# DESARROLLO DE UN WEB SERVICE PARA EL PROCESAMIENTO DE DATOS GEOGRÁFICOS

**NICOLÁS ANDRÉS CERÓN OROZCO**
**JOSÉ ARMANDO ORDÓÑEZ CORDOBA**

**Monografía presentada como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones.**

**Director: Ing. JUAN CARLOS CORRALES**

**UNIVERSIDAD DEL CAUCA**
**FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES**
**DEPARTAMENTO DE TELEMATICA**
**POPAYÁN**
**2003**

# LISTA DE FIGURAS

# LISTA DE TABLAS

# TABLA DE CONTENIDO

# CAPITULO 1. INTRODUCCIÓN

En la actualidad, el mundo entero está concentrando paulatinamente sus esfuerzos hacia la gestión eficiente de cada uno de los recursos con que cuenta, tanto en empresas privadas como en entidades gubernamentales. Para tal efecto es supremamente importante contar con la información veraz y oportuna para la toma de decisiones.

Sin embargo, en muchas ocasiones los mecanismos de planificación requieren más que la simple información tabular, y las graficas que de allí puedan obtenerse; por ejemplo, para un planificador el hecho de observar cifras de la población para la asignación de recursos entre divisiones administrativas en una tabla, no le permite gestionar tan eficientemente como otro que puede ver un mapa con las mismas divisiones y por medio de colores o etiquetas puede analizar en que sector de la ciudad o el país se concentran las poblaciones.

Otro ejemplo más diciente se presenta en el campo de la prevención epidemiológica. Los registros de los casos de una enfermedad en diferentes sectores de la ciudad, serán poco comunicativos en comparación con unos círculos rojos en el mapa de la región que señalan la alerta máxima, cuando el umbral de casos que se presentan en un centro de salud sobrepasa su limite.

Los sistemas de información geográficos son una tecnología que propende por la integración de la los datos geográficos y temáticos que llevan implícitos para conseguir una herramienta poderosa de gestión, planificación y administración de los recursos de las instituciones, los cuales dependen de la naturaleza de estas últimas y entre los cuales podemos encontrar bienes, recursos naturales o personas.

No cabe duda de la importancia de los sistemas de información geográfica y de su utilidad, sin embargo el trabajo no termina allí, pues al interior de las instituciones, las diferentes divisiones pueden tener información que es necesaria para su trabajo, y que aparece de manera redundante en otras. Por otro lado, una división puede alojar información que aunque puede ser útil para las demás, no puede ser aprovechada por

el desconocimiento de su existencia o por la complejidad del proceso de integración con el sistema de información de la división que la requiere.

Un reto aún más grande es ofrecido actualmente por la dinámica empresarial y gerencial, que nos presenta escenarios de fusión, integración o simplemente tratados de cooperación entre los entes institucionales lo que obliga a la compartición mutua de parte de la información incluyendo los datos geóreferenciados (información a la cual se ha asignado un identificador geográfico para su ubicación cartográfica).

Como es de esperarse, cada organización maneja interiormente un sistema de información, cuyos datos obedecen a un modelo de datos específico y a un Sistema de Gestión de bases de datos para un propósito definido, es decir manejan un formato, una semántica y una sintaxis propia. Si además dicha información aloja un componente geográfico, elementos como el formato de los mapas, el tipo de coordenadas o el software de gestión de la información geográfica propia, complican aún más el trabajo de integración con otros sistemas igualmente complejos.

Para resaltar la importancia de dicha integración, analicemos el ejemplo de las entidades estatales en Colombia que gestionan los recursos naturales, algunas instituciones como parques Naturales requieren de información detallada de los estudios realizados sobre los suelos y las aguas de una cuenca para analizar su evolución o deterioro, sin embargo ese mismo estudio puede ser muy útil para el análisis geológico que Ingeominas realiza dentro del plan de vigilancia de la actividad volcánica de un cráter que se encuentre cerca de la región, siendo a su vez estos datos imprescindibles para la planificación de cultivos o ganadería en la región por parte del gobernante o la administración del territorio.

