueta es usada dentro de `<image>`, será el texto asociado a la descripción de la imagen. Cuando esta etiqueta es usada dentro de `<channel>`, se convierte en el título del canal. Cuando esta etiqueta es usada dentro de `<input>`, se convierte en el título o descripción del elemento.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<url>`

Descripción	Representa una ubicación desde la que se espera obtener recursos. Esta etiqueta es diferente de la etiqueta `<link>` debido a que `<link>` especifica una URL en la que se espera que el usuario pueda hacer click para ser re direccionado a un sitio web.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<webMaster>`

Descripción	Dirección de correo electrónico (e-mail) del administrador del sitio web o de la persona que deberá de ser contactada en caso de tener problemas técnicos con el sitio web.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<width>`

Descripción	Especifica el ancho de una imagen `<image>`. Debe de ser un valor entero.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<?xml?>`

Descripción	Identifica al documento como un documento XML y especifica la codificación usada. Esta etiqueta debe de ser la primera línea del documento.
Atributos	• `version` – Debe de ser 1.0. • `encoding` – La codificación usada en el documento.
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

La figura 2.1 presenta un resumen, en forma de diagrama de árbol, de la estructura de etiquetas mencionada anteriormente.

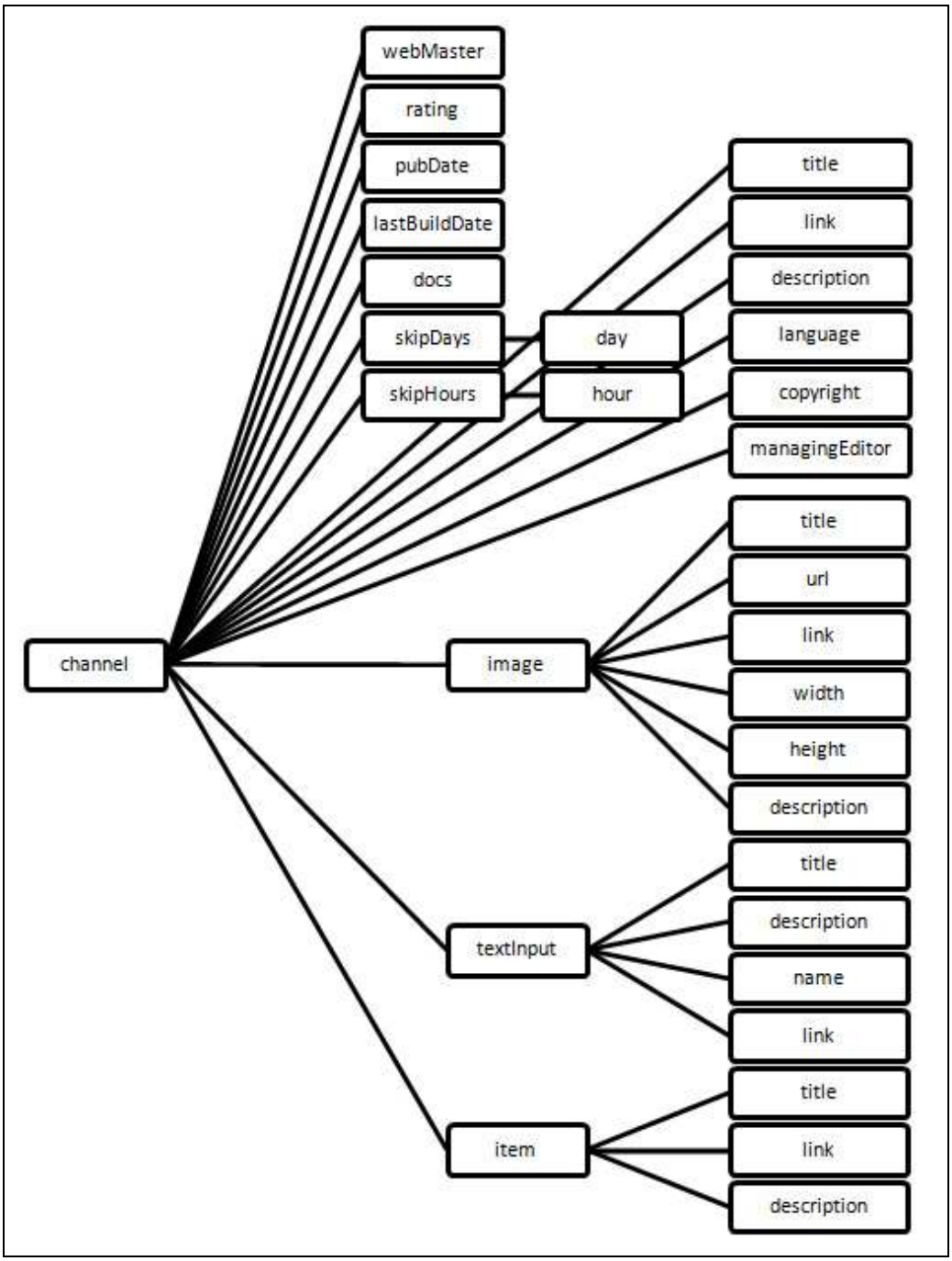

Figura 2. 1 - Elementos de RSS 0.91

Ejemplo

Siguiendo la estructura descrita anteriormente, la Figura 2.2 muestra un Ejemplo de cómo puede construirse un archivo RSS 0.91 haciendo uso correcto de todos los elementos definidos.

```
<?xml version="1.0"?>
<!DOCTYPE rss SYSTEM "http://my.netscape.com/publish/formats/rss-0.91.dtd">
<rss version="0.91">
    <channel>
        <pubDate>03 Apr 08 1500 GMT</pubDate>
        <lastBuildDate>03 Apr 08 1500 GMT</lastBuildDate>
        <docs>http://backend.userland.com/rss091</docs>
        <description>Ejemplo de fuente RSS 0.91</description>
        <title>Ejemplo de RSS0.91</title>
        <link>http://www.urldeejemplo.com.mx/index.html</link>
        <language>es-mx</language>
        <managingEditor>editor@urldeejemplo.com.mx </managingEditor>
        <webMaster>webmaster@urldeejemplo.com.mx </webMaster>
        <image>
            <title>Ejemplo de RSS 0.91</title>
            <url>http://www.urldeejemplo.com.mx/imagenes/logo.gif</url>
            <link>http://www.urldeejemplo.com.mx/index.html</link>
            <width>88</width>
            <height>31</height>
            <description>El Mundo a través de la Sindicación de
            Contenido</description>
        </image>
        <skipHours>
            <hour>6</hour>
            <hour>7</hour>
        </skipHours>
        <skipDays>
            <day>Sunday</day>
        </skipDays>
        <rating>(PICS-1.1 "http://www.rsac.org/ratingsv01.html" l gen true comment
            "RSACi North America Server" for "http://www.rsac.org" on
            "1996.04.16T08:15-0500" r (n 0 s 0 v 0 l 0))</rating>
        <item>
            <title>Primer Elemento</title>
            <link>http://www. urldeejemplo.com.mx/001.html</link>
            <description>Esta es la descripción del primer elemento.</description>
        </item>
        <item>
            <title>Segundo Elemento</title>
            <link>http://www. urldeejemplo.com.mx/002.html</link>
            <description>Esta es la descripción del segundo elemento.</description>
        </item>
        <textInput>
            <title>Búsqeuda</title>
            <description>Busca en la Base de Datos de Archivos</description>
            <name>consulta</name>
            <link>http://www. urldeejemplo.com.mx/busqueda.cgi</link>
        </textInput>
    </channel>
</rss>
```

Figura 2. 2 - Ejemplo de un Archivo RSS 0.91

RSS 0.92

RSS 0.92 fue liberado en Diciembre del 2000. Escrito por Dave Winer (Userland Software's ©), fue una expansión de RSS 0.91 agregando solamente 4 elementos adicionales y haciendo algunas modificaciones en las numerosas restricciones puestas originalmente a la especificación.

A finales del 2003, *Syndic8* publicó que cerca del 30% del Contenido Sindicado en la Web usaba RSS 0.92. De cualquier manera, esta cifra no resulta tan impactante tomando en cuenta que todas las fuentes en RSS 0.91 también están en RSS 0.92 y mucho del contenido construido supuestamente en RSS 0.92 no hace uso de ninguna de las mejoras que RSS 0.92 trajo sobre la versión 0.91.

Actualizaciones

Las actualizaciones que hizo RSS 0.92 sobre RSS 0.91 se resumen en:

- La etiqueta `<language>`, antes obligatoria como sub elemento de la etiqueta `<channel>`, ahora es opcional.
- Todos los sub elementos de la etiqueta `<item>` son ahora opcionales.
- En RSS 0.91 varios elementos estaban restringidos a 500 o 100 caracteres. Ahora se eliminaron todas las limitaciones en cuanto a la longitud de los elementos.
- En RSS 0.91 no podían existir más de 15 elementos `<item>` por canal, ahora esta cantidad es ilimitada.

Especificación

A continuación se mencionan solamente los elementos nuevos que fueron agregados a la versión 0.92. Todos los elementos de RSS 0.91 permanecen vigentes en esta nueva versión.

Etiqueta `<source>`

Descripción	Es un subelemento opcional de la etiqueta `<item>`. Contiene el nombre del canal RSS del que proviene el `<item>`. El propósito de este nuevo elemento es el de dar crédito a las ligas externas usadas dentro de una fuente. Permite hacer publicidad a el origen del contenido desplegado.
Atributos	• `url` – La liga a el archivo XML que contiene el canal al que se hace referencia.
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<enclosure>`

Descripción	Es un subelemento opcional de la etiqueta `<item>`. Hace referencia a un recurso externo que puede ser un archivo de cualquier tipo MIME válido.
Atributos	• `url` – La ubicación del recurso. • `length` – Qué tan grande es el archivo en bytes. • `type` – El tipo MIME del archivo.
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<category>`

Descripción	Es un subelemento opcional de la etiqueta `<item>`. Asocia una categoría al elemento en cuestión. La categoría puede ser determinada por el autor.
Atributos	• `domain` – Atributo opcional que hace referencia a algún documento o liga que contenga la jerarquía completa de clasificación usada por el sitio web o editor del Contenido Sindicado.
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

Etiqueta `<cloud>`

Descripción	Es un subelemento opcional de la etiqueta `<channel>`. Especifica un Servicio Web que soporte la interfaz *rssCloud* la cual puede ser implementada en *http-POST*, *XML-RPC* o *SOAP 1.1*. El propósito de esta etiqueta es permitir que los procesos basados en *Cloud* puedan notificar a los usuarios de actualizaciones en el canal.
Atributos	• `domain` – Atributo opcional que hace referencia a el Servicio Web que procesará la notificación de actualizaciones. • `port` – Puerto a través del cual se hará la conexión con el Servidor indicado en el atributo *domain*. • `path` – Ruta dentro del Servidor indicado en el atributo *domain* para llegar a la aplicación de notificación de actualizaciones. • `registerProcedure` – Procedimiento de registro que será usado para procesar el canal. • `protocol` – Protocolo de comunicación que será usado para la comunicación con el Servidor.
Sub elementos requeridos	Ninguno
Sub elementos opcionales	Ninguno

La Figura 2.3 muestra la estructura completa de los elementos que se definen en RSS 0.92.

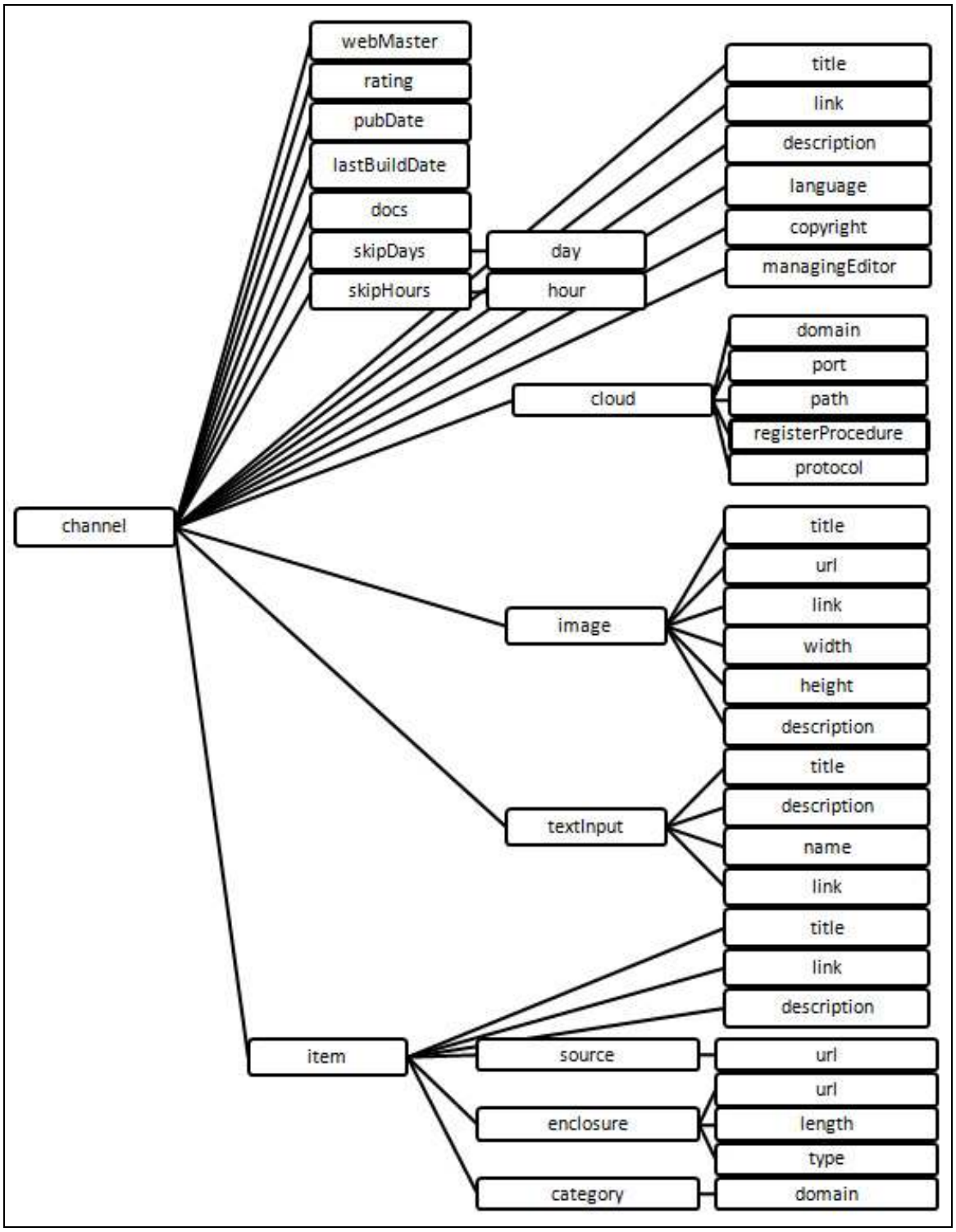

Figura 2. 3 - Elementos de RSS 0.92

Ejemplo

Siguiendo la normatividad de la especificación, la Figura 2.4 muestra un Ejemplo de la forma en la que se debe de construir un archivo RSS 0.92. Se hace uso de todos los elementos nuevos respetando la especificación de cada uno de ellos.

```
<?xml version="1.0"?>
<rss version="0.92">
     <channel>
        <title>Ejemplo de RSS0.92</title>
        <link>http://www.urldeejemplo.com.mx/index.html</link>
        <description>Ejemplo de fuente RSS 0.92</description>
        <language>es-mx</language>
        <managingEditor>editor@ urldeejemplo.com.mx </managingEditor>
        <webMaster>webmaster@ urldeejemplo.com.mx </webMaster>
        <rating> </rating>
        <pubDate>03 Apr 08 1500 GMT</pubDate>
        <lastBuildDate>03 Apr 08 1500 GMT</lastBuildDate>
        <docs>http://backend.userland.com/rss091</docs>
        <skipDays>
            <day>Friday</day>
        </skipDays>
        <skipHours>
            <hour>11</hour>
        </skipHours>
        <cloud domain="http://www.cloudejemplo.com" port="80" path="/RPC2"
        registerProcedure="notificame" protocol="XML-RPC" />
        <image>
            <title>Ejemplo de RSS 0.92</title>
            <url>http://www.urldeejemplo.com.mx/imagenes/logo.gif</url>
            <link>http://www.urldeejemplo.com.mx/index.html</link>
            <width>88</width>
            <height>31</height>
            <description>El Mundo a través de la Sindicación de Contenido</description>
        </image>
        <textInput>
            <title>Búsqeuda</title>
            <description>Busca en la Base de Datos de Archivos</description>
            <name>consulta</name>
            <link>http://www.urldeejemplo.com.mx/busqueda.cgi</link>
        </textInput>
        <item>
            <title>Primer Elemento</title>
            <link>http://www. urldeejemplo.com.mx/001.html</link>
            <description>Esta es la descripción del primer elemento.</description>
            <source url="http://www.sitioexterno.com/index.xml">Sitio Externo</source>
            <enclosure url="http://www.urldeejemplo.com.mx/001.mp3" length="543210"
                type"audio/mpeg"/>
            <category
                domain="http://www.dmoz.org">Business/Industries/Publishing/Publishers
                /Nonfiction/Business/>
        </item>
    </channel>
</rss>
```

Figura 2. 4 - Ejemplo de Archivo RSS 0.92

RSS 1.0

RSS 1.0 cambia drásticamente su funcionalidad y especificación con respecto a RSS 0.91 y RSS 0.92 debido a la incorporación de RDF.

La incorporación de RDF al estándar trajo consigo ventajas y desventajas. La principal desventaja fue que el estándar se volvió mucho más complejo, su sintaxis y construcción resultan mucho más difíciles de entender para usuarios promedio. Sin embargo, resulta ser fácil de extender y tiene un soporte más especializado y fuerte para el trabajo con metadatos.

El núcleo de RSS 1.0 fue construido sobre RSS 0.9. RSS 1.0 se enfoca en la extensibilidad a través de espacios de nombres y RDF sin dejar de ser compatible con RSS 0.9.

Especificación

A pesar de que RSS 1.0 siguió una corriente diferente a la seguida por RSS 0.91 y RSS 0.92, ambas ramas hacen uso de elementos similares. A continuación se muestra un resumen de las etiquetas de RSS 1.0 haciendo omisión de algunas que ya fueron descritas previamente en las secciones dedicadas a RSS 0.91 y RSS 0.92.

Etiqueta <rdf:RDF>

Descripción	Esta etiqueta debe de ser siempre el elemento externo de todo documento RSS 1.0. La etiqueta <rdf:RDF> de inicio asocia el prefijo rdf: del espacio de nombres con el esquema de sintaxis RDF y establece el esquema de RSS 1.0 como el espacio de nombres predeterminado para el documento. A pesar de que podría usarse cualquier otro prefijo asociado a un espacio de nombres diferente, es necesario usar el prefijo rdf: si se desea mantener la compatibilidad con RSS 0.9.
Atributos	Ninguno
Sub elementos requeridos	• channel – Cada archivo debe de contener, cuando menos, un canal. • item – Cada archivo debe de contener, cuando menos, un elemento <item>. En RSS 1.0, las etiquetas <item> son sub elementos tanto de la etiqueta <channel> como de la etiqueta <rdf:RDF>.
Sub elementos opcionales	• image – Imagen asociada al Contenido Sindicado. • textinput – Cuadro de texto editable para obtener información del usuario.

Etiqueta <items>

Descripción	Esta etiqueta representa la Tabla de Contenido del archivo RSS 1.0. La principal función de esta etiqueta es la de asociar cada uno de los elementos <item> dentro del documento, con un canal RSS en particular. Los elementos <item> que aparezcan en el documento pero que no sean miembros de las etiquetas <items> del canal pueden ser descartados al leer el archivo RSS.
Atributos	Ninguno
Sub elementos requeridos	• rdf:Seq – Etiqueta que define una secuencia. Es usada para ordenar una serie de elementos. En este caso, almacenaría las asociaciones especificadas dentro de la etiqueta <items>.
Sub elementos opcionales	Ninguno

La Figura 2.5 muestra un diagrama con la jerarquía de elementos que define RSS 1.0.

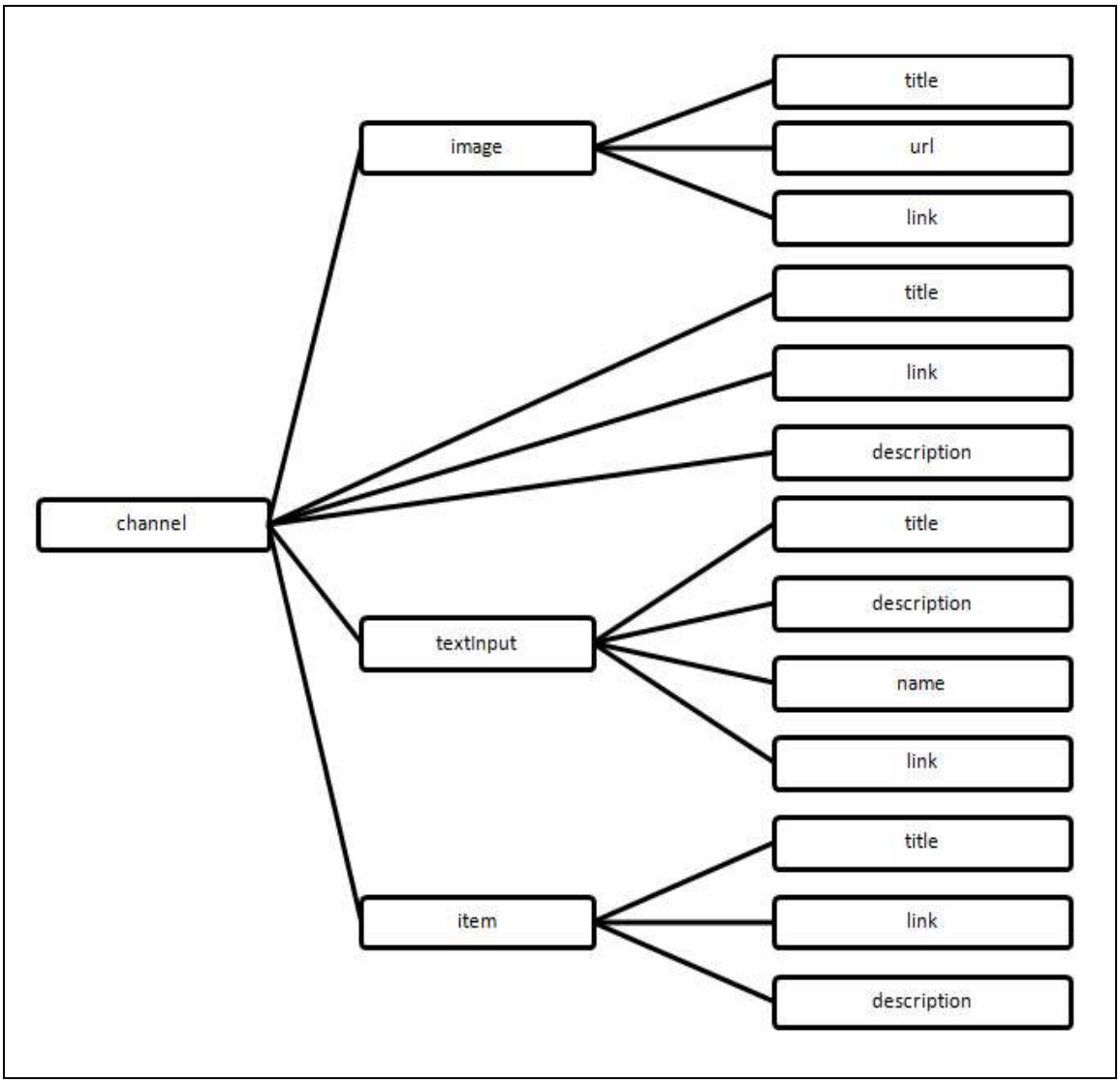

Figura 2. 5 - Elementos de RSS 1.0

Ejemplo

La Figura 2.6 muestra como ejemplo un Archivo RSS 1.0 que hace uso de los atributos definidos en la especificación discutida anteriormente.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
     xmlns:co="http://purl.org/rss/1.0/modules/company/"
     xmlns:ti="http://purl.org/rss/1.0/modules/textinput/"
     xmlns="http://purl.org/rss/1.0/">
     <channel rdf:about="http://www.urldeejemplo.com.mx/rss1.0">
        <title>Ejemplo de RSS 1.0</title>
        <link> www.urldeejemplo.com.mx/ejemplo</link>
        <description>Ejemplo de fuente RSS 1.0</description>
        <dc:publisher>Editor de la fuente</dc:publisher>
        <dc:creator>Creador de la fuente</dc:creator>
        <dc:date>2008-01-01T12:00+00:00</dc:date>
        <sy:updatePeriod>hourly</sy:updatePeriod>
        <sy:updateFrequency>2</sy:updateFrequency>
        <sy:updateBase>2008-01-01T12:00+00:00</sy:updateBase>
        <image rdf:resource="http://www.urldeejemplo.com.mx/imagen.gif" />
        <textinput rdf:resource="http://www.urldeejemplo.com.mx" />
        <items>
          <rdf:Seq>
             <rdf:li resource="http://www.acercadelrecurso.com" />
          </rdf:Seq>
        </items>
     </channel>
     <image rdf:about="http://www.urldeejemplo.com.mx/icono.gif">
        <title>Imagen Logotipo</title>
        <url>http://www.urldeejemplo.com.mx/icono.gif</url>
        <link>http://www.urldeejemplo.com.mx</link>
     </image>
     <textinput rdf:about="http://www.urldeejemplo.com.mx">
        <title>Búsqueda</title>
        <description>Busca en la Base de Datos</description>
        <name>query</name>
        <link>http://www.urldeejemplo.com.mx</link>
        <ti:function>search</ti:function>
        <ti:inputType>regex</ti:inputType>
     </textinput>
     <item rdf:about="http://www.ligaalanoticia.com.mx">
       <title>Titulo de la noticia</title>
       <link>http://www.ligaalanoticia.com.mx</link>
       <dc:description>Descripción de la Noticia</dc:description>
       <dc:publisher>Editor de la Noticia</dc:publisher>
       <dc:creator>Creador de la Noticia</dc:creator>
       <dc:subject>Tema de la Noticia</dc:subject>
       <co:name>Nombre de la empresa asociada con la Noticia</co:name>
       <co:market>NASDAQ</co:market>
       <co:symbol>Nombre abreviado de la empresa</co:symbol>
     </item>
</rdf:RDF>
```

Figura 2. 6 - Ejemplo de Archivo RSS 1.0

RSS 2.0

Fue a mediados del 2002 que Dave Winer y Userland Software©, tras anunciar la versión 2.0 de RSS, declararon que la especificación de RSS se encontraría "congelada", queriendo decir con esto que no se harían más actualizaciones a la especificación de RSS.

Se han publicado algunas otras versiones de RSS (2.0.1, 2.0.1-rv1, 2.0.1-rv2, 2.0.1-rv3, 2.0.1-rv4, 2.0.1-rv5, 2.0.1-rv6, 2.0.8, 2.0.9 y 2.0.10) pero ninguna de ellas ha hecho cambios significativos a la especificación original de 2.0.

Actualizaciones

Entre las actualizaciones más importantes que trajo RSS 2.0 se encuentran:

- RSS en todas sus versiones, establece restricciones acerca de la forma en la que debe de iniciar la cadena de texto contenida por los elementos `<link>` y `<url>`. Antes de RSS 2.0, la especificación solamente permitía que estas cadenas iniciaran con *http://* o con *ftp://*. RSS 2.0 permite que éstos elementos ahora inicien con cualquiera de los siguientes prefijos: *http://*, *https://*, *news://*, *mailto:* y *ftp://*.
- RSS 2.0 mejora el sistema de clasificación de canales introduciendo niveles dentro de las categorías. Con esto, el sistema de categorías para canales toma una estructura más organizada y estandarizada en forma de árbol que los editores de Sindicación de Contenido pueden implementar fácilmente.
- Tal vez la mejora más significativa de RSS 2.0 consiste en su capacidad de extensión. Para lograr que los usuarios y programadores pudieran extender RSS 2.0 con nuevos elementos y definiciones, el estándar define que una fuente RSS puede contener elementos que no estén en la especificación oficial de RSS 2.0 siempre y cuando estos elementos se encuentren definidos en un espacio de nombres RSS.

Especificación

Entre las etiquetas que sufrieron modificaciones con la versión 2.0 de RSS se encuentran:

Etiqueta `<channel>`

Descripción	La especificación de esta etiqueta sufre cambios constantemente debido a que es la etiqueta principal y externa dentro de la estructura XML de los archivos RSS. Como ha sido desde un principio, la etiqueta `<channel>` define un canal. Pueden existir uno o más canales dentro de un solo archivo RSS.
Atributos	Ninguno
Sub elementos requeridos	• `title` – El nombre del canal. Es la forma en la que los usuarios harán referencia al Servicio Web. Si se tiene un sitio HTML que contenga la misma información que el archivo RSS, el título del canal deberá de ser el mismo que el título del sitio web. • `link` – La URL del sitio web asociado al canal. • `description` – Frase o enunciado describiendo el canal.
Sub elementos opcionales	• `language` – El idioma en el que se encuentra escrito el canal. Esto permite a los agregadores de Contenido Sindicado agrupar, por ejemplo, todas las fuentes en Español de México en un solo lugar. El valor de esta etiqueta debe de ser uno de los valores listados en la especificación de RSS 2.0. • `copyright` – Derechos de autor asociados con el canal. • `managingEditor` – Dirección de correo electrónico de la persona responsable del contenido editorial del documento. • `webMaster` – Dirección de correo electrónico de la persona responsable por problemas técnicos relacionados con el canal. • `pubDate` – La fecha de publicación del contenido del documento. • `lastBuildDate` – La fecha de la última actualización del documento. • `category` – Especifica una o más categorías a las que pertenece el canal. • `generator` – Especifica el programa usado para generar el canal. • `docs` – URL que apunta a la documentación del estándar usado para construir el documento. • `cloud` – Permite la subscripción a Servicios Web que se encarguen de manejar *clouds* o servicios de actualizaciones. • `ttl` – Número de minutos que el documento permanecerá válido. Después de este tiempo, el canal debe de ser actualizado de su fuente original. • `image` – Especifica una imagen GIF, JPEG o PNG que puede ser

	desplegada con el canal. • `rating` – La audiencia medida en PICS del canal. • `textInput` – Define un cuadro de texto mediante el cual el editor puede recibir información del usuario como en un cuadro de texto HTML tradicional. • `skipHours` – Define las horas del día en las que este documento no será actualizado. • `skipDays` – Define los días de la semana en los que este documento no será actualizado.

Etiqueta `<item>`

Descripción	Un canal puede contener un número indefinido de elementos `<item>`. Un `<item>` representa una "historia", una entrada como una noticia de una revista o periódico. Cuando un `<item>` hace referencia a una noticia, su descripción es entonces una sinopsis de la noticia y su elemento `<link>` apunta a la versión completa de la historia.
Atributos	Ninguno
Sub elementos requeridos	Ninguno
Sub elementos opcionales	• `title` – El título del elemento. • `link` – La URL con la información completa de la noticia señalada por el evento. • `description` – La sinopsis del evento señalado por el elemento. • `author` – Dirección de correo electrónico del autor de la entrada. • `category` – Especifica una o más categorías para el elemento. • `comments` – URL de algún sitio web con comentarios acerca del contenido del elemento. • `enclosure` – Describe contenido multimedia asociado al elemento. • `guid` – Cadena que identifica de manera única al elemento. • `pubDate` – Indica la fecha de publicación del elemento. • source – El canal RSS del que proviene el elemento.

La Figura 2.7 muestra la estructura de elementos del estándar RSS 2.0.

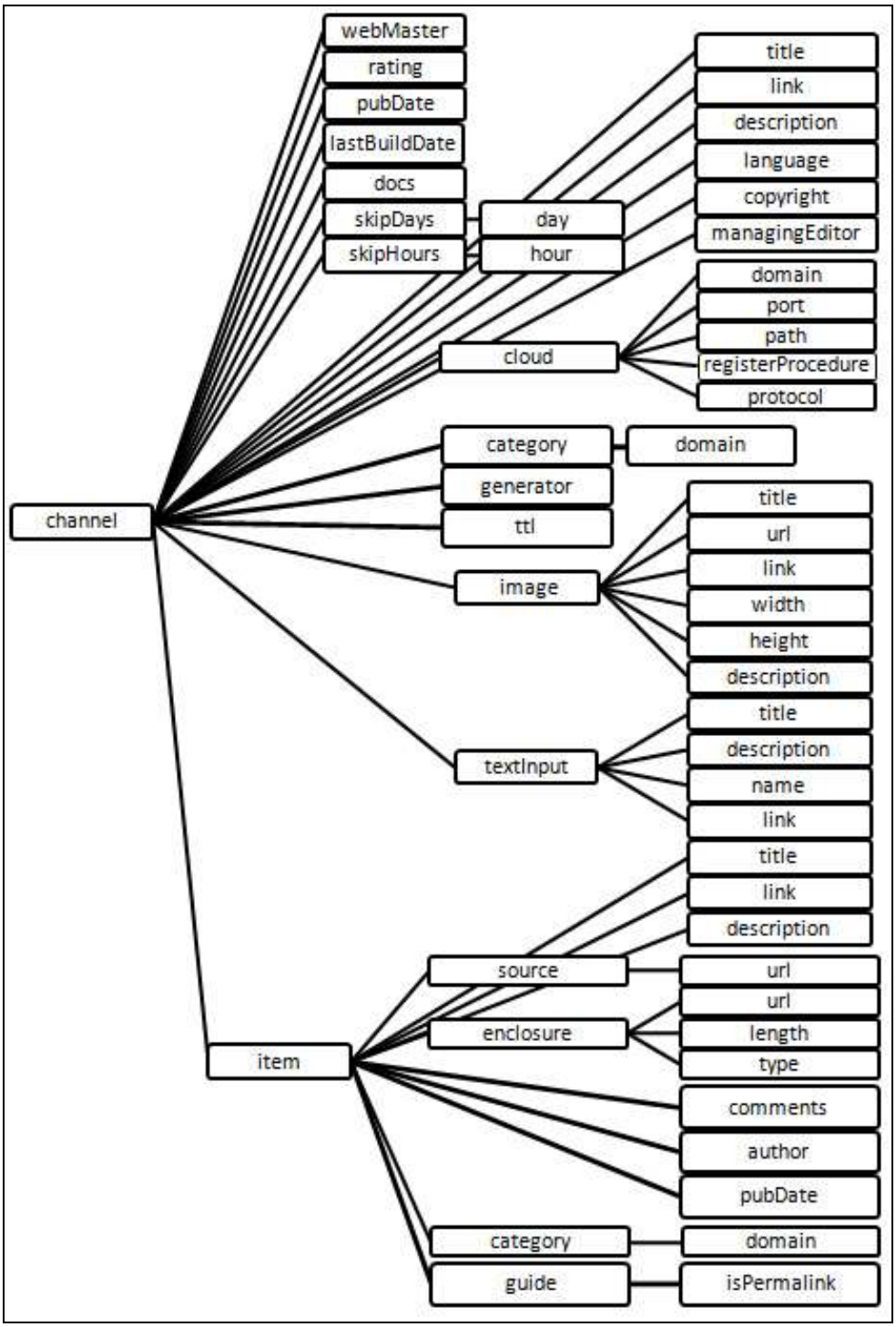

Figura 2. 7 - Elementos de RSS 2.0

Ejemplo

La Figura 2.8 muestra el contenido de un Archivo RSS 2.0 que tiene como objetivo ejemplificar el uso de dicho estándar.