Hoy en día el proceso de intercambio de información entre las entidades anteriores, se constituye como un ejercicio realmente complejo que involucra trámites burocráticos y desplazamientos, además de los tediosos procesos de conversión de formatos y comprensión de la semántica utilizada por los "dueños" de la información. En un proceso tan largo y tedioso, más que la inherente demora, lo más preocupante es la aparición de errores y la falta de apreciación de los datos en los mapas de manera óptima.

Apreciando escenarios como el anterior, entendemos que la integración aunque difícil es definitivamente necesaria y los frentes para atacar el problema son básicamente dos, el primero se relaciona con el rediseño y traslado de uno de los sistemas al formato y semántica del otro, mientras que el segundo se orienta a la búsqueda de un lenguaje común que sirva de puente entre las fuentes de datos.

El primero implica una altísima inversión monetaria, ya que sería echar al traste con los sistemas que regularmente utilizaban las empresas fusionadas y que los empleados y clientes conocen, además del asunto de las licencias y los requerimientos del sistema; mientras que la segunda requiere una gran inversión en tiempo y dedicación de personal para poner a punto un intercambio dinámico de información. El mercado y la mayoría de las empresas se inclinan por la segunda opción, pues además prevén la integración a futuro de nuevas fuentes de datos con diferentes características.

Al tener claro el papel que debe jugar un sistema de integración, es imprescindible que los mecanismos de transporte entre el mediador (componente encargado de la gestión del proceso de integración) y las fuentes de datos sean lo más accesibles que se pueda, por tal razón el corazón del intercambio de información debe estar poco acoplada con la tecnología de transporte y a su vez ésta última debe ofrecer las mejores posibilidades de acceso.

Entre las tecnologías de computación distribuidas que son propicias para sistemas de integración, los Servicios Web sobresalen por las características de sus protocolos base, XML y HTTP que hacen de ellos la mejor opción cuando de intercambio de datos de forma transparente se trata.

# CAPITULO 2. MARCO CONCEPTUAL

## 2.1 SISTEMAS DE INFORMACIÓN GEOGRÁFICA (SIG)

### 2.1.1 Definición

De manera general podemos definir un sistema de información geográfica como un sistema de información utilizado para representar y analizar las características geográficas presentes en la superficie de la Tierra y los eventos (atributos no espaciales enlazados a la geografía estudiada) que tienen lugar en ella. El objetivo de representar digitalmente estos aspectos es convertirlos de una forma análoga a una digital cuyo estudio cuantitativo sea más sencillo.

La tecnología SIG integra operaciones comunes de las bases de datos tales como solicitudes y análisis estadístico con los beneficios de visualización y análisis geográfico que solo pueden ofrecer los mapas. Estas capacidades y el hecho de que casi el 70% de los datos en el mundo tienen referencia geográfica, hacen el papel de los SIG muy importante como herramienta para la toma de decisiones y la gestión de datos referenciados geográficamente.

El hecho de que cualquier objeto presente en la Tierra puede ser georeferenciado (atributo o elemento al cual se le ha adicionado un identificador geográfico que le permite ser localizado en un mapa), hace posible la asociación de todas las base de datos con un SIG, entendiéndose por georeferenciado, un objeto que se puede definir inequívocamente por medio de un sistema referencial de coordenadas (sistema que permite representar inequívocamente los elementos georeferenciados)[GID02].

### 2.1.2 Ventajas de un SIG

El uso de los SIG se ha ido incrementando así como su valoración debido a las siguientes ventajas:

- **Contribuye en la planeación de proyectos:** La buena planeación es siempre importante. La creación de mapas temáticos de los diferentes factores que pueden incidir en el transcurso de cualquier proyecto, ayudando a notar su

influencia fácilmente, por ejemplo, para la construcción o compra de una granja, los mapas de composición del suelo, vegetación y topografía pueden dar puntos de partida muy útiles y sólidos[WIS00]

- **Ayuda en la toma de decisiones:** Siempre la mejor información contribuye a mejores decisiones, aunque el SIG no es un sistema de toma de decisiones automático, la tecnología puede usarse para la presentación de la información de manera que esta labor se facilite ampliamente.