```
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
   <channel>
        <title>Ejemplo de Archivo de RSS 2.0</title>
        <link>http://www.ligadeejemplo.com.mx</link>
        <description>Ejemplo de una fuente RSS 2.0</description>
        <language>es - mx</language>
        <managingEditor>editor@ligadeejemplo.com.mx</managingEditor>
        <webMaster>webmaster@ligadeejemplo.com.mx</webMaster>
        <pubDate>03 Apr 08 1500 GMT</pubDate>
        <lastBuildDate>03 Apr 08 1500 GMT</lastBuildDate>
        <docs>http://backend.userland.com/rss</docs>
        <skipDays>
          <day>Monday</day>
        </skipDays>
        <skipHours>
          <hour>20</hour>
        </skipHours>
        <category
            domain="http://www.dmoz.org">Business/Industries/Publishing/Publishers/
        Nonfiction />
        <generator>NewsAggregator'o'Matic</generator>
        <ttl>30<ttl>
        <cloud domain="http://www.exampleurl.com" port="80" path="/RPC2"
            registerProcedure= "pleaseNotify" protocol="XML-RPC" />
        <image>
          <title>Ejemplo de RSS 2.0</title>
          <url>http://www.ligadeejemplo.com.mx/logo.gif</url>
          <link>http://www.ligadeejemplo.com.mx/index.html</link>
          <width>88</width>
          <height>31</height>
          <description>Descripcion de la Imagen</description>
        </image>
        <textInput>
          <title>Búsqueda</title>
          <description>Busca en la Base de Datos</description>
          <name>query</name>
          <link>http://www.ligadeejemplo.com.mx/busqueda.cgi</link>
        </textInput>
        <item>
          <title>Primer Elemento</title>
          <link>http://www.ligadeejemplo.com.mx/001.html</link>
          <description>Descripcion del primer elemento.</description>
          <dc:creator>Autor</dc:creator>
          <source url="http://www.otrositio.com/index.xml">Otro Sitio</source>
          <category
              domain="http://www.dmoz.org">Business/Industries/Publishing/Publishers/
          Nonfiction" />
          <author>Autor</author>
          <pubDate>Sat, 01 Jan 2008 0:00:01 GMT</pubDate>
          <guid isPermaLink="true">http://www.ligadeejemplo.com/001.html</guid>
        </item>
   </channel>
</rss>
```

Figura 2. 8 - Ejemplo de Archivo RSS 2.0

CAPITULO 3

Especificación del Sistema

En los capítulos anteriores, se dio una base teórica general acerca de la Sindicación de Contenido. En este capítulo y en los subsecuentes, se ahondará en las aplicaciones que esta teoría puede tener.

La estandarización de la Sindicación de Contenido permite a los editores contar con una herramienta sencilla y confiable para publicar su información pero también provee a los programadores y desarrolladores de Software de un entorno sólido para poder construir aplicaciones que faciliten y promuevan el uso global de esta tecnología.

El objetivo de los Ingenieros de Software en cuanto a Sindicación de Contenido es el de proveer herramientas para editores y usuarios finales que permitan la creación, manipulación, búsqueda, consulta y almacenamiento de Contenido Sindicado.

Dado que actualmente en la Web se tienen miles y miles de fuentes de Contenido Sindicado, una de las tareas indispensables que el Software tiene que facilitar es la de poder consultar y agrupar varias fuentes en un solo lugar. El usuario no desea tener que navegar de un sitio web a otro para poder consultar todas las fuentes de Contenido Sindicado que le interesan, por el contrario, el usuario busca tener toda

la información de su agrado disponible en un solo sitio y en un mismo instante. Es entonces que programas que se encarguen de recopilar Contenido Sindicado de múltiples fuentes son una parte esencial de las soluciones de Software para Contenido Sindicado.

No pasó mucho tiempo antes de que este tipo de aplicaciones obtuvieran su propio nombre y terminología. En inglés llamados *Web Aggregators* y en español conocidos bajo una mala traducción: *Agregadores Web*, estos programas se usan tanto como Aplicaciones de Escritorio como Servicios Web y hoy en día son una herramienta indispensable para cualquier usuario del Contenido Sindicado.

A lo largo de este capítulo y en los capítulos subsecuentes, se analiza y documenta el proceso de Ingeniería de Software que se trabajó para el desarrollo de un Agregador Web. El proceso comprende cuatro etapas:

1. Especificación
2. Análisis
3. Diseño
4. Implementación

Cada una de ellas será desglosada de manera individual en un capítulo independiente. Este capítulo da inicio a la documentación de este proceso de Ingeniería de Software desarrollando la etapa de Especificación del mismo.

Introducción

El proceso de Especificación pretende definir, en lenguaje humano, el comportamiento y funcionalidad que se espera tenga el programa.

La Especificación aquí presentada se encuentra dividida a su vez en dos secciones: la Especificación Funcional y la Especificación No Funcional.

Dentro de la Especificación Funcional se hablará de los casos y escenarios de uso del programa y se definirá detalladamente cual se espera sea el resultado final de cada uno de éstos.

Dentro de la Especificación No Funcional se incluyen detalles acerca de la usabilidad, desempeño, escalabilidad, requisitos de Hardware y requisitos de Software del programa.

Por último, se incluye un Diccionario de Términos. En este diccionario se incluye la definición de todas las palabras que constituyen el dominio del programa.

Especificación Funcional

El programa a desarrollar tiene el nombre de: *Agregador Web BiDi*. El hecho de que el nombre haga uso del término *Agregador Web* se ha esclarecido en párrafos anteriores, por lo que solamente resta especificar que la inclusión de la palabra *BiDi* se debe a que la aplicación fue desarrollada expresamente para ser usada dentro de la Biblioteca Digital (BiDi) de la Dirección General de Bibliotecas de la UNAM.

Descripción General

Agregador Web BiDi es un programa que debe recopilar información de fuentes de Contenido Sindicado en la Web devolviéndola de forma ordenada y estandarizada.

Agregador Web BiDi deberá de poder recopilar información de fuentes que sigan cualquiera de los estándares mencionados anteriormente: RSS 0.91, RSS 0.92, RSS 1.0 o RSS 2.0.

Agregador Web BiDi es una biblioteca, no es un Servicio Web ni una Aplicación de Escritorio. *Agregador Web BiDi* permitirá que Servicios Web y aplicaciones ejecutables puedan tener la funcionalidad de trabajar con fuentes de Contenido Sindicado.

Escenarios de Uso

Los Escenarios de Uso de una aplicación son las diferentes maneras en las que esta aplicación puede ser usada y que se definen normalmente como secuencias de acciones con estados inicial y final. *Agregador Web BiDi* es un componente con una funcionalidad concreta y muy bien definida que se sintetiza en un único Escenario de Uso.

✓ Escenario de Uso: Recopilación de Contenido Sindicado

La Figura 3.1 muestra el diagrama del flujo general de este Escenario.

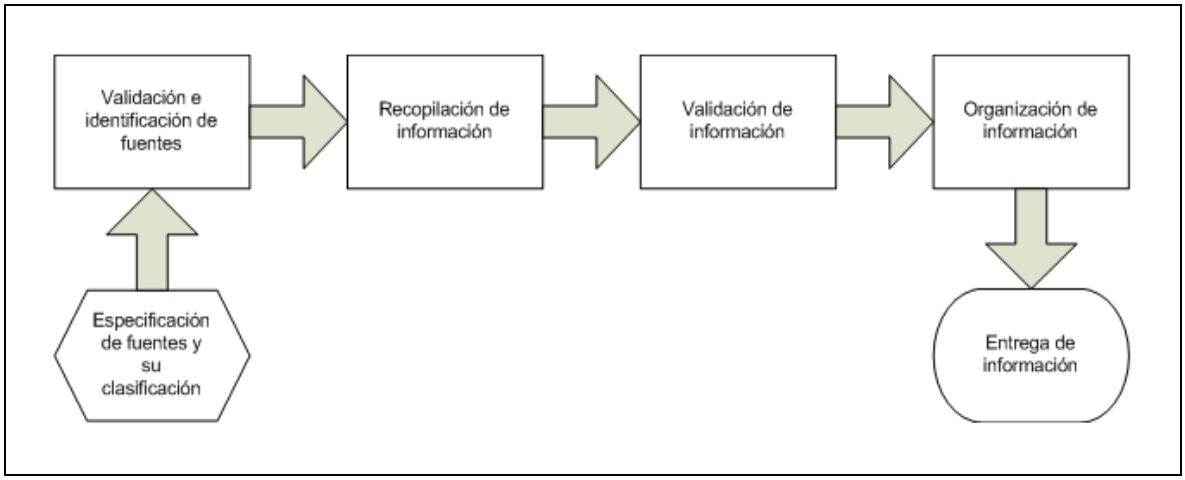

Figura 3. 1 - Escenario de Uso

1. Especificación de las fuentes de las que se recopilará información.
   El usuario especifica cuáles son las fuentes de las que se recopilará información mediante la descripción de los datos requeridos de cada una de ellas.
2. Especificación de la clasificación de las fuentes.
   El usuario especifica los Grupos en los que desea que las fuentes sean clasificadas. La clasificación será muy sencilla: cada fuente puede pertenecer solamente a un grupo, cada grupo puede contener cero o más fuentes y los grupos no pueden contener sub grupos internos.

3. Validación e identificación de las fuentes.
   Agregador Web BiDi procesará la información provista por el usuario anteriormente y verificará que ésta sea válida y completa.
4. Recopilación de información de cada una de las fuentes.
   Se hará la comunicación con cada una de las fuentes solicitando la información requerida.
5. Validación de la información recopilada.
   Una vez recopilada la información, se verificará que ésta sea válida y completa.
6. Organización de la información recopilada.
   Ya que la información ha sido recopilada y validada, se organizará alfabéticamente respetando la clasificación de grupos especificada por el usuario al inicio del proceso.
7. Salida de la información recopilada.
   Ya terminado el proceso, se entrega como salida toda la información recopilada previamente organizada y validada.
   La salida de la aplicación deberá de ser un objeto que contenga toda esta información organizada.

Casos de Uso

Como Casos de Uso se señalan eventos y pequeñas sub secuencias de acciones que suceden o pueden suceder durante el Escenario de Uso de la aplicación. Es importante aclarar que en el proceso de Especificación no se usa ningún lenguaje computacional, por lo que si se quiere conocer información acerca de detalles más profundos de algún Caso de Uso en particular, es necesario consultar la sección de Análisis.

- ✓ Caso de Uso: Especificación de las fuentes

El usuario provee la información de todas las fuentes de las que desea obtener información. El formato mediante el cual el usuario especificará la información, los datos que serán necesarios para describir cada una de las fuentes y la forma en la

que la aplicación hará uso de este contenido será desarrollado en la sección de Análisis. La Especificación de las Fuentes abarca tres Sub Casos de Uso: Saber de qué Fuentes se desea obtener información, Obtener los datos necesarios de cada una de las Fuentes y Recopilar los datos de todas las Fuentes en una única Especificación.

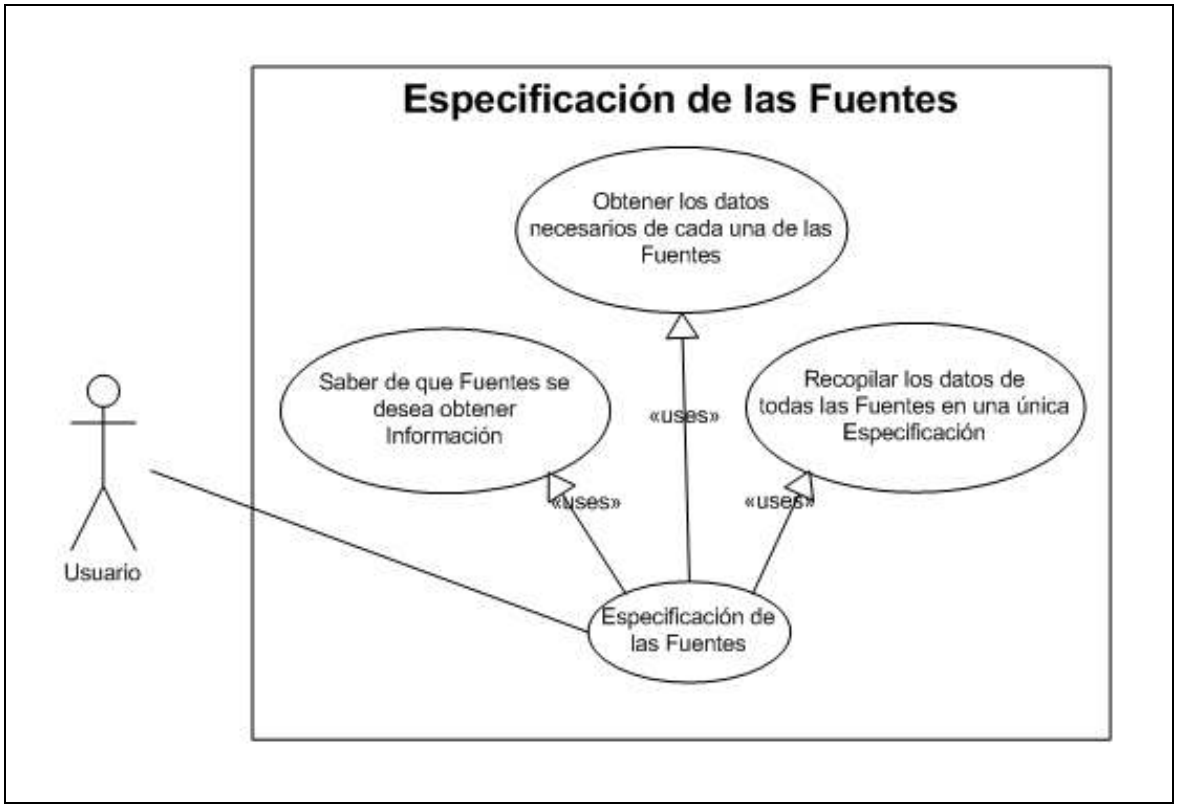

Figura 3. 2 - Caso de Uso: Especificación de las Fuentes

✓ Caso de Uso: Saber de qué Fuentes se desea obtener información

El usuario determina qué fuentes le interesan. Este Caso de Uso no incluye la construcción de ningún documento o compendio, solo define el proceso de conocer el conjunto de fuentes, sin sus datos particulares, que desean ser consultadas.

✓ Caso de Uso: Obtener los datos necesarios de cada una de las Fuentes

El usuario deberá de conocer cuáles son los datos que debe de recopilar de cada una de las fuentes. El conjunto de estos datos es aclarado en la Sección de Análisis.

- ✓ Caso de Uso: Recopilar los datos de todas las Fuentes en una única Especificación

El usuario deberá de conocer cuál es el formato mediante el cual Agregador Web BiDi recibirá la información acerca de las fuentes. Una vez conocido este formato, construye un ejemplar de este formato con los datos que haya recopilado de las fuentes que serán consultadas.

- ✓ Caso de Uso: Error al leer la Especificación de las fuentes

A pesar de que está fuera de los alcances de la Especificación el determinar el manejo de errores de la aplicación, es necesario decir que si se produce cualquier tipo de error al leer las fuentes, el proceso deberá de ser cancelado.

En caso de que el error se refiera solamente a la lectura de la información de una fuente en específico, se deberá de notificar al usuario del error y el proceso deberá de continuar normalmente haciendo omisión de la fuente que presentó el error.

- ✓ Caso de Uso: Detección de fuente inválida

Si dentro de la especificación de las fuentes se encuentra que los datos introducidos son semánticamente incorrectos, se deberá de notificar al usuario del error y la fuente será omitida en la recopilación.

- ✓ Caso de Uso: La información obtenida es inválida

Si al obtener información de una fuente se encuentra que ésta es inválida, la información será descartada por motivos de seguridad. Una vez acabado el proceso, se deberá de notificar al usuario que la información fue descartada y la razón de dicha acción.

- ✓ Caso de Uso: Organización de la información

La información será organizada por grupos de fuentes y por orden alfabético. El proceso de organización deberá de ser optimizado lo más posible. El resultado que se regrese al usuario al finalizar el proceso deberá de estar organizado por estos parámetros. En la documentación de la aplicación, se deberá de especificar la forma en la que el usuario puede sacar provecho de dicha organización.

✓ Caso de Uso: Notificación de Errores

Tanto el manejo de errores como el proceso de notificación de los mismos son temas fuera del alcance de la Especificación. Sin embargo, es necesario mencionar que el usuario deberá de ser notificado de cualquier anomalía que se haya presentado durante la ejecución del proceso de recopilación, no deberán por ningún motivo de existir casos en los que ocurran errores y el usuario no reciba una descripción clara de por qué el proceso de recopilación se detuvo.

Especificación No Funcional

Los requerimientos básicos de software y hardware que deberá de respetar la aplicación se condensan en la siguiente tabla.

Requerimiento	Especificación
Lenguaje de Programación en el que será desarrollada	Java 1.6.0
Sistemas Operativos en los que podrá ser usada	Cualquier Sistema Operativo en el que se pueda instalar una Máquina Virtual de Java
Arquitecturas en las que podrá ser usada	Cualquier Arquitectura que soporte alguno de los Sistemas Operativos requeridos además de la instalación y ejecución de la Máquina Virtual de Java

Figura 3. 2 - Requerimientos básicos de Hardware y Software

Diccionario de Términos

Agregador Web. Programa que se encarga de recopilar información de diferentes Fuentes de Contenido Sindicado. También usado como Agregador.

Aplicación de Escritorio. Programa que es ejecutado en la máquina del usuario y que no es ejecutado por otro programa o dentro de otro programa. También conocidas como Aplicaciones *Stand-Alone*.

Aplicación Web. Programa con una arquitectura cliente – servidor que se caracteriza por manejar comunicaciones a través de una red.

Contenido Sindicado. Información editada por un sitio web a la que se le da un formato específico y que suele ser breve y de temas principalmente noticiosos.

Estándar de Contenido Sindicado. Estándar desarrollado para unificar la edición de Contenido Sindicado. También usado como Estándar RSS o Estándar de Sindicación.

Fuente de Contenido Sindicado. Archivo o programa que provee un Contenido Sindicado. También usado como Fuente.

Grupo de Fuentes. Clasificación mediante la cual se pueden agrupar varias Fuentes de Contenido Sindicado. Los Grupos de Fuentes serán definidos por el usuario.

Servicio Web. Aplicación Web que provee de un servicio mediante la Web.

Sindicación de Contenido. Acción de convertir cualquier información en Contenido Sindicado.

Sitio web. Conjunto de archivos y programas que normalmente son públicos y que pueden ser consultados mediante Internet.

Usuario. Define al usuario de la aplicación *Agregador Web BiDi*. El usuario de la aplicación puede ser una persona interesada en usar la funcionalidad provista por la aplicación o un programa que esté haciendo uso de *Agregador Web BiDi* como una librería para alcanzar un resultado más complejo.

CAPITULO 4

Análisis del Sistema

Durante el proceso de Análisis, se pretende dar un toque computacional a la Especificación, transformando la misma en un lenguaje computacional. Para lograr este objetivo, el proceso de Análisis se puede documentar agregando cuantioso contenido *UML* que permita especificar con mayor detalle y en términos computacionales lo que la Especificación indica.

El primer gran paso que da el Análisis, es la inclusión de un diagrama general de clases para la aplicación. Este primer diagrama sólo incluye una visión superficial de la estructura de clases que incluye:

- Especificación de las clases
- Distribución de estas clases en paquetes
- Especificación de las relaciones entre las clases

Posteriormente, el Análisis comprende la construcción de Diagramas de Colaboración UML y Diagramas de Secuencia UML para cada uno de los Casos de Uso que se definieron en la Especificación de la aplicación los cuales permitirán entrar al proceso de Diseño con un panorama más limpio y ordenado.

Diagrama de Clases

Sería inútil desplegar solamente un diagrama de clases que comprendiera toda la interacción, clasificación y funcionalidad de la estructura de clases de la aplicación debido a que el diagrama general es demasiado complejo. Por eso se hace una construcción gradual empezando con la estructura de paquetes desde el nivel superior y llegando hasta la especificación de los atributos e interacción de cada una de las clases.

Estructura de Paquetes

La estructura de clases se divide en dos paquetes principales: el paquete **process** y el paquete **rss**.

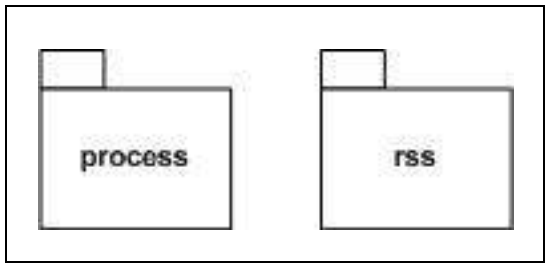

Figura 4. 1 – Paquetes del nivel superior

El paquete **rss**, contiene la definición de los conceptos de Sindicación de Contenido (como canales, ítems, recursos, etc.) además de la clase principal que se encarga de agregar todo el contenido recopilado en un solo objeto.

El paquete **rss** se compone a su vez por cuatro paquetes:

- Paquete **aggregator**. Contiene las clases de control. Se encarga de hacer uso de todas las demás clases del paquete rss para lograr el funcionamiento buscado.
- Paquete **elements**. Contiene la definición de los elementos de un canal como se detallan en las especificaciones de RSS.

- Paquete **resources**. Contiene la definición de los elementos de más alto nivel de la abstracción de la Sindicación de Contenido como canales e ítems.
- Paquete **util**. Contiene recursos auxiliares para la ejecución central de la aplicación como los parámetros de configuración introducidos por el usuario.

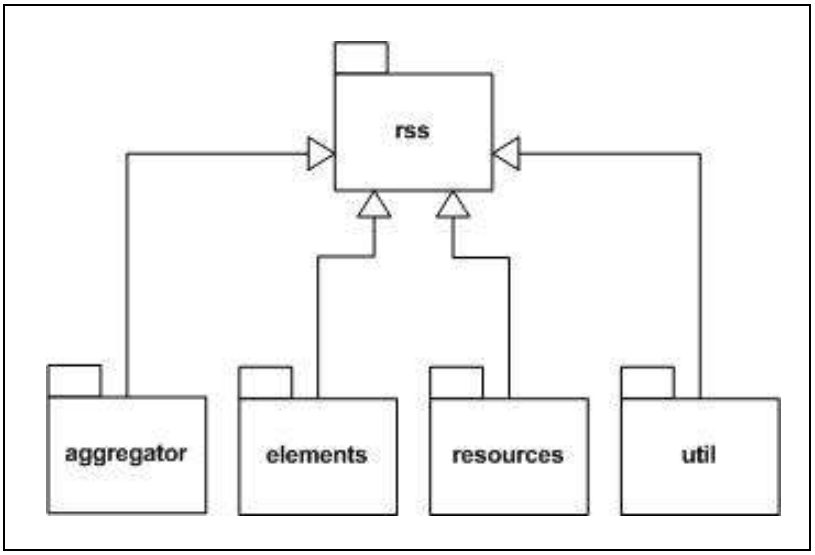

Figura 4. 2 - Estructura interna del Paquete rss

El paquete **process**, contiene la implementación de las acciones de comparación, análisis, búsqueda y ordenamiento que se ejecutarán sobre los elementos definidos en el paquete rss.

El paquete **process** está compuesto por cinco paquetes internos:

- Paquete **comparing**. Contiene la funcionalidad para realizar comparaciones entre los elementos de la aplicación.
- Paquete **parsing**. Contiene la funcionalidad para analizar fuentes y archivos y poder construir objetos del sistema.
- Paquete **searching**. Contiene la funcionalidad para realizar búsquedas sobre las fuentes y poder obtener la información necesaria para construir canales.
- Paquete **sorting**. Contiene la funcionalidad necesaria para ordenar los elementos de la aplicación.
- Paquete **util**. Contiene herramientas auxiliares para la ejecución de los procesos definidos en los otros cuatro paquetes.

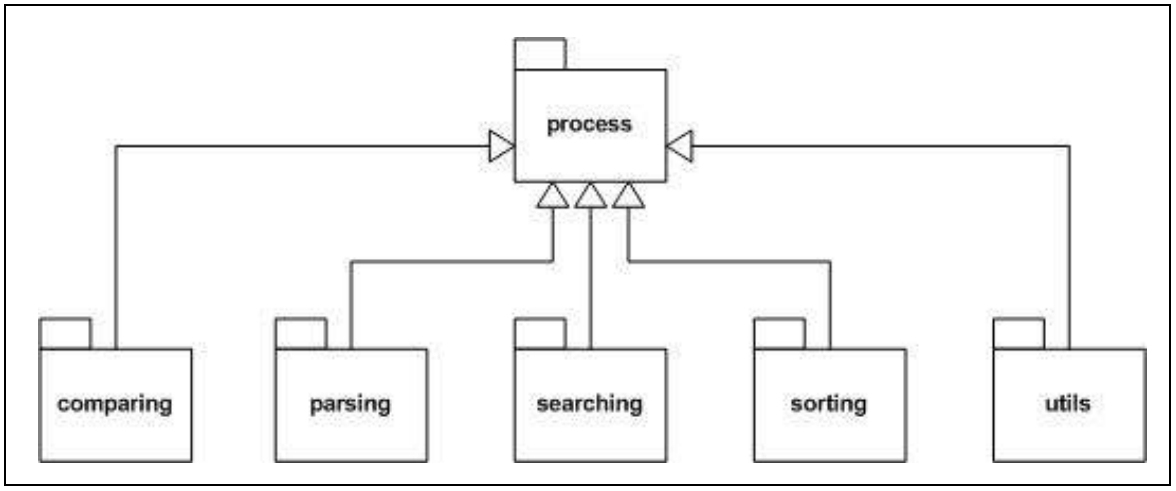

Figura 4. 3 - Estructura interna del Paquete process

Finalmente, la estructura de paquetes queda conformada por dos niveles, donde el Bajo Nivel se conforma por nueve clases agrupadas en dos paquetes.

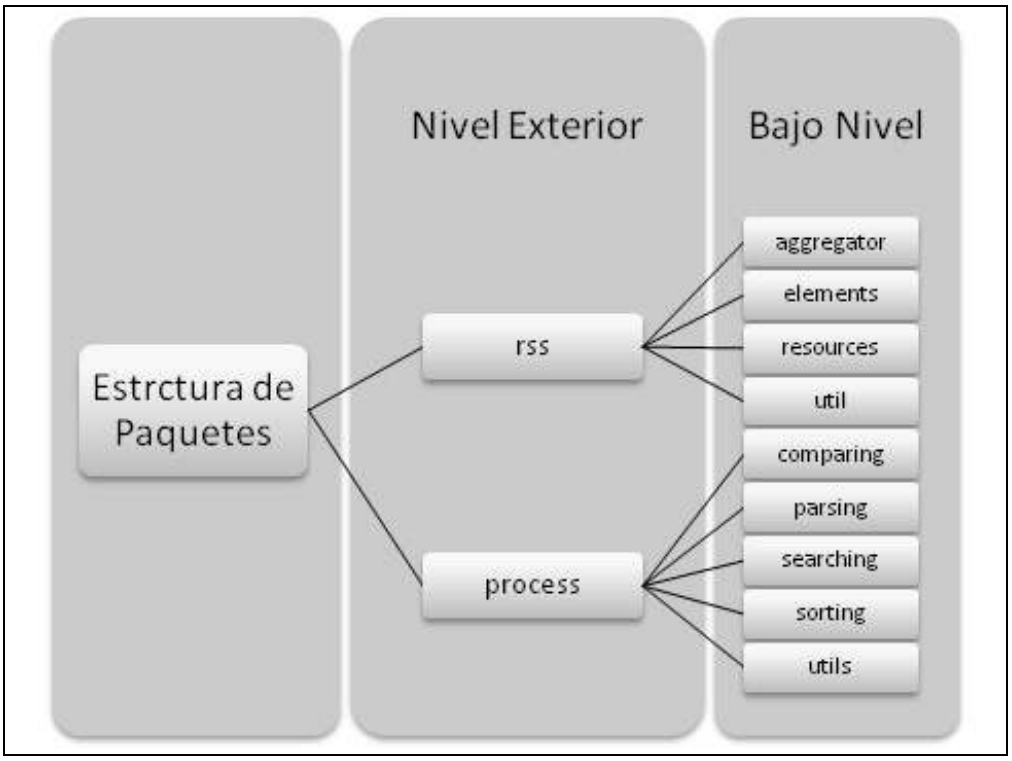

Figura 4. 4 - Estructura completa de Paquetes

Estructura de Clases

Se empezará detallando la estructura de clases por paquetes visitando cada uno de los paquetes de bajo nivel. Al final de esta sección, se incluirá un resumen con la estructura general de clases de toda la aplicación.

Paquete rss.aggregator

Una sola clase es la que conforma este paquete: **rss.aggregator.Aggregator**. La clase Aggregator es el control de la aplicación. Desde esta clase se invocan todas las acciones que se ven involucradas en el proceso de recolección.

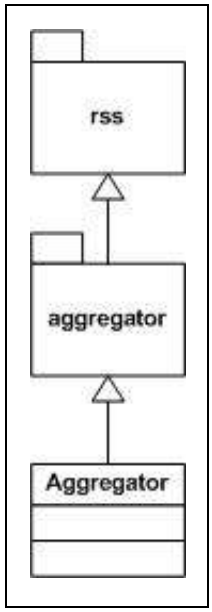

Figura 4. 5 - Paquete rss.aggregator

Paquete rss.elements

Este paquete almacena la descripción de los elementos sobresalientes de un canal según la especificación del estándar RSS descrito en los primeros capítulos de este documento. El paquete posee una clase independiente por cada uno de los elementos representados.

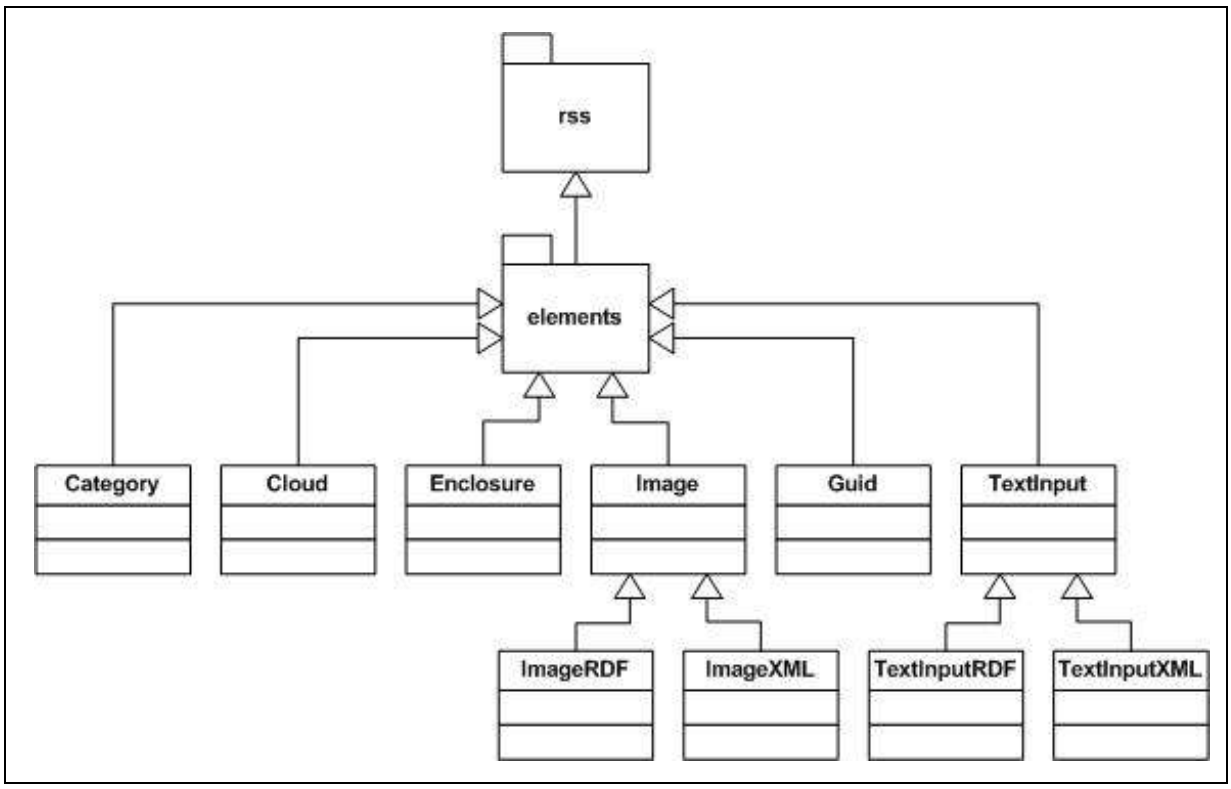

Figura 4. 6 - Paquete rss.elements

- Clase **rss.elements.Category**. Define el elemento *category* especificado para RSS 1.0 y RSS 2.0.
- Clase **rss.elements.Cloud**. Define el elemento *cloud* especificado para RSS 1.0 y RSS 2.0.
- Clase **rss.elements.Enclosure**. Define el elemento *enclosure* especificado para RSS 1.0 y RSS 2.0.
- Clase **rss.elements.Image**. Define una clase padre para especificar elementos *image* para RSS 1.0 y RSS 2.0.
- Clase **rss.elements.ImageXML**. Define el elemento *image* especificado en RSS 2.0.
- Clase **rss.elements.ImageRDF**. Define el elemento *image* especificado en RSS 1.0.
- Clase **rss.elements.Guid**. Define el elemento *guid* especificado en RSS 2.0 y RSS 1.0.
- Clase **rss.elements.TextInput**. Define una clase padre para especificar elementos *text input* para RSS 1.0 y RSS 2.0.

- Clase **rss.elements.TextInputXML**. Define el elemento *text input* especificado en RSS 2.0.
- Clase **rss.elements.TextInputRDF**. Define el elemento *text input* especificado en RSS 1.0.

Paquete rss.resources

Las seis clases que conforman este paquete permiten definir los canales y sus ítems según la especificación de los estándares 1.0 y 2.0 de RSS.

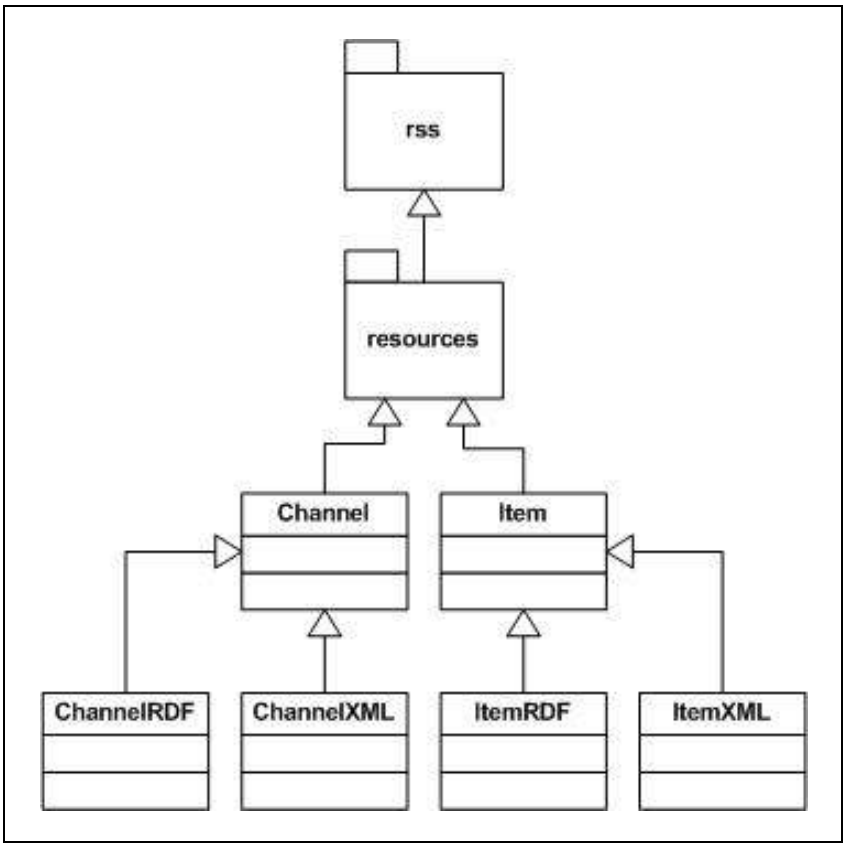

Figura 4. 7 - Paquete rss.resources

- Clase **rss.resources.Channel**. Define un canal genérico para ser extendido por las definiciones particulares de cada estándar.
- Clase **rss.resources.ChannelRDF**. Define un canal según el estándar RSS 1.0.
- Clase **rss.resources.ChannelXML**. Define un canal según el estándar RSS 2.0.
- Clase **rss.resources.Item**. Define un ítem genérico para ser extendido por las definiciones particulares de cada estándar.

- Clase **rss.resources.ItemRDF**. Define un ítem según el estándar RSS 1.0.
- Clase **rss.resources.ItemXML**. Define un ítem según el estándar RSS 2.0.

Paquete rss.util

Las clases del paquete **rss.util** agrupan las funcionalidades auxiliares que son requeridas por los demás miembros del paquete rss. Este paquete se conforma por tres clases.

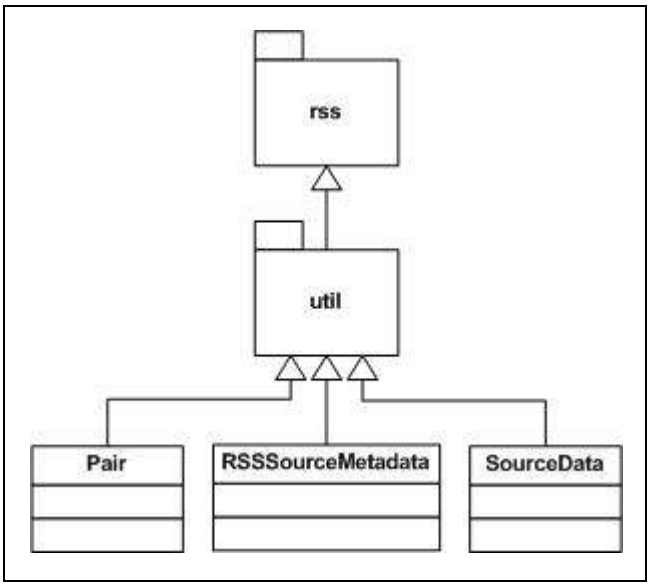

Figura 4. 8 - Paquete rss.util

- Clase **rss.util.Pair**. Define una estructura formada por un recurso y un valor. Esta estructura permite agregar información adicional a ítems y canales.
- Clase **rss.util.RSSSourceMetadata**. Almacena los datos de una fuente de acuerdo a los metadatos que hayan sido introducidos por el usuario.
- Clase **rss.util.SourceData**. Se encarga de leer, almacenar y manejar la información de configuración que haya sido introducida por el usuario.

Paquete process.comparing

Para poder establecer un orden sobre los ítems y canales, la aplicación cuenta con el paquete **process.sorting.** Este paquete se encarga de aplicar los comparadores necesarios a ítems y canales para ordenar la información recopilada. Las clases del paquete **process.comparing**, le permiten a las clases de ordenamiento poder establecer una relación de orden entre los elementos que se van a comparar.

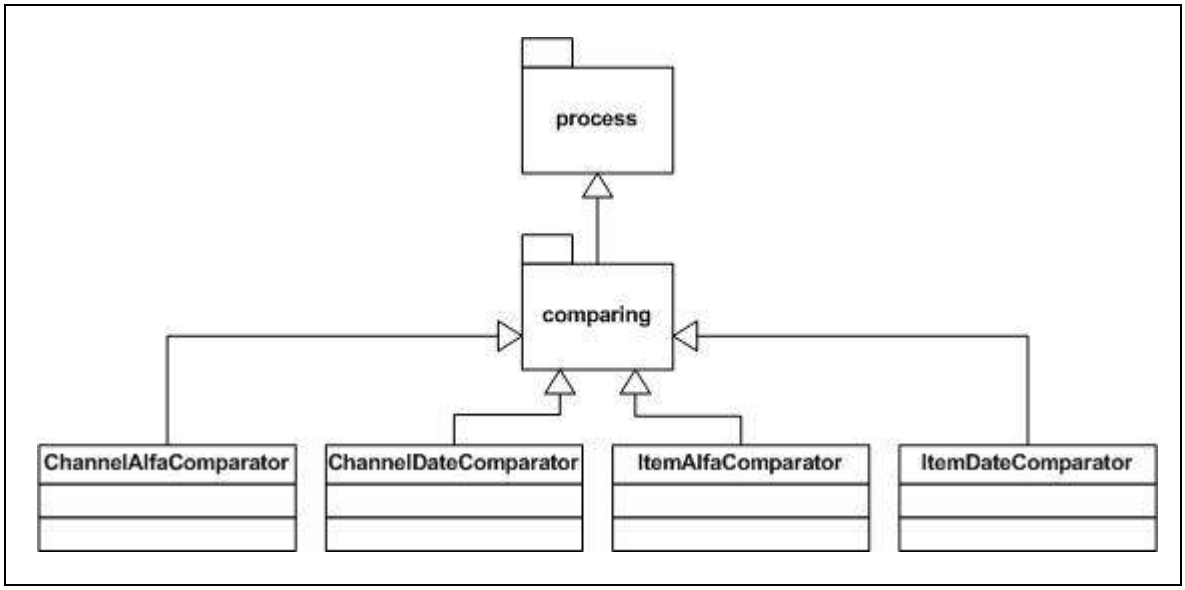

Figura 4. 9 - Paquete process.comparing

Este paquete contiene una clase por cada relación de orden establecida sobre canales e ítems.

- Clase **process.comparing.ChannelAlfaComparator**. Define una relación de orden sobre canales de acuerdo al ordenamiento alfabético de sus títulos.
- Clase **process.comparing.ChannelDateComparator**. Define una relación de orden sobre canales de acuerdo al ordenamiento de sus fechas de publicación.
- Clase **process.comparing.ItemAlfaComparator**. Define una relación de orden sobre ítems de acuerdo al ordenamiento alfabético de sus títulos.
- Clase **process.comparing.ItemDateComparator**. Define una relación de orden sobre ítems de acuerdo al ordenamiento de sus fechas de publicación.

Paquete process.parsing

La información recopilada de las fuentes es, de primera instancia, un documento RSS. Este documento debe de ser analizado para poder construir el contenido que se está buscando. Las clases de este paquete se encargan de llevar a cabo este proceso.

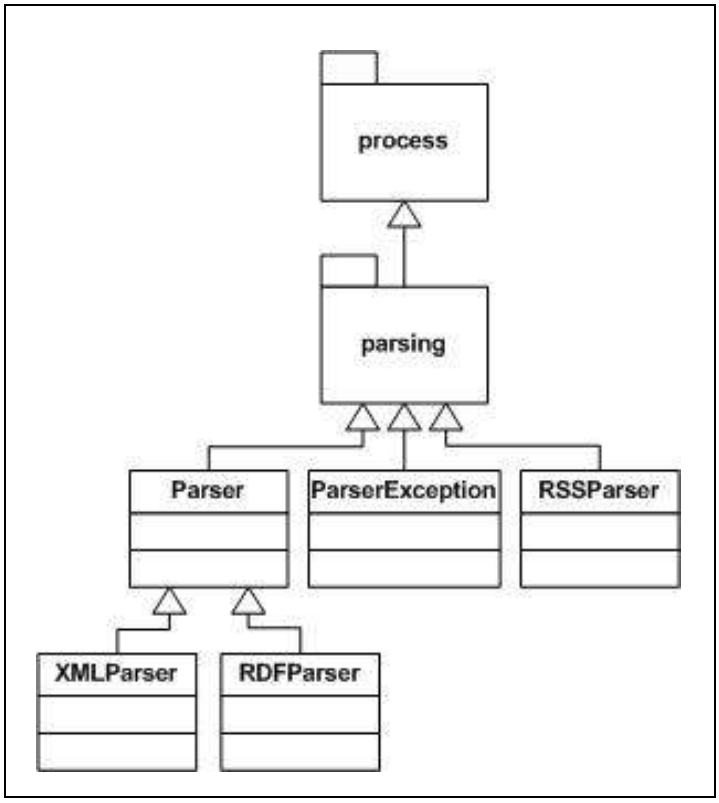

Figura 4. 10 - Paquete process.parsing

Son cinco las clases que conforman este paquete:

- Clase **process.parsing.Parser**. Clase abstracta que define el comportamiento que debe de tener un analizador de documentos RSS.
- Clase **process.parsing.XMLParser**. Clase que extiende la clase **process.parsing.Parser**. Define un analizador para documentos de RSS 2.0.
- Clase **process.parsing.RDFParser**. Clase que extiende la clase **process.parsing.Parser**. Define un analizador para documentos de RSS 1.0.

- Clase **process.parsing.ParserException**. Clase que permite manejar las excepciones que se puedan presentar al analizar un documento RSS.
- Clase **process.parsing.RSSParser**. Clase que se encarga de determinar qué analizador específico le corresponde al documento RSS que se desea analizar.

Paquete process.searching

Una vez que se tienen los datos de una fuente, el siguiente paso es hacer una conexión con la misma que arroje como resultado la recopilación de la información que se le está solicitando. Las clases del paquete **process.searching** se encargan de ejecutar los diferentes protocolos de comunicación necesarios para lograr una transacción exitosa con cada una de las fuentes.

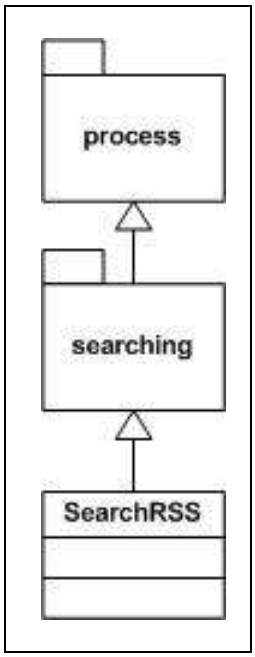

Figura 4. 11 - Paquete process.searching

La única clase miembro de este paquete es la clase **process.searching.SearchRSS** que se encarga de ejecutar la búsqueda y conexión con la fuente una vez que ya se tiene toda la información de la misma.

Paquete process.sorting

Tanto los ítems como los canales pueden ser almacenados respetando diferentes relaciones de orden, como puede ser la relación de orden definida por el ordenamiento alfabético del título del ítem o canal o la relación de orden definida por el ordenamiento de las fechas de publicación de los ítems o canales.

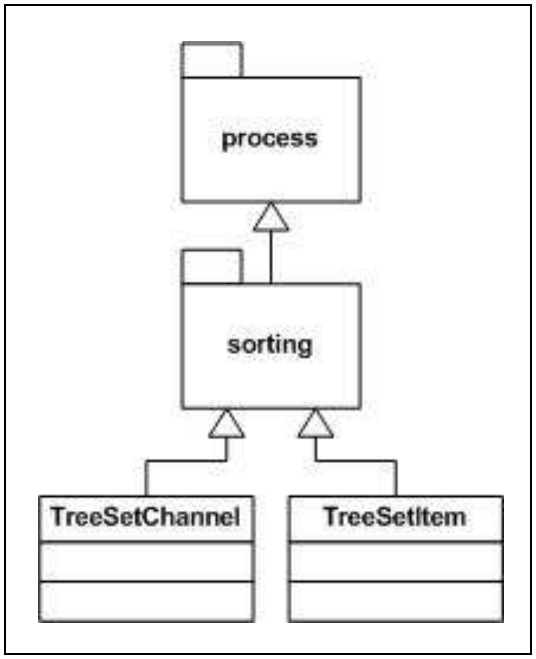

Figura 4. 12 - Paquete process.sorting

El paquete contiene una clase para ordenar canales y otra clase para ordenar ítems:

- Clase **process.sorting.TreeSetChannel**. Esta clase se encarga de ordenar canales de acuerdo a una relación de orden previamente establecida por un comparador haciendo uso de una estructura de árbol binario de búsqueda.
- Clase **process.sorting.TreeSetItem.** Esta clase se encarga de ordenar ítems de acuerdo a una relación de orden previamente establecida por un comparador haciendo uso de una estructura de árbol binario de búsqueda.

Paquete process.util

Este paquete provee a las demás clases del paquete **process** de funciones auxiliares que permitan lograr el comportamiento buscado por cada una de estas respetando la modularidad, cohesión y acoplamiento del sistema.

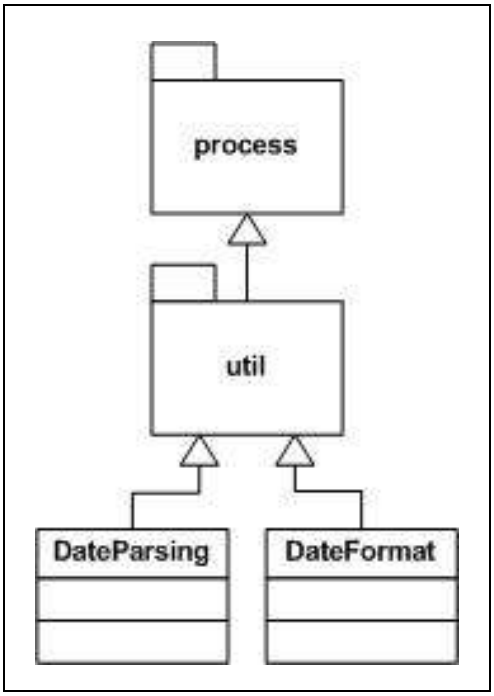

Figura 4. 13 - Paquete process.util

La clase **process.util.DateParsing** se encarga de obtener fechas de diferentes orígenes haciendo análisis sobre cadenas de texto que pueden contener horas y fechas en diferentes formatos.

Por otro lado, la clase **process.util.DateFormat** se encarga de manejar los diferentes formatos en los que se pueden presentar las fechas que se manejen en el sistema.

Diagrama general de Clases

Con toda la información presentada, el sistema estaría compuesto por la siguiente estructura jerárquica de clases.

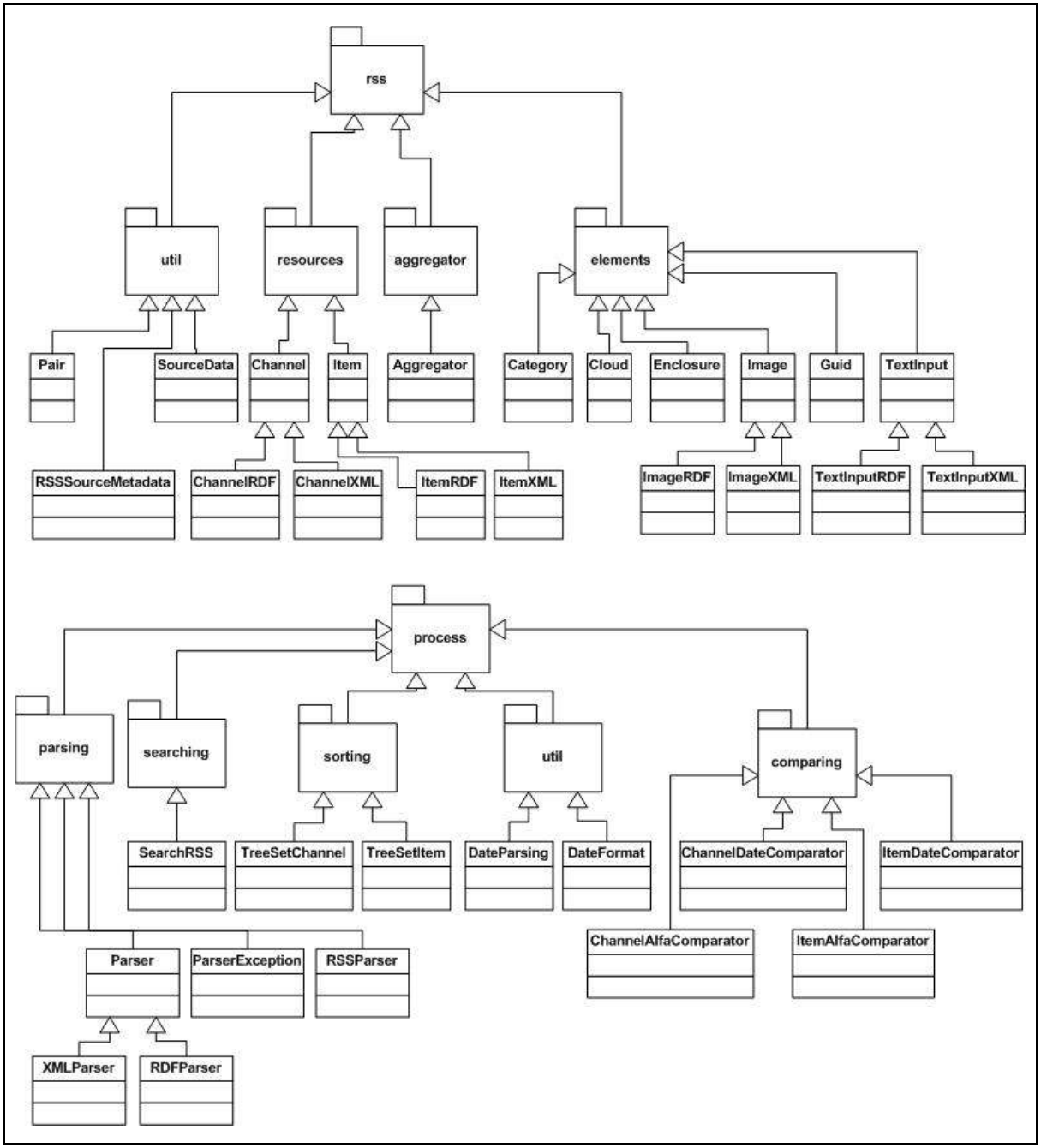

Figura 4. 14 - Estructura general completa de paquetes

Casos de Uso

Se dio un esbozo de los Casos de Uso de la aplicación en la sección de Especificación. Es trabajo del Análisis dar una versión más detallada y en lenguaje computacional de estos Casos de Uso. Primero se engloba en un solo diagrama, todos los Casos de Uso que se presentan en el sistema.

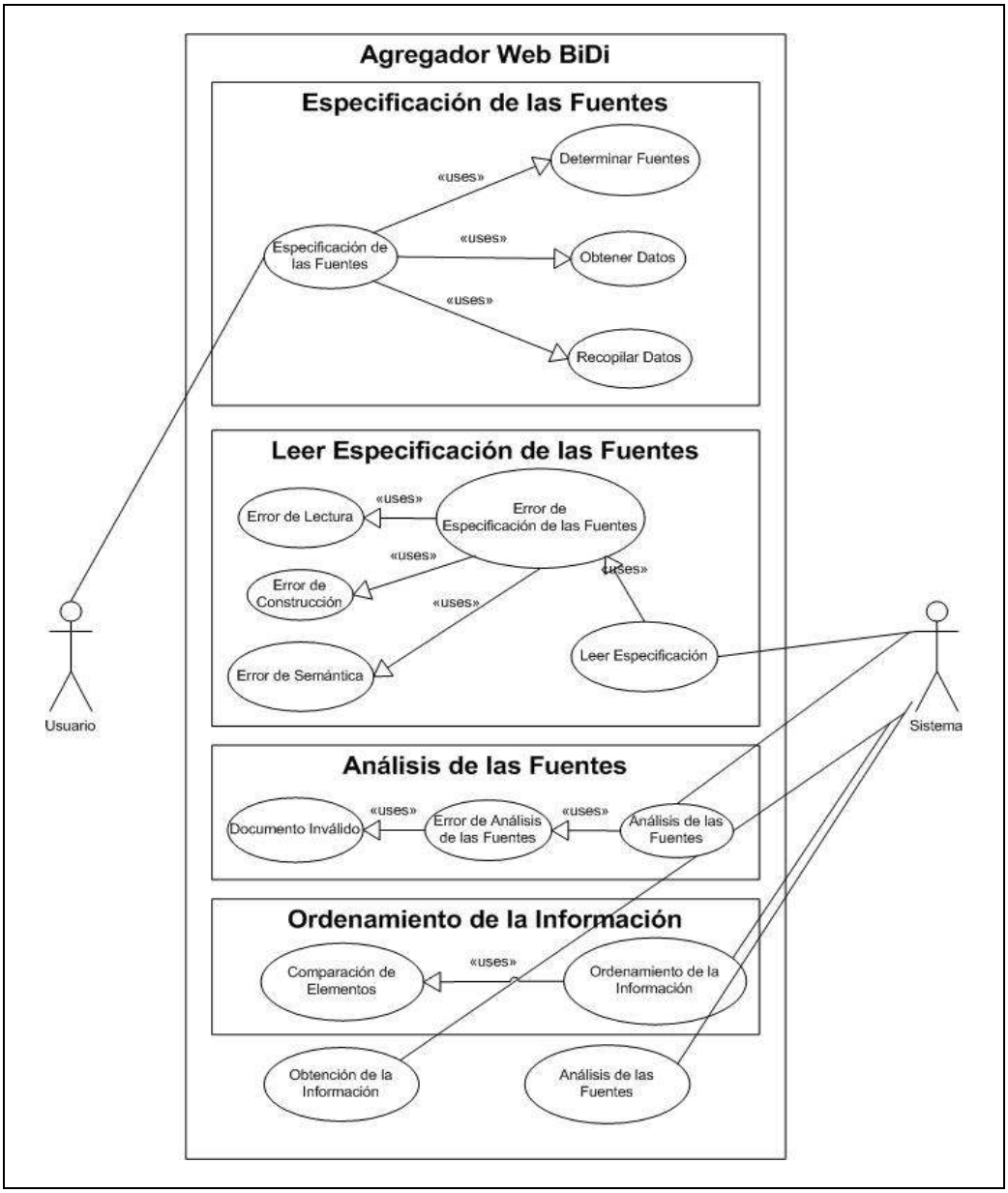

Figura 4. 15 - Casos de Uso del Sistema

Diagramas de Colaboración

Los Diagramas de Interacción modelan el comportamiento de los Casos de Uso describiendo la forma en la que grupos de objetos interactúan entre sí para completar una tarea específica. Existen dos tipos de Diagramas de Interacción: los Diagramas de Colaboración y los Diagramas de Secuencia.

Los Diagramas de Colaboración muestran la relación entre objetos y el orden de los mensajes que son enviados entre ellos.

Caso de Uso: Leer Especificación de las Fuentes

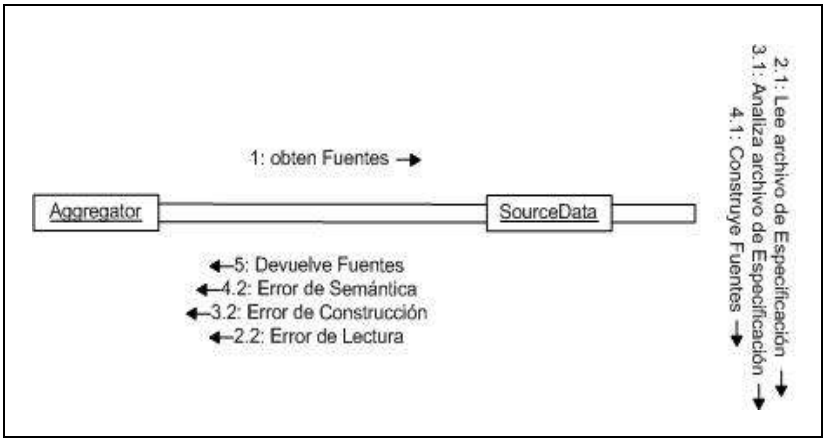

Figura 4. 16 - Diagrama de Colaboración: Leer Especificación de las Fuentes

Caso de Uso: Obtención de la Información

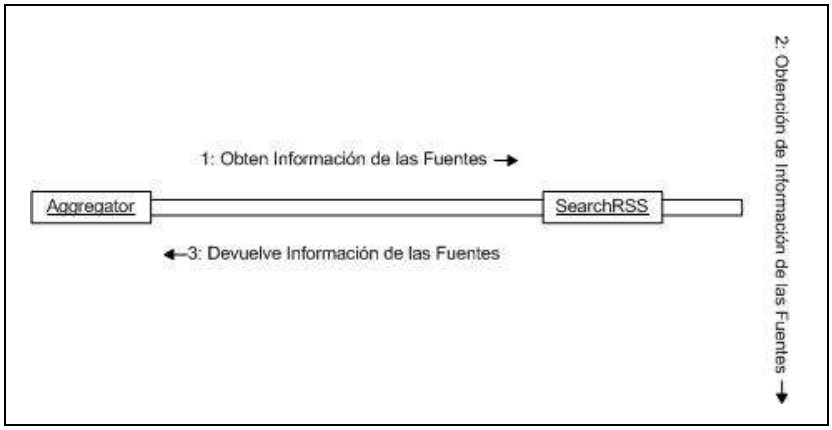

Figura 4. 17 - Diagrama de Colaboración: Obtención de la Información

Caso de Uso: Análisis de las Fuentes

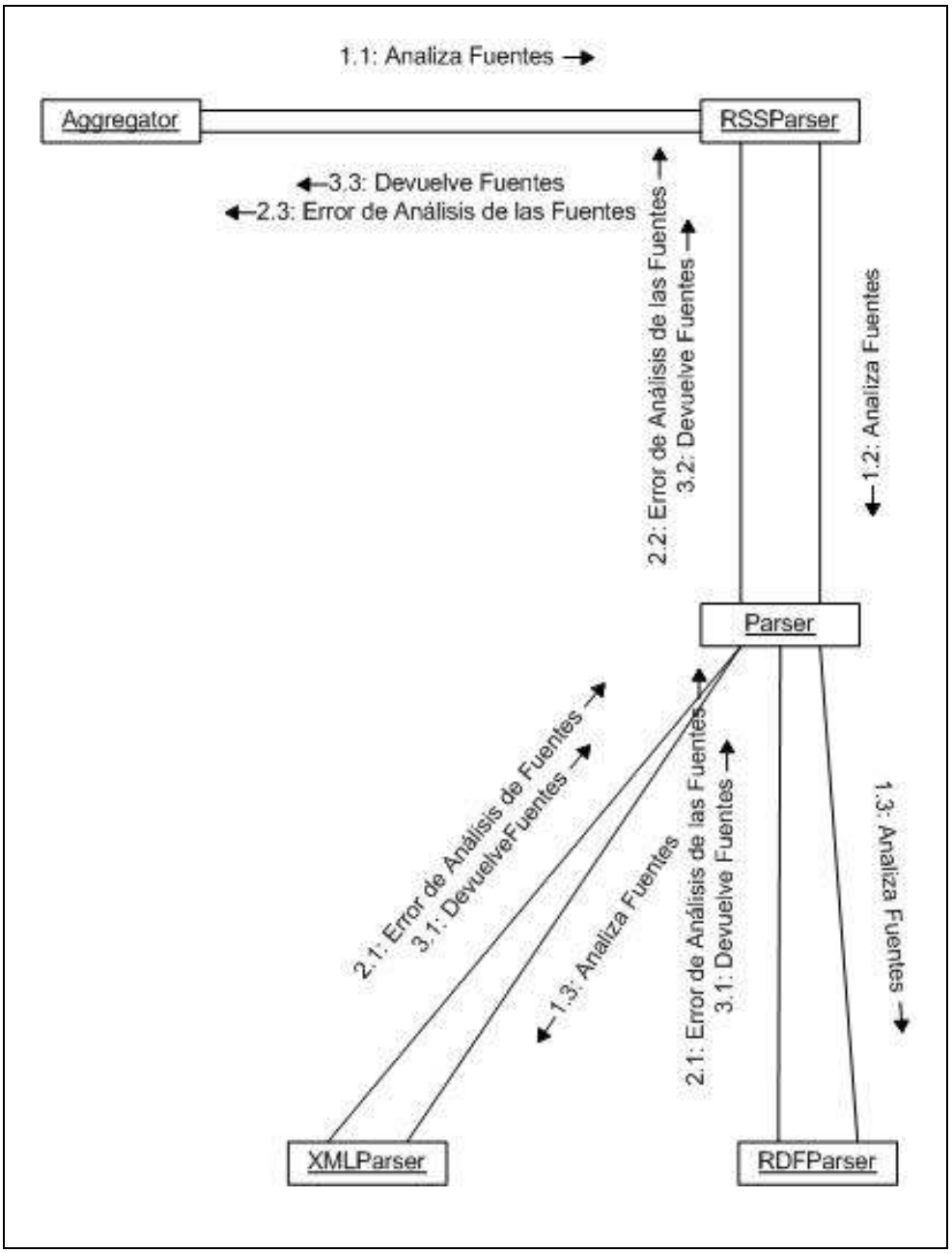

Figura 4. 18 - Diagrama de Colaboración: Análisis de las Fuentes

Caso de Uso: Ordenamiento de la Información

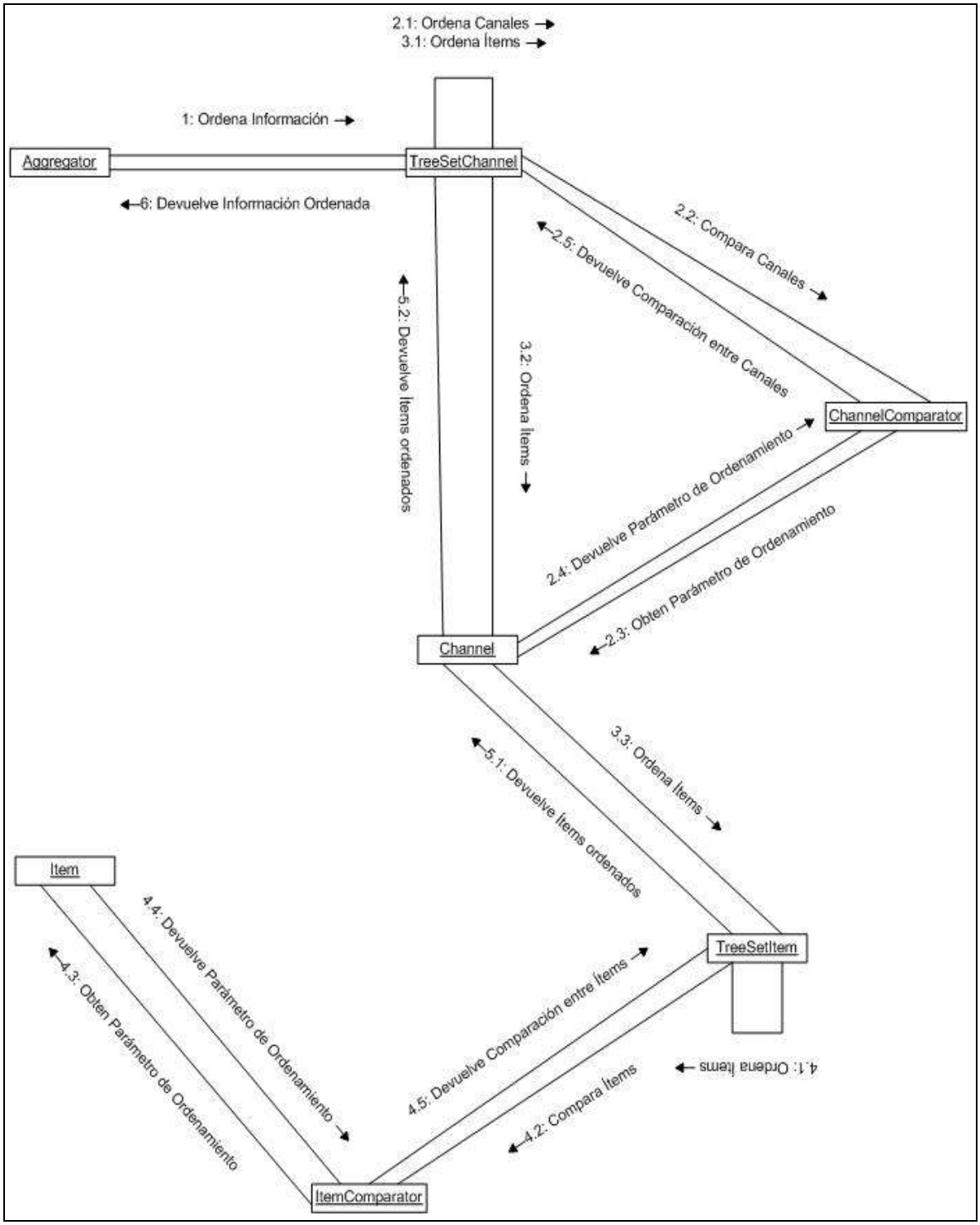

Figura 4. 19 - Diagrama de Colaboración: Ordenamiento de la Información

Diagramas de Secuencia

Los Diagramas de Secuencia son el otro tipo de Diagramas de Interacción. El objetivo de estos Diagramas es demostrar el comportamiento de los objetos en un Caso de Uso mediante la descripción de los mensajes que éstos envían.

Caso de Uso: Análisis de las Fuentes

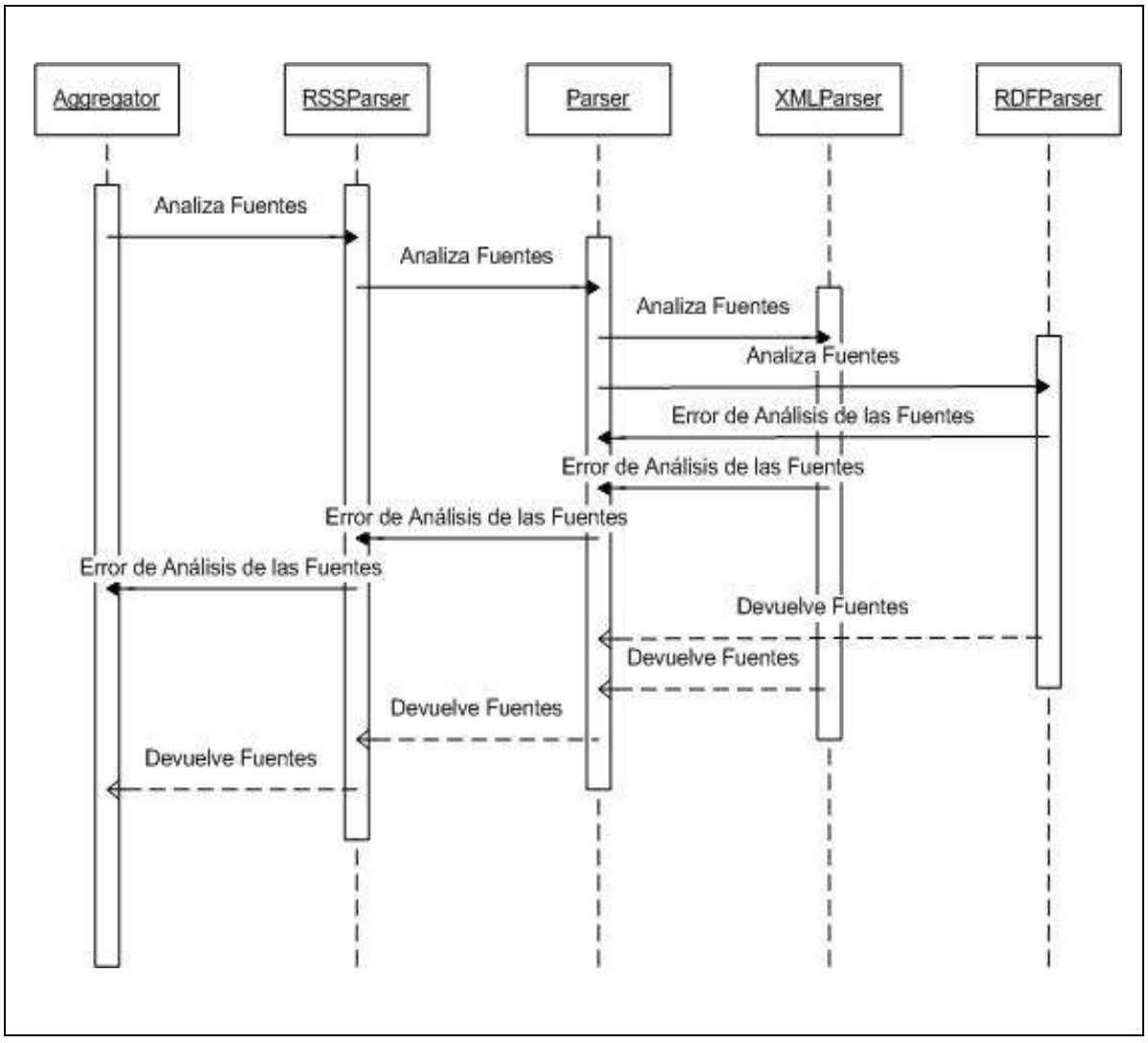

Figura 4. 20 - Diagrama de Secuencia: Análisis de las fuentes

Caso de Uso: Leer Especificación de las Fuentes

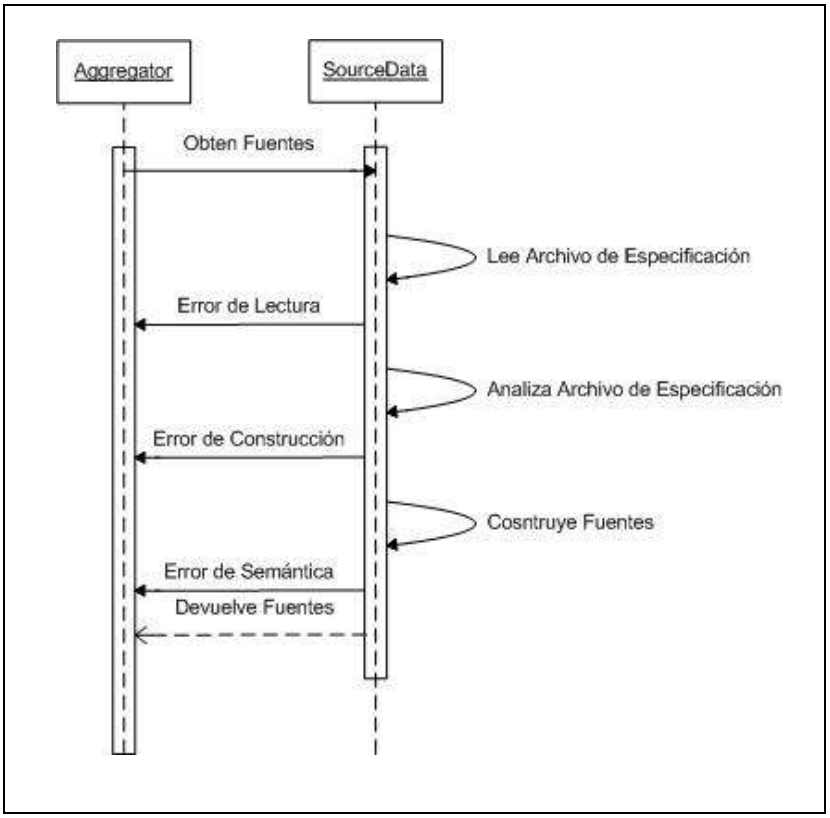

Figura 4. 21 - Diagrama de Secuencia: Leer Especificación de las Fuentes

Caso de Uso: Obtención de la Información

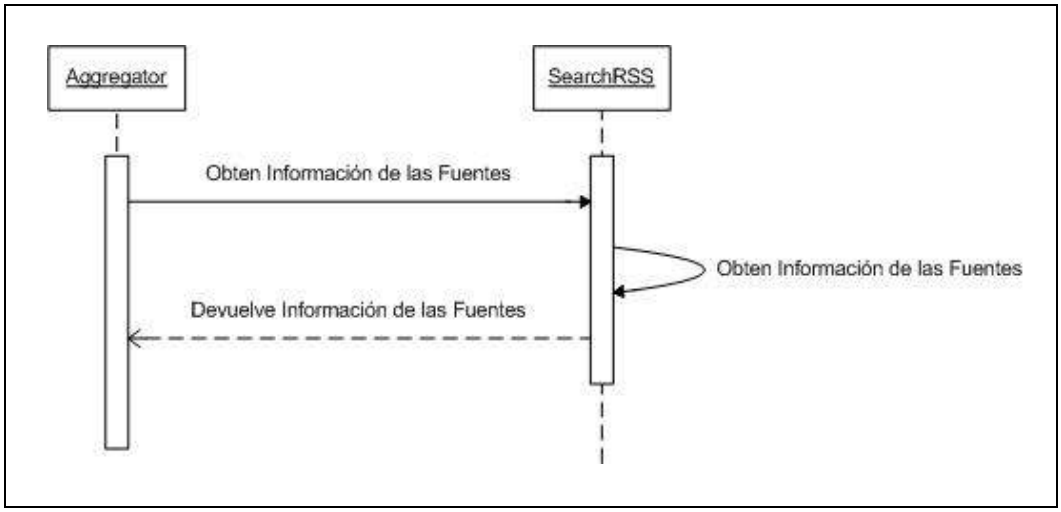

Figura 4. 22 - Diagrama de Secuencia: Obtención de la información

Caso de Uso: Ordenamiento de la Información

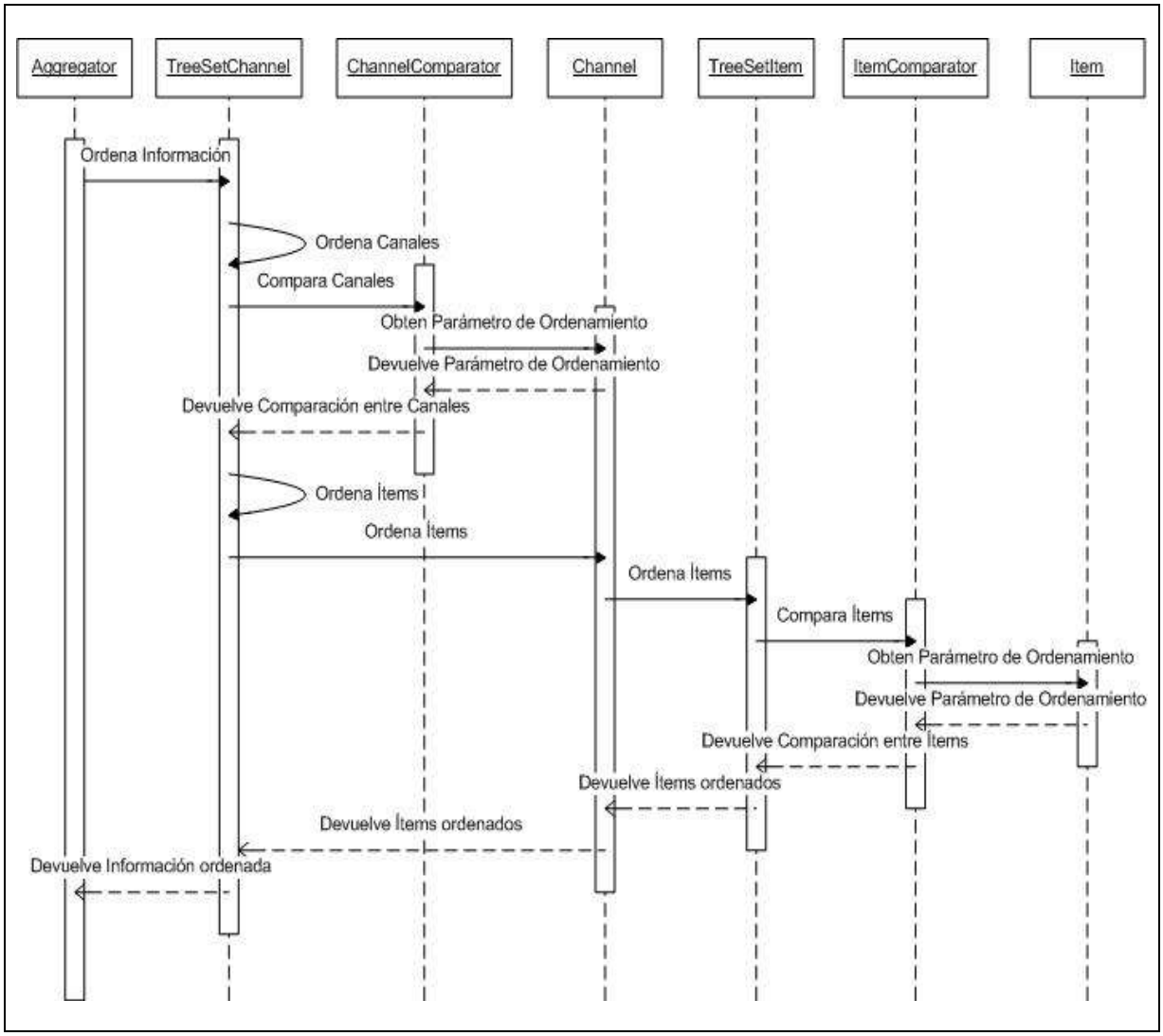

Figura 4. 23 - Diagrama de Secuencia: Ordenamiento de la Información

Conclusiones

A lo largo de este capítulo se documentó el Análisis de la aplicación Agregador Web BiDi. Gracias a los Diagramas incluidos en esta sección, se puede continuar con la siguiente fase de Ingeniería de Software que es el Diseño de la aplicación.

Mientras que la Especificación da una idea general y en lenguaje humano de cuál debe de ser la funcionalidad del Agregador, el Análisis toma los elementos que la Especificación brinda y los convierte a lenguaje Computacional.

Para que el grado de especificación sea mayor sin perder la fácil comprensión, se hace uso de Diagramas UML que son un estándar internacional y bien aceptado. Los Diagramas UML usados durante el Análisis fueron los Diagramas de Casos de Uso y los Diagramas de Interacción.

CAPITULO 5
Diseño del Sistema

En este etapa del proceso profundiza más el contenido mostrado en el Análisis. Dentro del Diseño del Sistema Agregador Web BiDi se incluye:

- Especificación de los Atributos de las Clases
- Especificación de los Métodos de las Clases
- Especificación de los tipos de regreso de los Métodos de las Clases
- Especificación de los parámetros recibidos por los Métodos de las Clases

Para lograr los cuatro puntos anteriores, se hace uso de Diagramas de Clases UML siguiendo la misma especificación vista en las secciones anteriores de este documento.

Introducción

Con el objetivo de dar una visión estructurada de la jerarquía de clases del Sistema Agregador Web BiDi, se dará inicio a este documento de Diseño mostrando modularmente los diagramas de clase de cada uno de los paquetes que conforman el Sistema. Como se describió en el capítulo anterior, se muestra en la Figura 5.1 el Diagrama de Paquetes que clasifican las clases del Agregador.

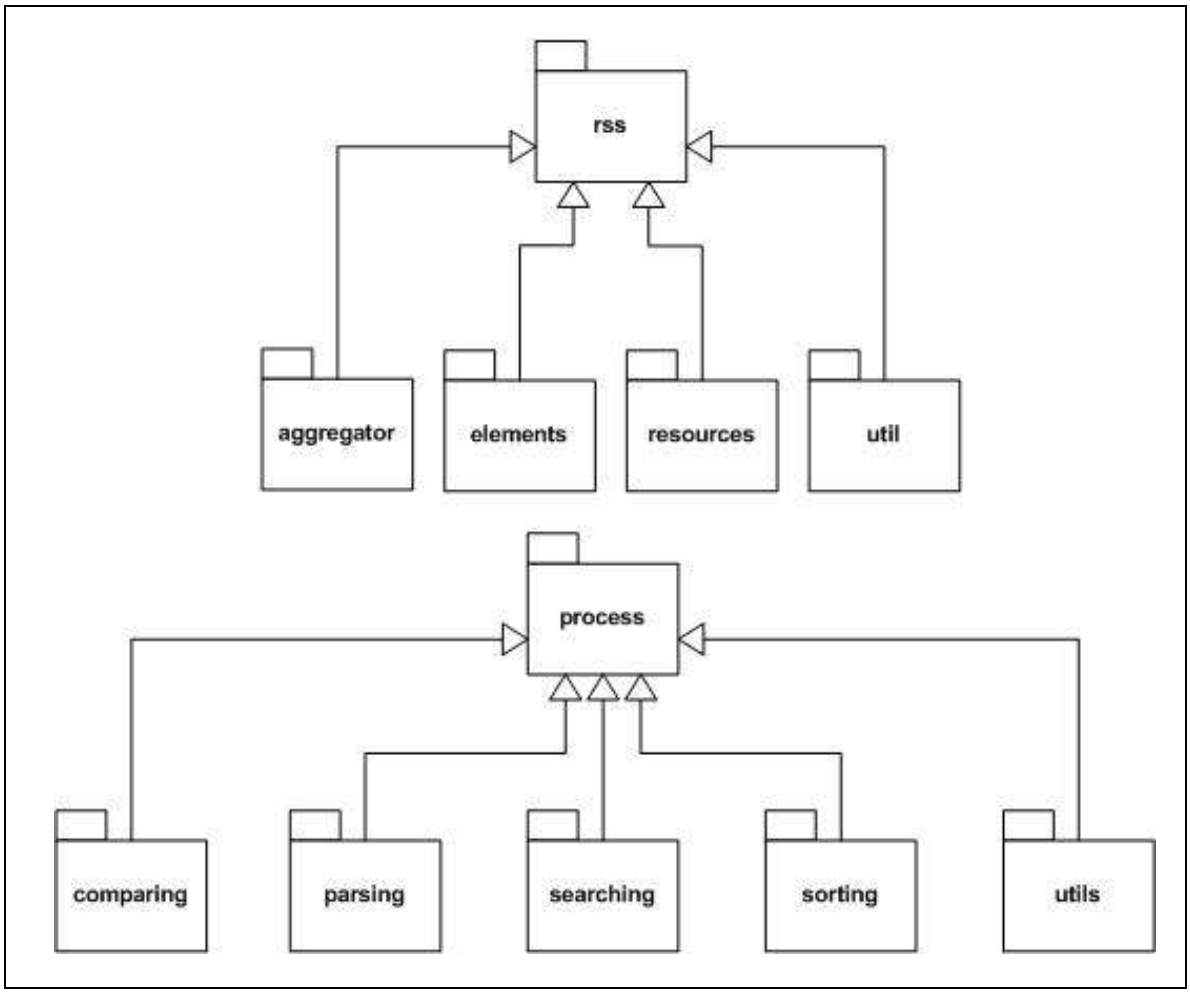

Figura 5. 1 - Estructura de Paquetes del Sistema Agregador Web BiDi

Paquete rss

Paquete rss.aggregator

El paquete rss.aggregator se conforma por una clase que tiene un papel primordial dentro del Sistema completo al ser la clase principal (a la que llamaríamos clase ejecutable si este Sistema no fuera una biblioteca).

La clase *rss.aggregator.Aggregator* tiene 4 importantes atributos:

- *LinkedList<RSSSourceMetadata> sources*. El atributo *sources* es una lista ligada de objetos de la clase *rss.util.RSSourceMetadata* (la cual se detallará más adelante). Esta lista ligada almacenará los datos de las fuentes de las que se obtendrá información. Los datos almacenados en esta lista son los que permitirán establecer una conexión exitosa con las fuentes. Cada entrada de esta lista tiene entonces una relación directa con una de las fuentes seleccionadas para consultar.
- *SourceData sourceData*. El atributo *sourceData* de la clase *rss.util.SoruceData* se encargará de leer los datos de todas las fuentes por consultar del archivo de configuración XML que el usuario haya proporcionado. A partir de este objeto será que se podrá construir la lista *sources*.
- *LinkedList<Channel> buffer*. Esta lista ligada almacenará la información obtenida de las fuentes. Como se mencionó en secciones anteriores, el Contenido Sindicado tiene una jerarquía bien marcada encabezada por un elemento denominado Canal. Es por esto que los elementos de esta lista son objetos de la clase *rss.resources.Channel*.
- *TreeSetChannel channelSet*. Esta estructura permitirá establecer un orden sobre la lista de objetos *Channel* almacenada en el atributo *buffer* mediante un sistema de indexación.

La Figura 5.2 muestra el Diagrama de Clases UML del paquete rss.aggregator.

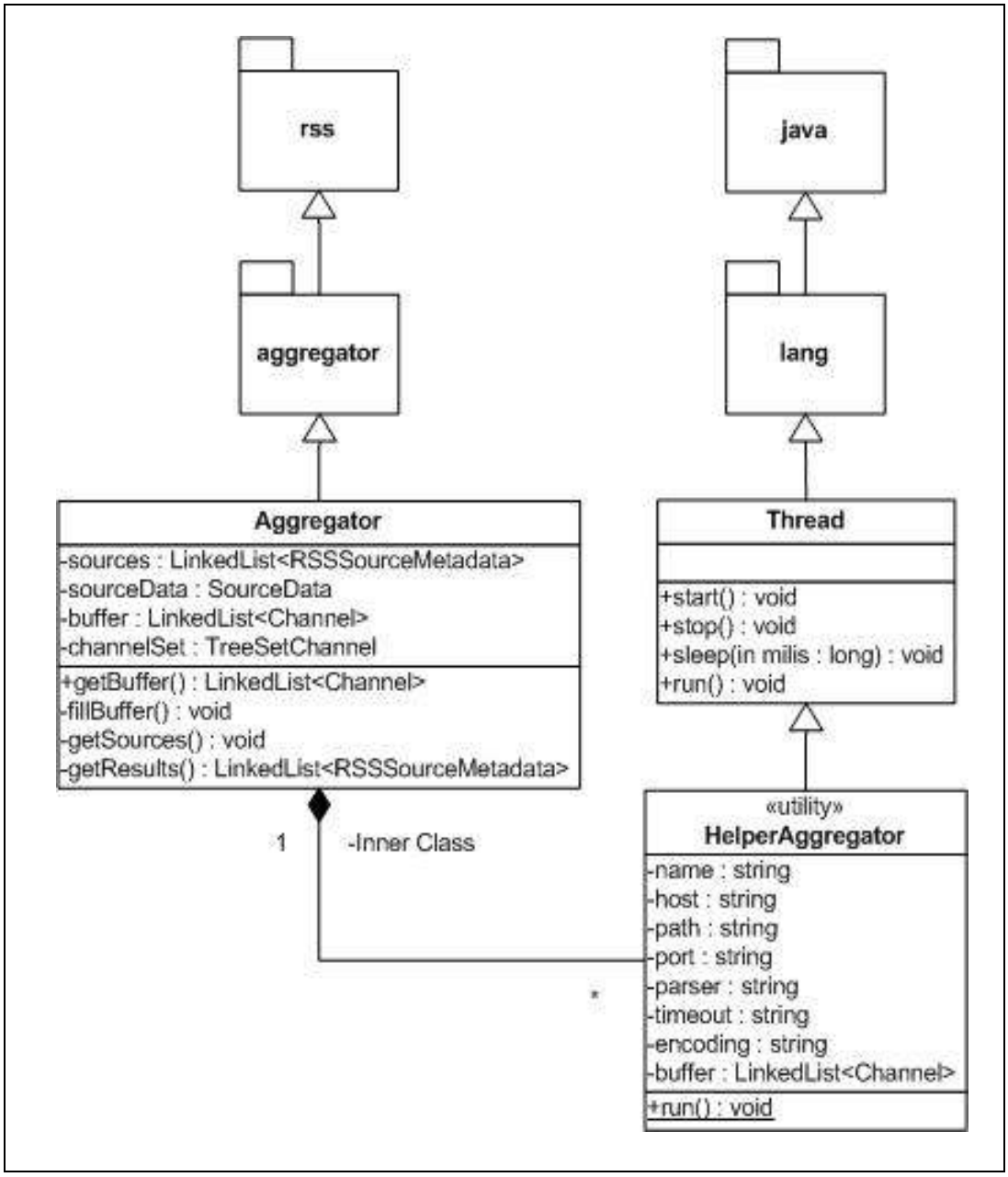

Figura 5. 2 - Diagrama de Clases de rss.aggregator.Aggregator

La clase rss.aggregator.Aggregator contiene dos Listas Ligadas que almacenarán las fuentes y, posteriormente, el Contenido Sindicado obtenido de dichas fuentes. Se decidió usar Listas Ligadas como Estructura de Datos por su propiedad de poder crecer indefinidamente –no se sabe de antemano cuantos canales se pueden recibir de una fuente ni cuantas fuentes serán analizadas- y por permitir libre iteración entre sus elementos –para poder realizar la recopilación y ordenamiento del Contenido Sindicado-.

La clase HelperAggregator es una clase interna de rss.aggregator.Aggregator. Esta clase permite que la recopilación de Contenido Sindicado se haga mediante Multiprocesamiento construyendo un hilo de ejecución diferente para cada fuente; esta decisión de Diseño hace a la aplicación más eficiente al reducir el tiempo de recopilación solapando el tiempo de espera que se tiene con cada una de las peticiones http que se hacen para obtener el Contenido Sindicado.

Esta clase se define como interna debido a que solamente es usada dentro del método getResources de la clase rss.aggregator.Aggregator y no tiene el objetivo de ser completa por lo que se busca evitar que otras clases quieran hacer uso de ella.

Paquete rss.elements

Las clases dentro de este paquete son los elementos definidos en la especificación del estándar RSS mencionada en secciones anteriores.

Los atributos de las clases de este paquete son entonces los elementos definidos en la especificación de cada uno de ellos.

La Figura 5.3 muestra el Diagrama de Clases UML de la clase rss.elements.Category, mientras que la Figura 5.4 muestra el Diagrama de Clases UML de la clase rss.elements.Cloud. La Figura 5.5 muestra el Diagrama de la clase rss.elements.Enclosure.

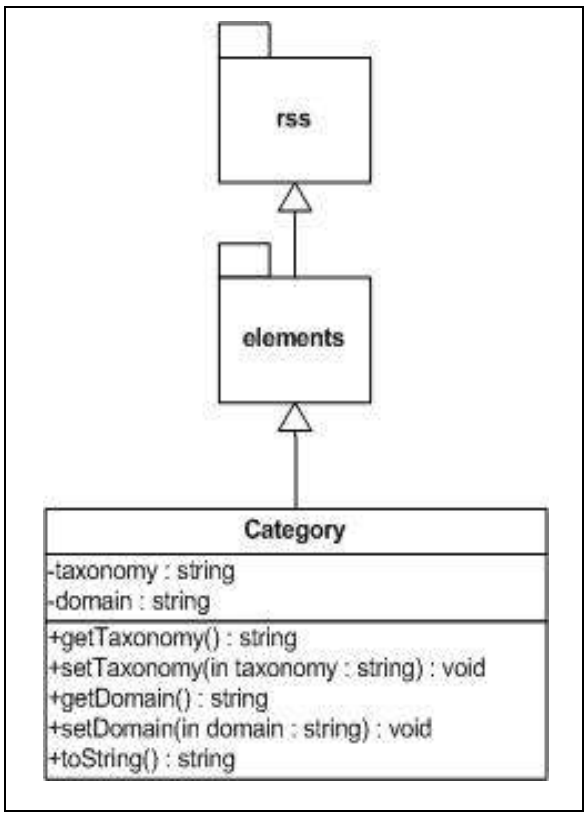

Figura 5. 3 - Diagrama de Clases de rss.elements.Category

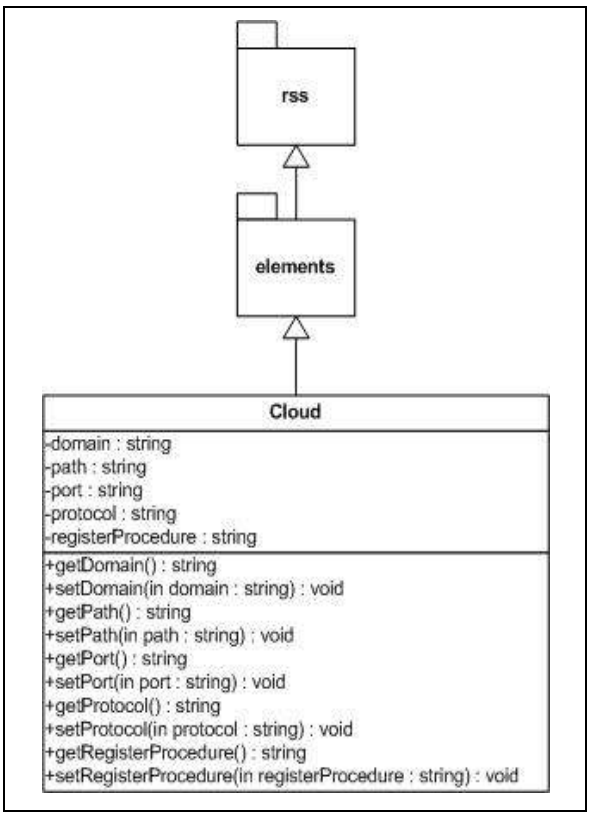

Figura 5. 4 - Diagrama de Clases de rss.elements.Cloud

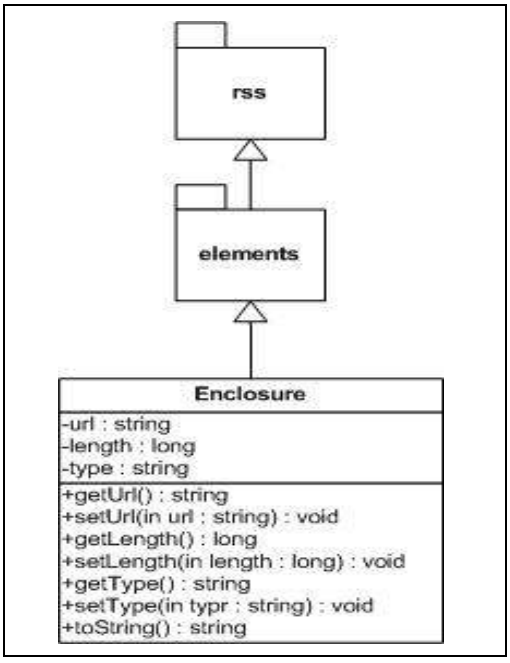

Figura 5. 5 - Diagrama de Clases de rss.elements.Enclosure

La Figura 5.6 muestra el Diagrama de Clases UML de la clase rss.elements.Guid.

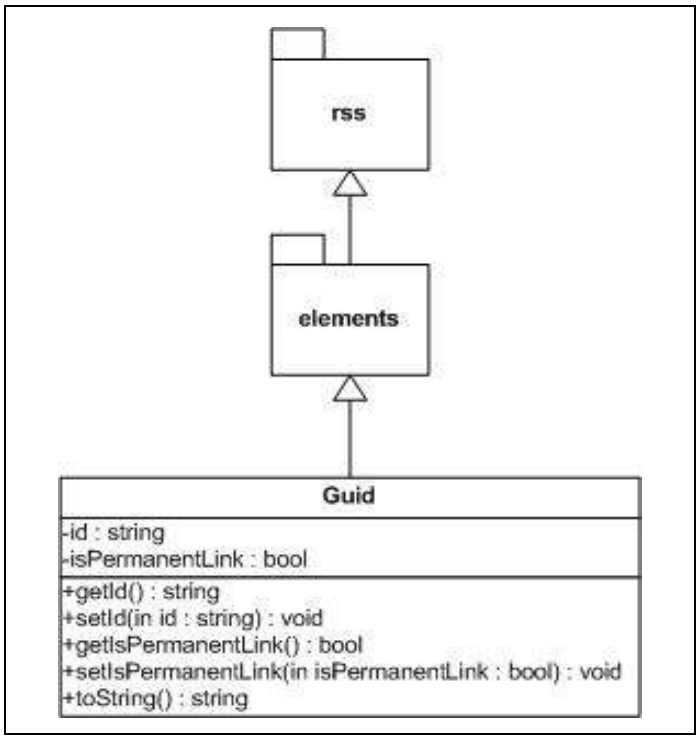

Figura 5. 6 - Diagrama de Clases de rss.elements.Guid

La clase rss.elements.Guid contiene un atributo booleano llamado *isPermanentLink*. Este atributo define si la liga o *link* definida por este objeto se considera permanente o no. Debido a que solo se tienen dos opciones (ser permanente o no serlo) este atributo se define como booleano y no como una cadena (como el resto de los atributos de los elementos RSS).

La Figura 5.7 muestra el Diagrama de Clases UML de la clase rss.elements.Image y sus clases hijas rss.elements.ImageRDF y rss.elements.ImageXML.

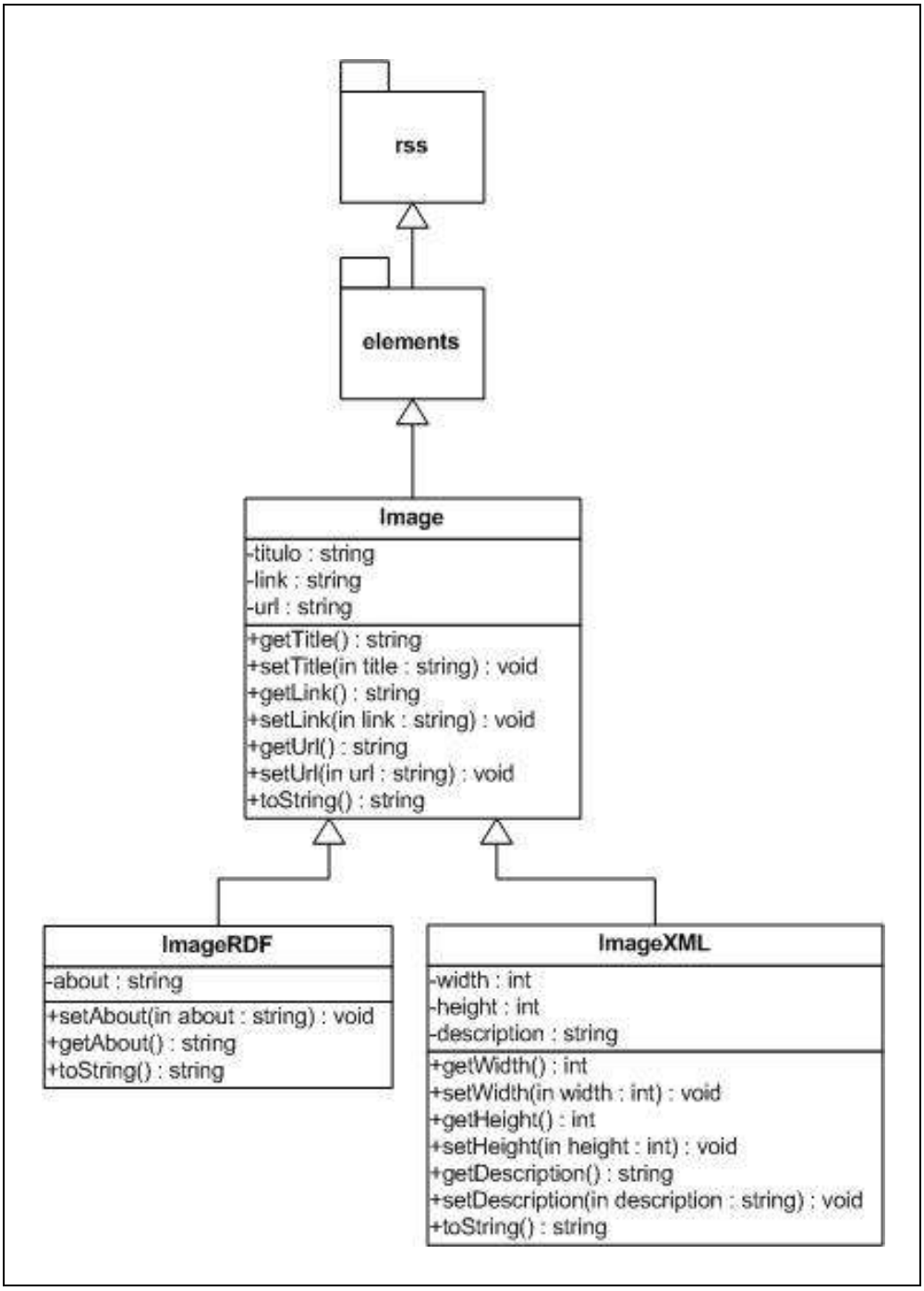

Figura 5. 7 - Diagrama de Clases de rss.elements.Image y sus clases hijas

La Figura 5.8 muestra el Diagrama de Clases UML de la clase rss.elements.TextInput y de sus clases hijas rss.elements.TextInputRDF y rss.elements.TextInputXML.

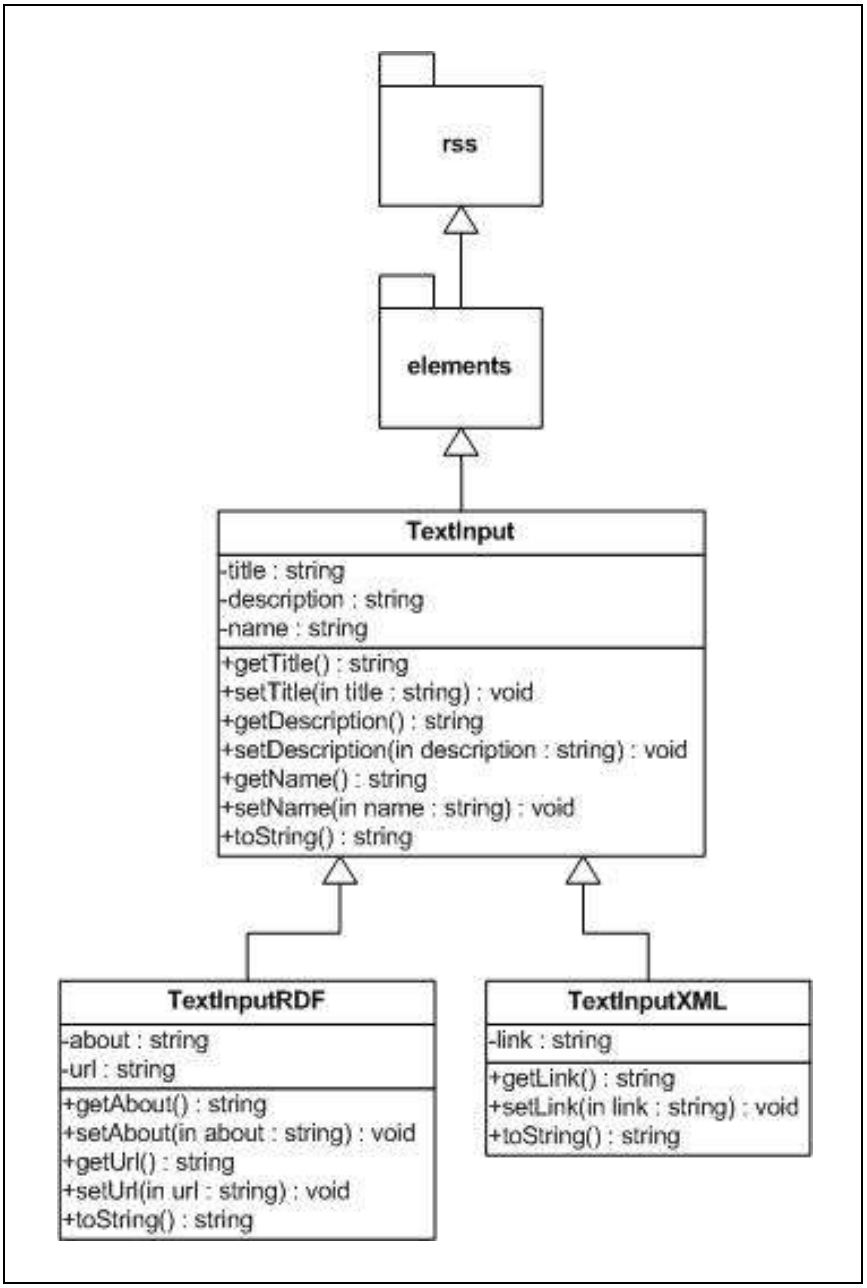

Figura 5. 8 - Diagrama de Clases de rss.elements.TextInput y sus clases hijas

Paquete rss.resources

La Figura 5.9 muestra el Diagrama de Clases UML de la clase rss.resources.Channel y sus clases hijas rss.resources.ChannelRDF y rss.resources.ChannelXML.

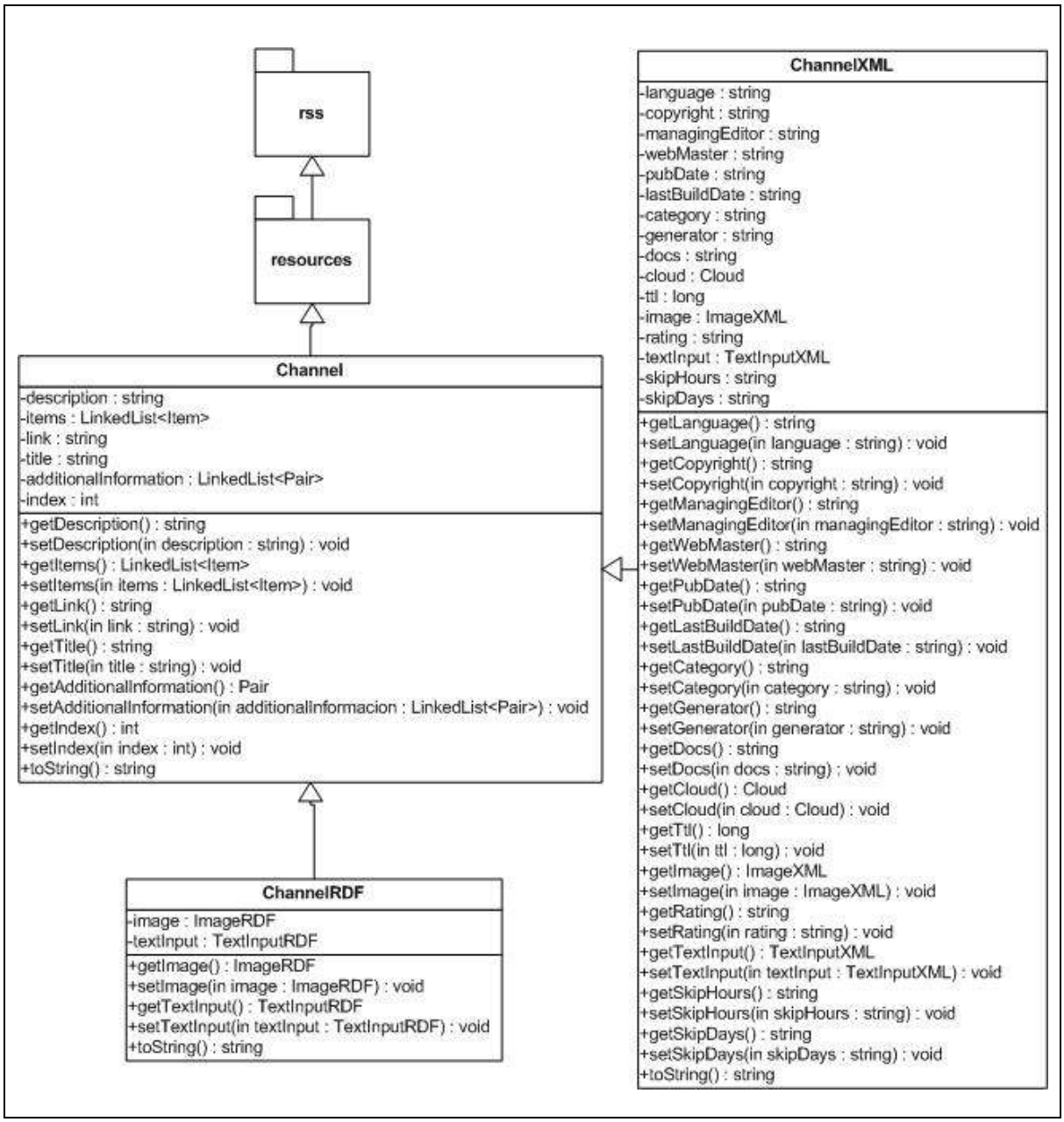

Figura 5. 9 - Diagrama de Clases de rss.resources.Channel y sus clases hijas

La Figura 5.10 muestra el Diagrama de Clases UML de la clase rss.resources.Item y sus clases hijas rss.resources.ItemRDF y rss.resources.ItemXML.

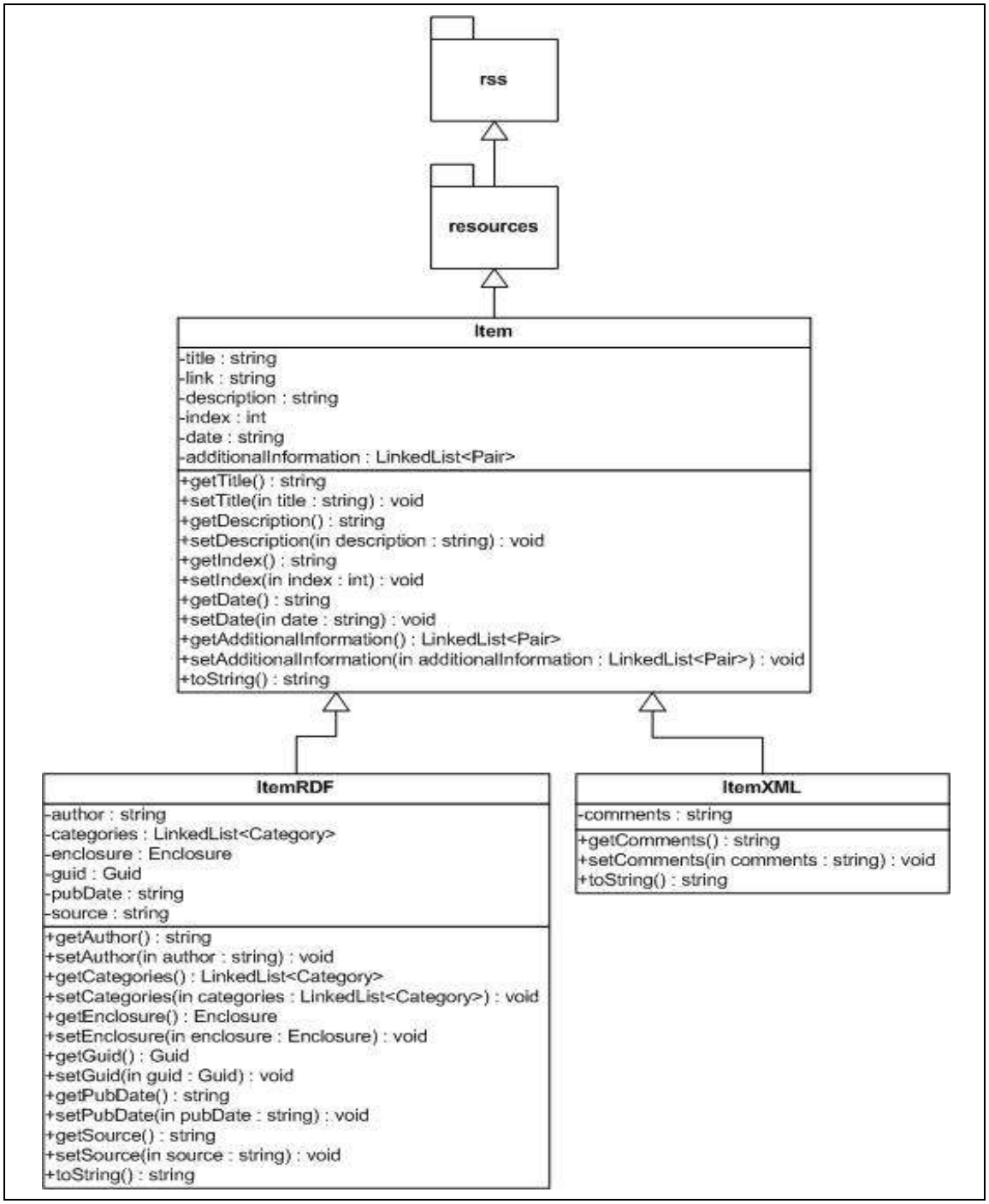

Figura 5. 10 - Diagrama de Clases de rss.resources.Item y sus clases hijas

Paquete rss.util

La Figura 5.11 muestra el Diagrama de Clases UML de la clase rss.util.Pair.

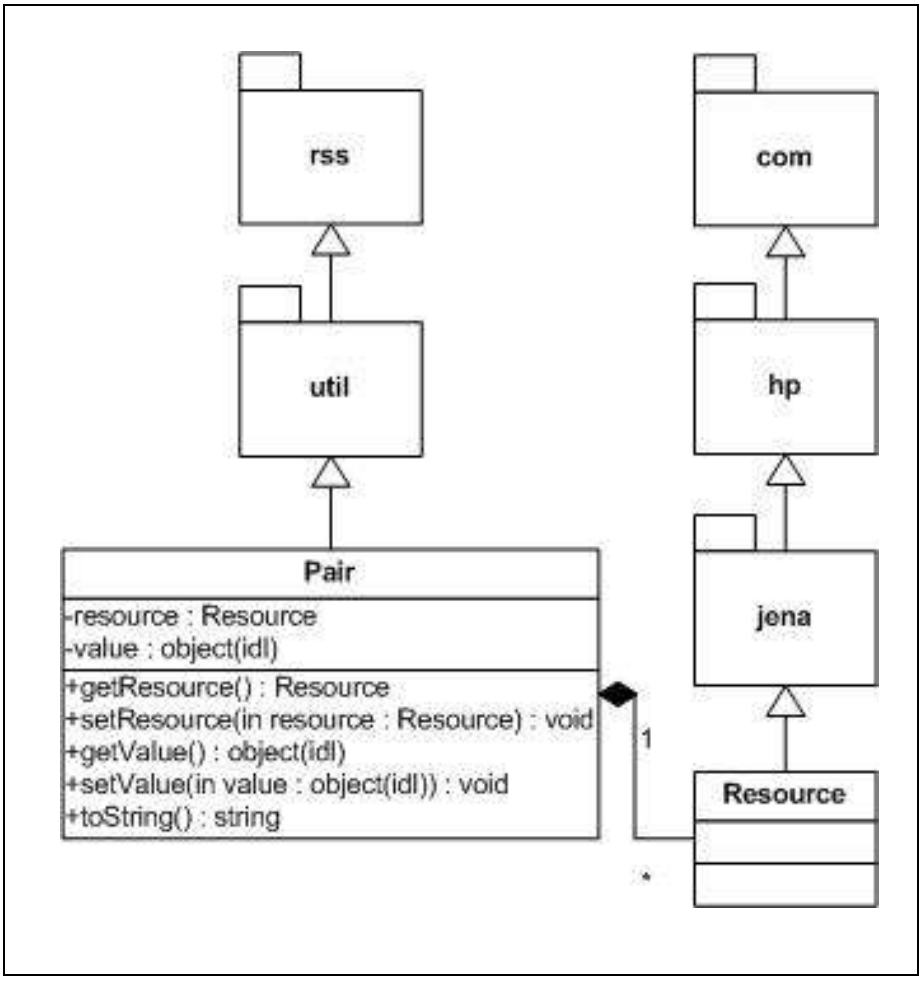

Figura 5. 11 - Diagrama de Clases de rss.util.Pair

La Figura 5.12 muestra el Diagrama de Clases UML de la clase rss.util.RSSSourceMetadata.

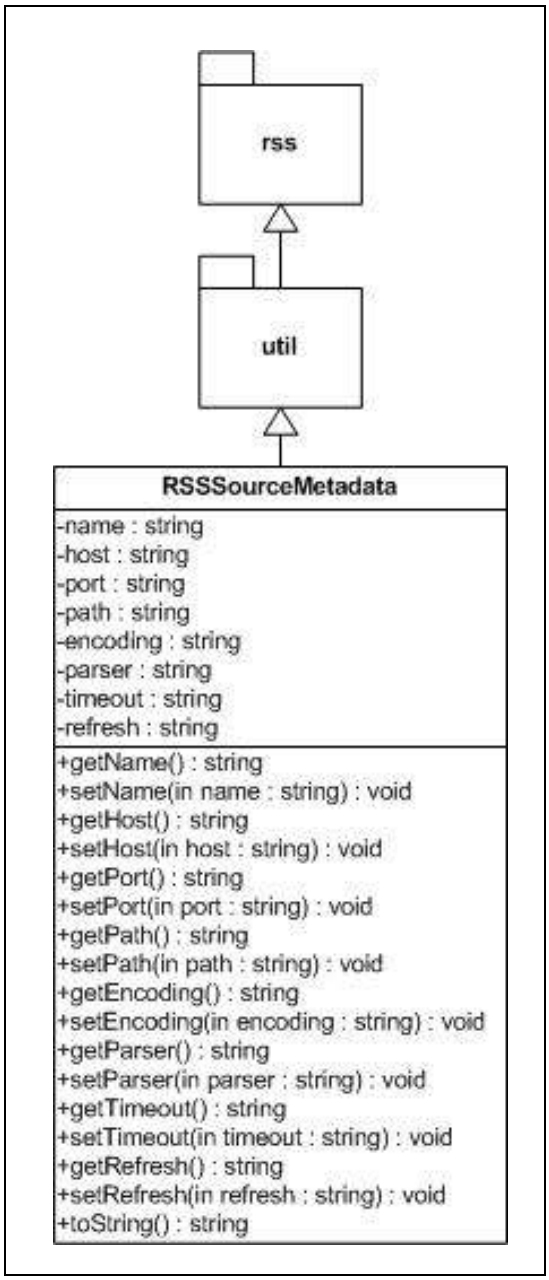

Figura 5. 12 - Diagrama de Clases de rss.util.RSSSourceMetadata

La Figura 5.13 muestra el Diagrama de Clases UML de la clase rss.util.SourceData.

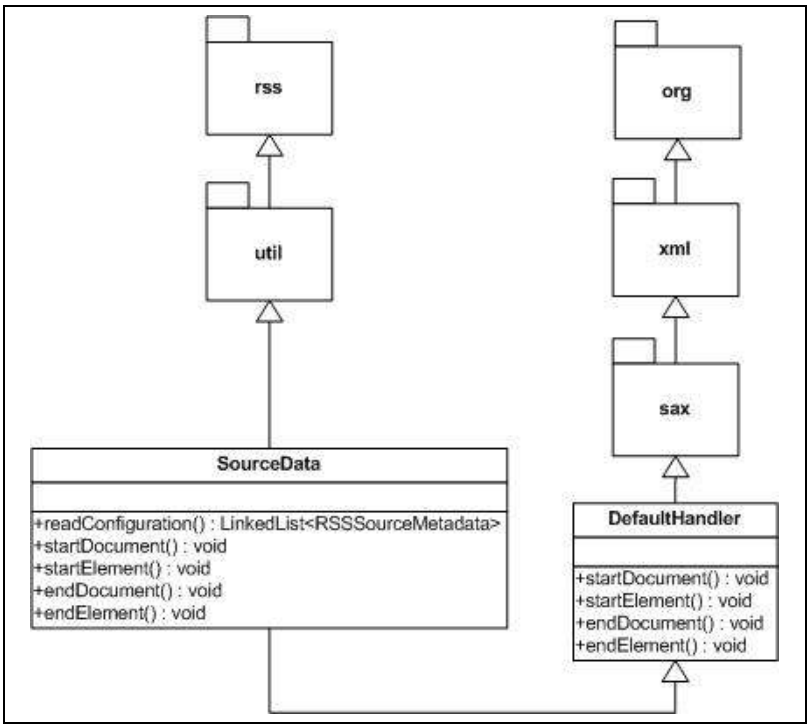

Figura 5. 13 - Diagrama de Clases de rss.util.SourceData

Paquete process

Paquete process.comparing

La Figura 5.14 muestra el Diagrama de Clases UML de la clase process.comparing.ChannelAlfaComparator.

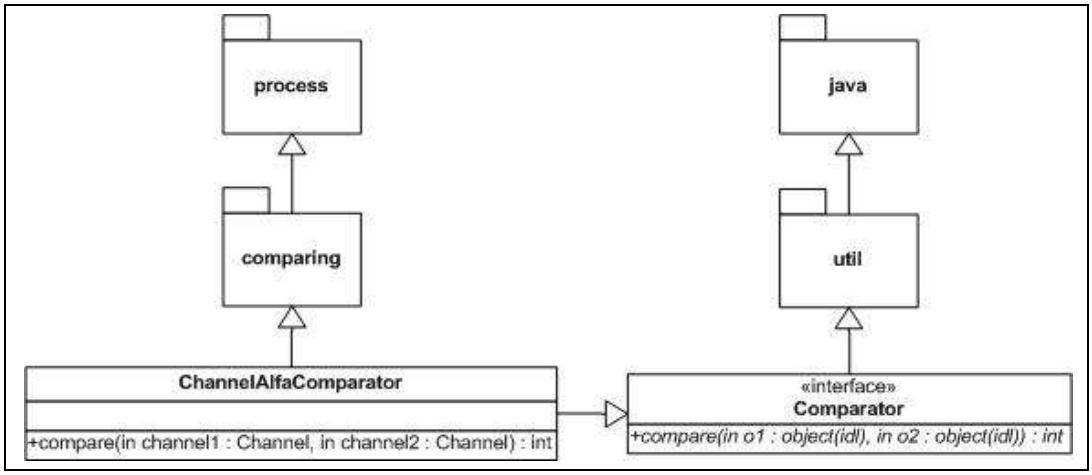

Figura 5. 14 - Diagrama de clases de process.comparing.ChannelAlfaComparator

La Figura 5.15 muestra el Diagrama de Clases UML de la clase process.comparing.ChannelDateComparator.

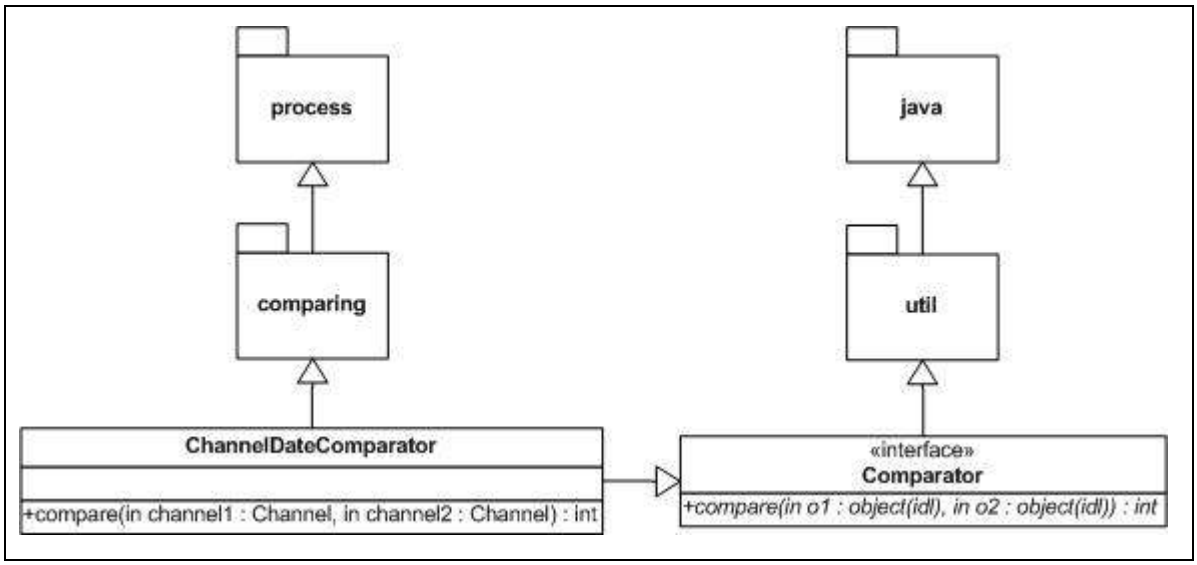

Figura 5. 15 - Diagrama de Clases de process.comparing.ChannelDateComparator

La Figura 5.16 muestra el Diagrama de Clases UML de la clase process.comparing.ItemAlfaComparator.

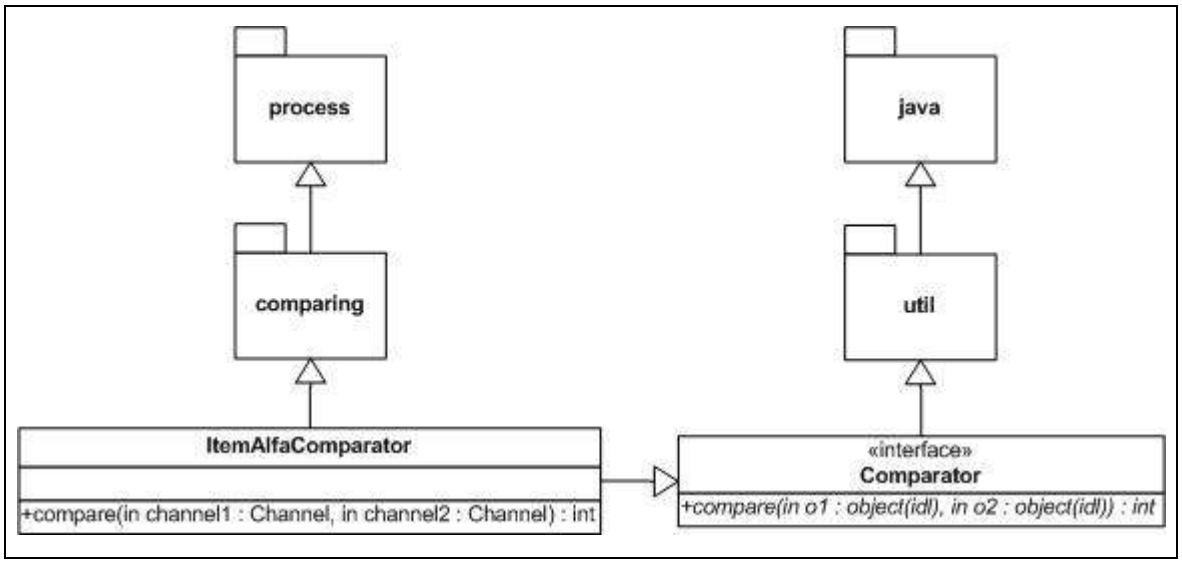

Figura 5. 16 - Diagrama de Clases de process.comparing.ItemAlfaComparator

La Figura 5.17 muestra el Diagrama de Clases UML de la clase process.comparing.ItemDateComparator.

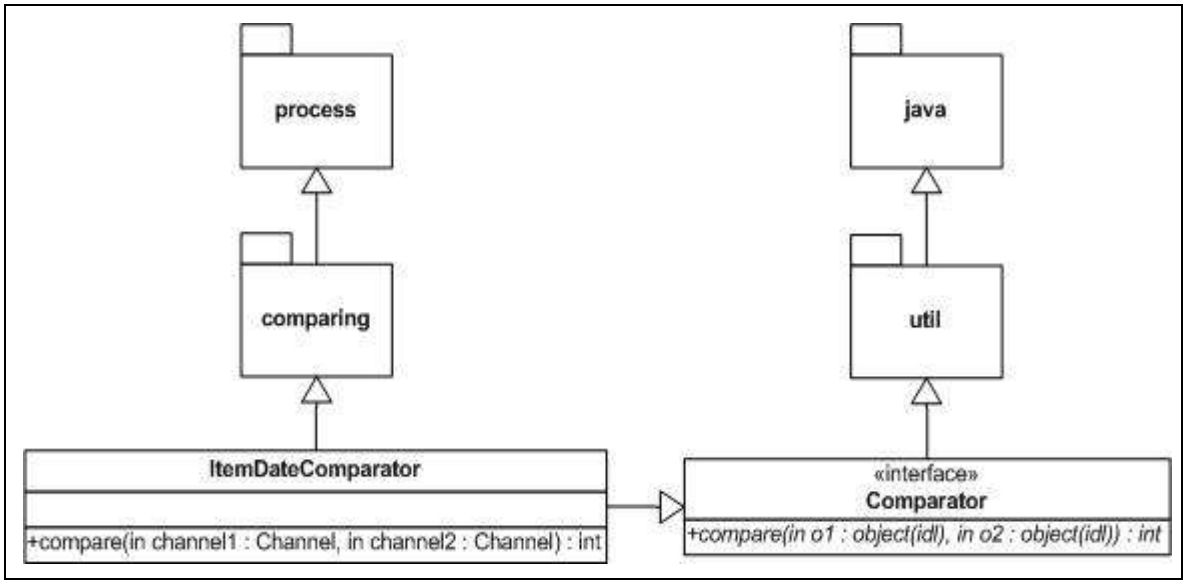

Figura 5. 17 - Diagrama de Clases de process.comparing.ItemDateComparator

Paquete process.parsing

La Figura 5.18 muestra el Diagrama de Clases UML de la clase process.parsing.XMLParser junto con el diagrama de su clase padre: process.parsing.Parser.

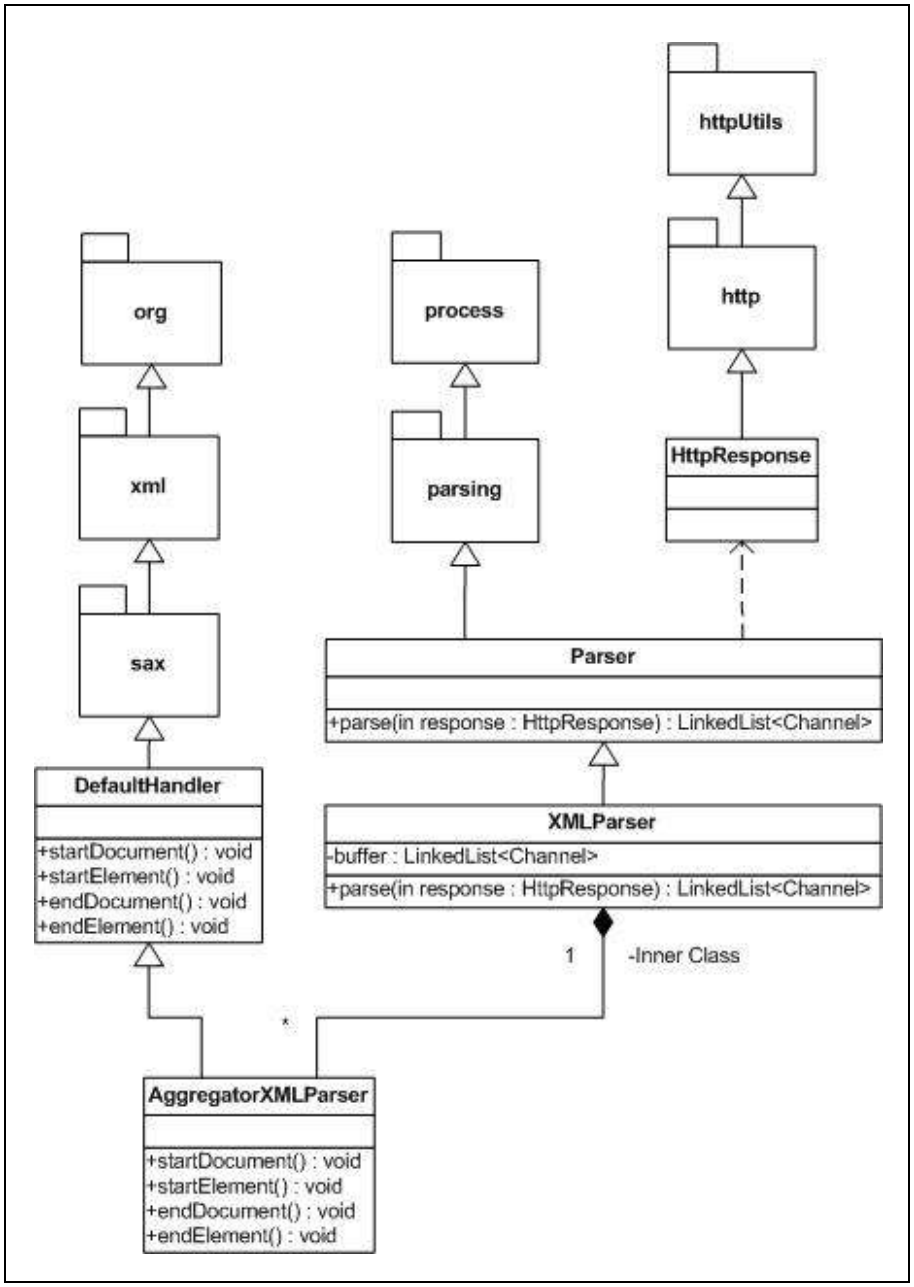

Figura 5. 18 - Diagrama de Clases de process.parsing.XMLParser

La Figura 5.19 muestra el Diagrama de Clases UML de la clase process.parsing.RDFParser junto con el diagrama de su clase padre: process.parsing.Parser.

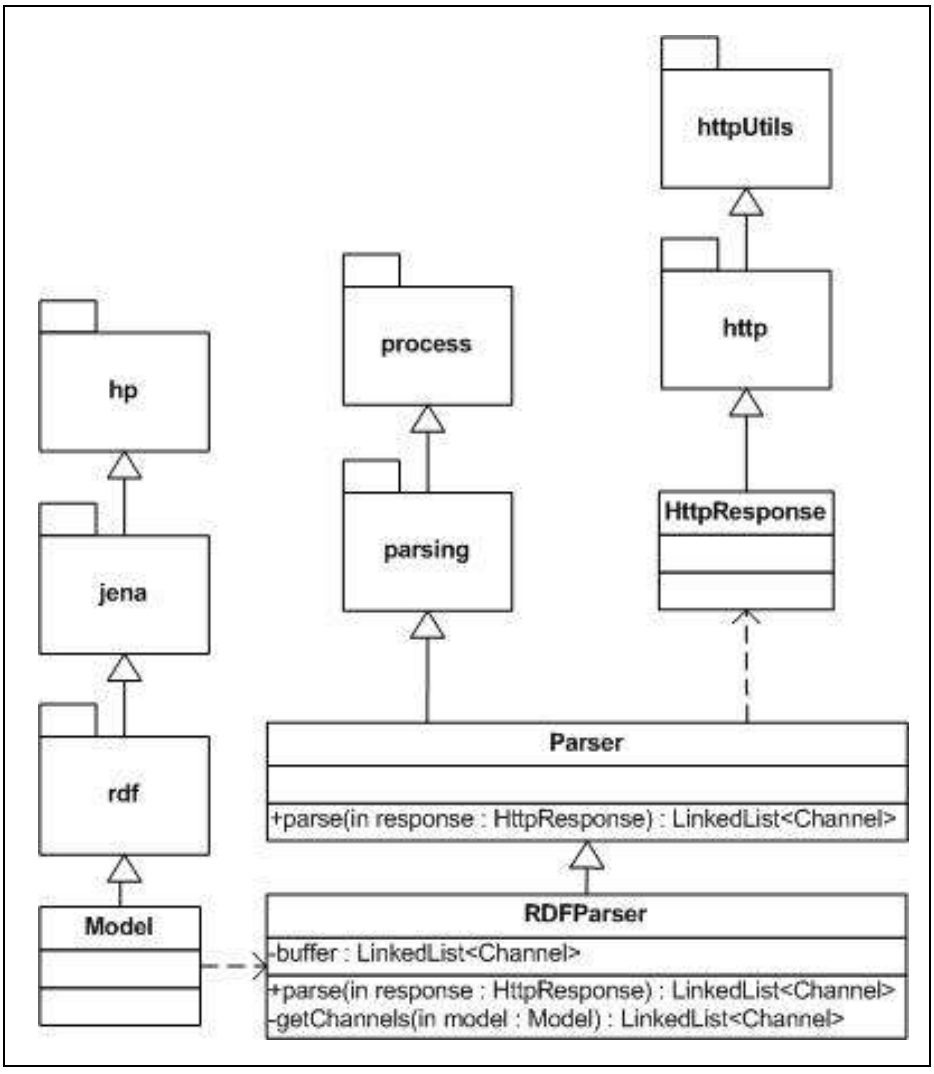

Figura 5. 19 - Diagrama de Clases de process.parsing.RDFParser

La Figura 5.20 muestra el Diagrama de Clases UML de la clase de excepción process.parsing.ParserException.

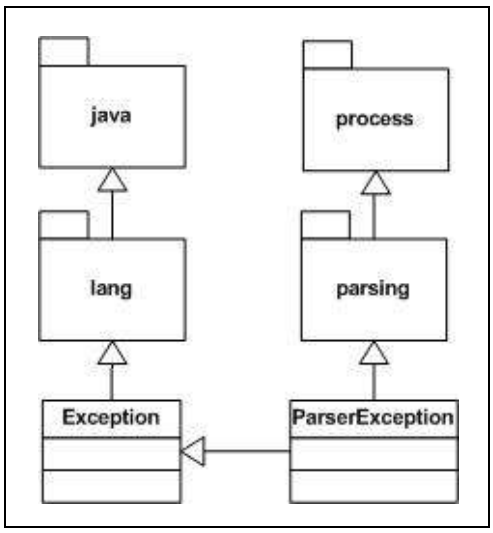

Figura 5. 20 - Diagrama de Clases de process.parsing.ParserException

La Figura 5.21 muestra el Diagrama de Clases UML de la clase de excepción process.parsing.RSSParser.

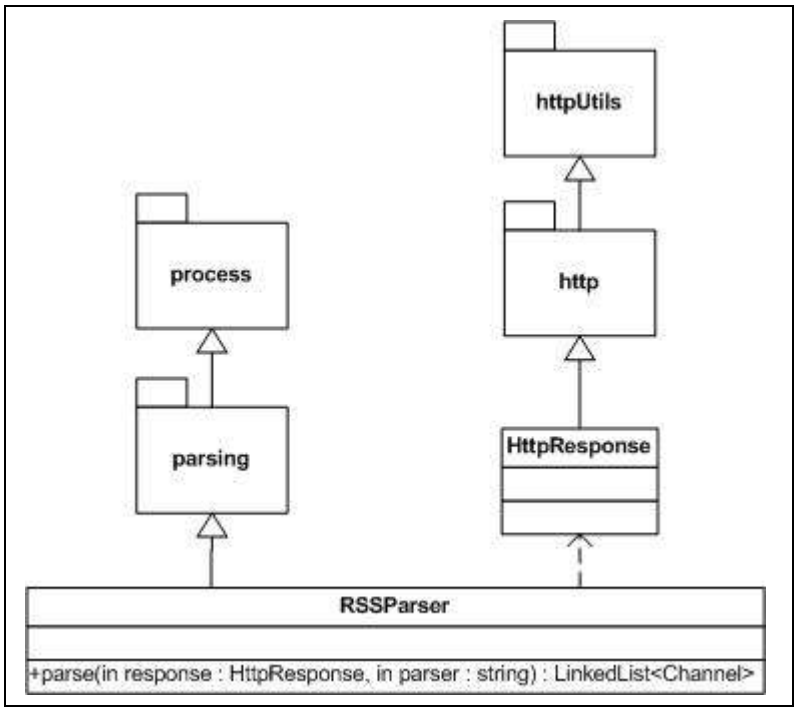

Figura 5. 21 - Diagrama de Clases de process.parsing.RSSParser

Paquete process.searching

La Figura 5.22 muestra el Diagrama de Clases UML de la única clase que conforma el Paquete process.searching: process.searching.SearchRSS.

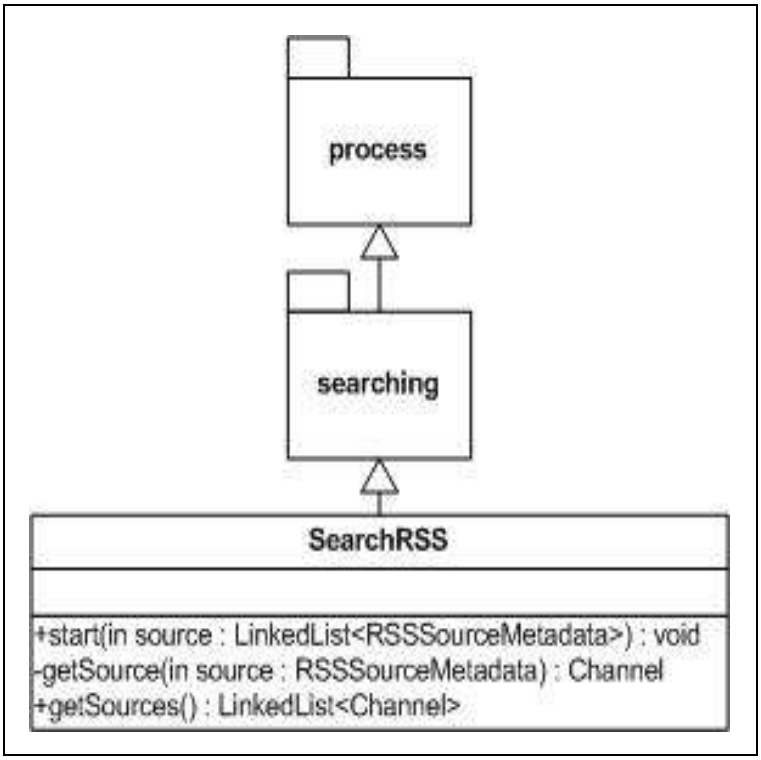

Figura 5. 22 - Diagrama de Clases de process.searching.SearchRSS

Paquete process.sorting

La Figura 5.23 muestra el Diagrama de Clases UML de la clase process.sorting.TreeSetChannel.

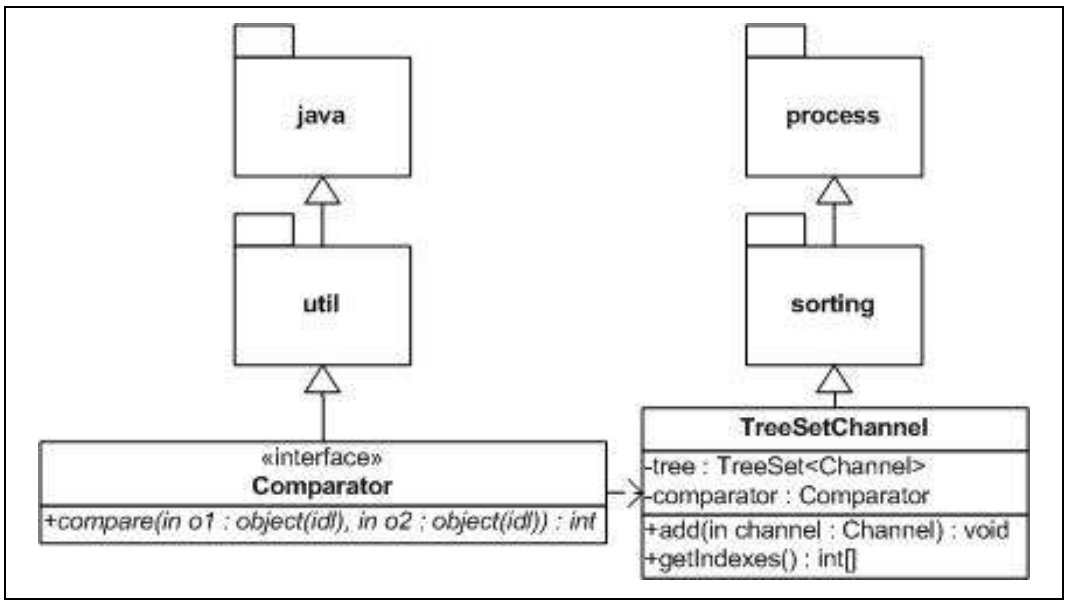

Figura 5. 23 - Diagrama de Clases de process.sorting.TreeSetChannel

La Figura 5.23 muestra el Diagrama de Clases UML de la clase process.sorting.TreeSetChannel.

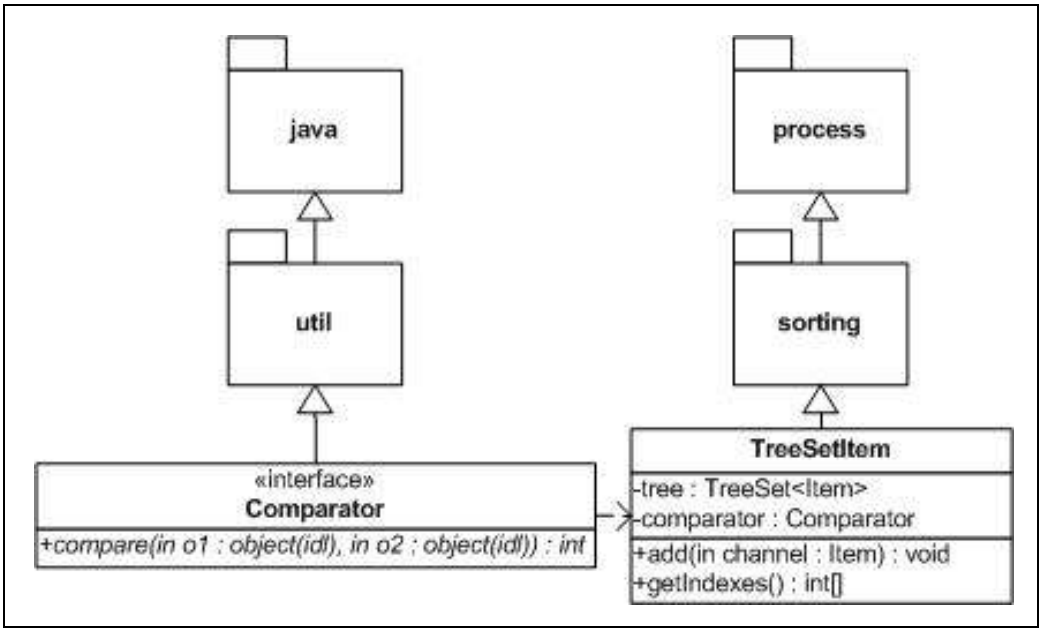

Figura 5. 24 - Diagrama de Clases de process.sorting.TreeSetItem

Conclusiones

Aquí termina el proceso de Diseño del Sistema que constituye uno de los pasos más grandes en la programación de una aplicación.

Lo que el Diseño nos dejó aquí fue un Sistema compuesto por 9 paquetes agrupados en dos categorías según su funcionalidad –definición de elementos y procesamiento– los cuales alojan más de 30 clases en las que se preservan los principios de Diseño en Orientación a Objetos:

- *Alta Primitividad*. Resaltada por las clases básicas del paquete rss.
- *Alta Cohesión*. Obtenida al seguir una clasificación por funcionalidad que permite que los elementos internos de cada clase se relacionen fuertemente entre sí.
- *Bajo Acoplamiento*. Al minimizar la interacción entre clases reduciendo la dependencia entre paquetes y objetos.

CONCLUSIONES

Resultados

Como parte de la Introducción de este documento se habló de los objetivos que debería de cumplir *Agregador Web BiDi*, una vez que se desarrolló la construcción de la aplicación se comprueba el cumplimiento de los objetivos propuestos resaltando sus **bondades**.

Objetivo: Construir un Sistema que permita desplegar un cuadro de noticias en la Página Web de la Biblioteca Digital.

Para mostrar como el Sistema cumplió con este objetivo –el principal objetivo–, la Figura 6.1 muestra la Página Web de la Biblioteca Digital con la inclusión del cuadro de noticias generado por *Agregador Web BiDi.*

Figura 6. 1 - Portal de la Biblioteca Digital de la UNAM

Objetivo: Que el cuadro de noticias despliegue toda la información necesaria de cada elemento que se muestre.

Como se muestra en la Figura 6.2 (enfoque al cuadro de noticias de la Figura 6.1) y como se puede apreciar en el Diseño del Sistema, cada entrada del cuadro de noticias cuenta con un encabezado o título, un breve resumen y una liga (contenida en el encabezado del elemento) que permite que el usuario consulte la noticia completa desde su ubicación original.

Figura 6. 2 - Módulo de Contenido Sindicado generado por Agregador Web BiDi

Objetivo: Que el Sistema sea sencillo de usar.

La forma en la que el usuario interactúa con la aplicación es mediante el archivo de configuración que controla, entre otras cosas, las fuentes de las que se desea obtener información. A continuación se muestra como ejemplo el archivo de configuración usado para generar el cuadro de noticias mostrado en las Figuras 6.1 y 6.2, este ejemplo deja ver que la construcción del archivo de configuración es clara y sencilla lo que permite poner el aplicativo en función rápidamente.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<config>

<sources>
    <source>
        <name>UNAM - Canal Universitario</name>
        <host>orion.dgsca.unam.mx</host>
        <path>/~rss/info.xml</path>
        <port>80</port>
        <parser>rss-2.0</parser>
        <timeout>20000</timeout>
        <encoding>UTF-8</encoding>
    </source>
    <source>
        <name>BBC</name>
        <host>newsrss.bbc.co.uk</host>
        <path>/rss/spanish/news/rss.xml</path>
        <port>80</port>
        <parser>rss-2.0</parser>
        <timeout>20000</timeout>
        <encoding>ISO-8859-1</encoding>
    </source>
    <source>
        <name>Blackwell Synergy</name>
        <host>www.blackwell-synergy.com</host>
        <path>/action/showFeed</path>
        <port>80</port>
        <parser>rss-2.0</parser>
        <timeout>20000</timeout>
        <encoding>UTF-8</encoding>
        <refresh>Mensual</refresh>
    </source>
    <source>
        <name>Biblioteca Perú</name>
        <host>bibliosperu.com</host>
        <path>/willana/?feed=rss2</path>
        <port>80</port>
        <parser>rss-2.0</parser>
        <timeout>20000</timeout>
        <encoding>UTF-8</encoding>
    </source>
    <source>
        <name>Knowledge Manager</name>
        <host>www.mapasconceptuales.info</host>
```

```
        <path>/rss/KnowledgeManager-esp.rss</path>
        <port>80</port>
        <parser>rdf</parser>
        <timeout>20000</timeout>
        <encoding>UTF-8</encoding>
    </source>
    <source>
        <name>Digital Library</name>
        <host>www.dlib.org</host>
        <path>/rss/dlib.rss</path>
        <port>80</port>
        <parser>rdf</parser>
        <timeout>20000</timeout>
        <encoding>UTF-8</encoding>
    </source>
    <source>
        <name>Nature Issues</name>
        <host>www.nature.com</host>
        <path>/nature/current_issue/rss/index.html</path>
        <port>80</port>
        <parser>rdf</parser>
        <timeout>20000</timeout>
        <encoding>UTF-8</encoding>
    </source>
    <source>
        <name>BiDi</name>
        <host>132.248.9.9</host>
        <path>/RSSAdmin/bidi.rss</path>
        <port>80</port>
        <parser>rss-2.0</parser>
        <timeout>20000</timeout>
        <encoding>ISO-8859-1</encoding>
    </source>
    <source>
        <name>Noticias de la Ciencia y Tecnologia</name>
        <host>feeds.feedburner.com</host>
        <path>/NoticiasDeLaCienciaYLaTecnologia</path>
        <port>80</port>
        <parser>rss-2.0</parser>
        <timeout>20000</timeout>
        <encoding>ISO-8859-1</encoding>
    </source>
    <source>
        <name>Tendencias 21</name>
        <host>www.tendencias21.net</host>
        <path>/xml/syndication.rss</path>
        <port>80</port>
        <parser>rss-2.0</parser>
        <timeout>20000</timeout>
        <encoding>ISO-8859-1</encoding>
    </source>
</sources>

<categories>
       <category default="false">
              <name>BiDi</name>
              <keyWord>Biblioteca Digital</keyWord>
              <keyWord>BiDi</keyWord>
       </category>
    <category default="false">
        <name>UNAM</name>
        <keyWord>UNAM</keyWord>
        <keyWord>Universitario</keyWord>
    </category>
    <category default="true">
        <name>Académica</name>
        <keyWord>Science</keyWord>
        <keyWord>Wall Street</keyWord>
```

```
    </category>
    <category default="false">
      <name>Periodísticas</name>
        <keyWord>Noticias</keyWord>
        <keyWord>Tendencias</keyWord>
        <keyWord>Conocimientos</keyWord>
    </category>
</categories>

</config>
```

Objetivo: Que el Sistema sea susceptible a actualizaciones.

Haciendo una revisión del Diseño del Sistema se pueden notar las siguientes cualidades de la implementación:

- Alta Primitividad. Se puede observar en las clases básicas del Sistema como las clases que definen los términos y elementos de RSS.
- Alta Cohesión. Como se detalla en el Diseño del Sistema, los métodos y atributos de cada clase están fuertemente relacionados entre sí de acuerdo a su funcionalidad esto le da una Alta Cohesión al Sistema haciéndolo propicio a actualizaciones.
- Bajo Acoplamiento. El Diseño no promueve la dependencia entre clases, la comunicación que se realiza entre cada una de las clases de control del Sistema se hace únicamente en los momentos indispensables y sin dejar de respetar la jerarquía de clases.

Estas tres cualidades permiten que el hacer modificaciones al Sistema sea fácil, sin tener que modificar las clases básicas y solamente agregando funcionalidad adicional a la ya existente.

Mejoras al Sistema

Como trabajos futuros para realizarse sobre la aplicación quedan aspectos como:

- Incorporación de nuevos estándares. Los estándares RSS se mantienen en constante evolución, por lo que ante la publicación de nuevos estándares

será necesario actualizar la aplicación para que ésta pueda seguir manejando todos los formatos usados en Internet para la difusión de Contenido Sindicado.

- Manejo eficiente de Contenido Multimedia. En el último año cada vez más fuentes de Contenido Sindicado se conforman por elementos Multimedia (como por ejemplo los *podcasts*). Ya que el Contenido Multimedia se caracteriza por distribuirse en archivos de gran tamaño la aplicación podría presentar problemas en un futuro para manejar muchas fuentes que contuvieran Contenido Multimedia. Una mejora necesaria y muy útil sería incluir un manejo más eficiente de la memoria que se utiliza para almacenar el Contenido Sindicado recopilado de las fuentes.

Reflexión

Este documento constituye mi Trabajo de término de la Licenciatura en Ciencias de la Computación. El objetivo de este trabajo fue mostrar una de las formas en las que he podido aplicar los muchos conocimientos adquiridos durante la carrera.

En particular este proyecto, *Agregador Web BiDi*, fue seleccionado para ser analizado por requerir de muchas áreas de las Ciencias de la Computación para su elaboración.

Se requirieron conocimientos de Algoritmos para elaborar los métodos de ordenamiento para los ítems y canales.

Se usó fuertemente la teoría de Estructuras de Datos para el almacenamiento y análisis de la información recopilada de las fuentes.

La recopilación de la información se programó de forma paralela haciendo uso de Multiprocesamiento para hacer más eficiente la tarea.

Como se puede ver claramente en este documento, se siguió la metodología de Ingeniería de Software de cuatro fases estudiada en la carrera durante este curso.

Para el Análisis y Diseño del Sistema se hizo uso en todo momento de Patrones de Diseño tal como se estudia en materias como Diseño y Programación Orientada a Objetos.

El resultado es, finalmente, una aplicación con fuertes cimientos científicos; que no solamente logra el objetivo planeado sino que lo hace de manera eficiente y correcta. Todo esto se debe a las bases que las Ciencias de la Computación me dieron y seguirán dando.

APÉNDICES

Apéndice A: Recolección de Información

La comunicación con cada una de las fuentes se procesa en un hilo de ejecución diferente. La ventaja que esta decisión de Diseño trae es el ahorro de tiempo en el proceso de recolección de información. Existe un periodo de tiempo en el que el servidor en el que se encuentre corriendo la aplicación *Agregador Web BiDi* solamente estará esperando por una respuesta de una fuente determinada; en este tiempo no se estará ejecutando ninguna tarea en particular por lo que resulta un buen ahorro de tiempo solapar la comunicación con todas las fuentes en un solo instante de tiempo.

Para ejecutar la recolección de información en hilos de ejecución separados se usa la clase HelpAggregator –interna de la clase rss.aggregator.Aggregator-. Se muestra a continuación el código del método getResults de la clase rss.aggregator.Aggregator.

```
private void getResults( ) {
        for(int i=0; i<sources.size(); i++) {
             new HelperAggregator(sources.get(i)).run();
        }
}
```

La clase HelpAggregator extiende la clase java.util.Thread que permite definir hilos de ejecución. El método run de la clase java.util.Thread contiene la funcionalidad que el hilo de ejecución procesará. El método run de la clase HelpAggregator se encuentra programado como sigue:

```
public void run() throws ParserException, SearcherException{
       SearchRSS s = new SearchRSS();
       s.start(new SearcherArgumentsCollection(source));
       buffer.add(s.getResources());
}
```

Apéndice B: Peticiones *http* y Búsqueda de Información

La clase process.searching.SearchRSS es la que se encarga de establecer la comunicación con el servidor que contiene la fuente para poder obtener la información buscada.

El método init de la clase process.searching.SearchRSS es el método más representativo dentro del proceso de comunicación.

```
private void init() throws SearchException, IOException {

      CookiesContext cookies = new CookiesContext();
      HttpRequest request = new HttpRequest(path, HttpRequest.GET);
      request.addHeaderField("Host", host + ":" + port);
      HttpResponse  response  =  HttpHelper.getResponse(host,  port,
                                 request,     cookies,     timeout,
                                 encoding);
      getResources(response);

}
```

Apéndice C: Análisis de la Información

El paquete process.parsing contiene las clases encargadas de analizar los documentos obtenidos de las fuentes y que almacenan el Contenido Sindicado.

La clase process.parsing.RSSParser se encarga de determinar que analizador en particular deberá de analizar la fuente determinada. Esta decisión se toma en el siguiente método:

```
public LinkedList<Channel> parse(HttpResponse res, String parser)
throws ParserException {

      if(parser.equals(PARSER_RDF)) {
            return new RDFParser().parse(res);
      } else if(parser.equals(PARSER_RSS_2)) {
            return new XMLParser().parse(res);
      }
      return new LinkedList<Channel>();

}
```

En caso de que el archivo a analizar sea un archivo RSS 0.91 o RSS 0.92 la clase que se encargará de terminar de analizar la fuente será la clase rss.parsing.RDFParser. En caso de que el archivo a analizar sea un archivo RSS 1.0 o un archivo RSS 2.0 la clase que se encargará de terminar el proceso de análisis será rss.parsing.XMLParser.

La clase rss.parsing.RDFParser hace uso de una biblioteca llamada *Jena*. Esta biblioteca ofrece herramientas para trabajar con contenido RDF simplificando la tarea de análisis dentro de la clase rss.parsing.RDFParser al siguiente método:

```
public LinkedList<Channel> parse(HttpResponse res) throws
ParserException {

        Model model = ModelFactory.createDefaultModel();
        model.read(res.getContent());
        return getChannels(model);

}
```

Las clases Model y ModelFactory del paquete *Jena* se encargan de analizar el archivo RDF. Finalmente el método getChannels de la clase rss.parsing.RDFParser analiza el modelo resultante y lo convierte en objetos.

Por otro lado, la clase rss.parsing.XMLParser usa las herramientas que el paquete *Sax* ofrece para analizar documentos XML. Mediante el uso de esta biblioteca el código del método parse de la clase rss.parsing.XMLParser está escrito como sigue:

```
public LinkedList<Channel> parse(HttpResponse res) throws
IOException, SAXException {

      XMLReader parser =
                XMLReaderFactory.createXMLReader("SAXParser");
      parser.setContentHandler(new ResultsLoader());
      parser.parse(new InputSource(
                HttpHelper.getContentFromResponse(res)));
      return buffer;

}
```

BIBLIOGRAFÍA

[1] *A Generalized Approach to Document Markup.*
Charles F. Goldfarb.
SIGPLAN Notices, 1986.
Annex A of ISO 8879.

[2] *A Software Infrastructure for RSS Deployment and Linking on the Web.*
José Antonio Camacho-Guerrero y Alessandra Alaniz Macedo.
Universidad de Sao Paulo. Publicaciones Sistemas de Informática.
Ribeirao Preto, Brazil, 2005.
ISSN: 1467-9087

[3] *Bitácoras y sindicación de contenidos: dos herramientas para difundir información.*
Jorge Franganillo y Marcos Antonio Catalán.
Universidad de Barcelona, Enero 2005.
Textos universitarios de biblioteconomía y documentación.
ISSN: 1575-5886

[4] *The Evolution of RSS.*
Andrew King.
Web Reference Articles, June 2006.

[5] *Information retrival on the Web.*
Mei Kobayashi and Koichi Takeda.
IBM Research. Scientific publications.
ACM Computing Surveys, Vol. 32, No. 2, June 2000.

[6] *Sharing headlines and information using XML. Content Syndication with RSS.*
Ben Hammersley.
O'Reilly publications©.
Primera edición, Marzo 2003.
ISBN 10: 0-596-00383-8 | ISBN 13: 9780596003838

[7] *RSS 0.91 Oficial Specification, revision 3*
Dan Libby
Netscape Communications
Revision III July, 1999

[8] *RSS 0.92 Oficial Specification, revision 1*
Dan Libby
UserLand Software ©
Diciembre 2000

[9] *RSS 1.0 Oficial Specification*
Gabe Beged-Dov, Dan Brickley, et all.
RSS-DEV Working Group
Diciembre 2000

[10] *RSS 2.0 Oficial Specification*
Dan Libby
UserLand Software ©
Agosto 2002