- **Análisis Visual:** Usando el modelo digital de terreno (utilidad importante de los SIG) , el paisaje puede visualizarse mejor, muchos cálculos como volumen de erosión del suelo pueden hacerse más fácil, mientras que otros datos de interés como el estudio del impacto ambiental y el estudio de la distribución son interpretables cómodamente por los administradores.

### 2.1.3 Funcionamiento

Un SIG almacena información como una colección de niveles temáticos relacionados por geografía como se muestra en la figura 2.1.


**FIGURA 2.1. CAPAS DE UN SIG**

La información geográfica contiene ya sea una referencia geográfica explícita tal como latitud y longitud, una coordenada de un sistema nacional, o una referencia implícita tal como domicilio, código postal o nombre de calle.

Los datos del SIG (en cada una de sus capas), está compuesta de características geográficas, por ejemplo una calle se representa por una línea, un barrio por un polígono, y una casa por un punto [TUG03], cada elemento geográfico tiene una tabla asociada (Figura 2.2), por ejemplo en la gráfica cada escuela se representa por un punto:


**FIGURA 2.2. ASIGNACIÓN DE ATRIBUTOS**

La funcionalidad de los SIG se puede dividir en:

**Adquisición e ingreso de datos**: Los datos provenientes de imágenes satelitales, fotos aéreas o mapas se digitalizan y convierten a un formato compatible del SIG para su análisis.

**Almacenamiento y gestión de los datos**: Los datos se guardan en una base de datos que se encarga de la seguridad e integridad de los datos.

**Manipulación y análisis**: El análisis geográfico requiere de una asociación entre los datos de la base de datos y la información espacial. El SIG provee herramientas para tal fin de manera que los atributos de una característica geográfica de un mapa (punto, línea, etc) pueda almacenarse en una base de datos relacional, las consultas posteriores generan un mapa mostrando la información obtenida de la base de datos y la generación dinámica del mapa de acuerdo a los resultados. De igual manera las consultas se pueden hacer orientadas a la geografía o a los datos.

**Presentación de los datos**: los datos puede ser mostrados en la interfaz de usuario del programa SIG o en papel si se imprimen los mapas consultados.

### 2.1.4 Formato de los Mapas Digitales

El término formato de archivo se refiere a la estructura lógica usada para guardar la información en un archivo SIG. Casi todos los SIG tienen su propio formato, estos formatos son diseñados para su uso óptimo dentro del software y son usualmente propietarios, por lo que en ocasiones es necesario hacer conversiones entre formatos.

#### 2.1.4.1 Formatos vectoriales

Muchas aplicaciones SIG están basados en tecnología vectorial, estos son los más comunes y los más complejos ya que hay muchas formas de guardar las coordenadas, enlaces a los atributos, estructura de las bases de datos e información de despliegue[GID03]. Algunos formatos son:

| Nombre del Formato | Software | Desarrollador | Comentario |
|---|---|---|---|
| Arc Export | ARC/INFO* | Environmental Systems Research Institute, Inc. (ESRI) | Permite la transferencia de datos entre plataformas |
| ARC/INFO | | | |
| AutoCAD Drawing Files (DWG) | AutoCAD* | Autodesk | |
| Autodesk Data Interchange File (DXF™) | Varios | Autodesk | Widely used graphics transfer standard. |
| Hewlett-Packard Graphic Language (HPGL) | Varios | Hewlett-Packard | Usado en Plotters HP |
| MapInfo Map Files | MapInfo* | MapInfo Corp. | |
| MicroStation Design Files (DGN) | MicroStation* | Bentley Systems, Inc. | |

**TABLA 2.1. FORMATOS VECTORIALES DEL MERCADO**

#### 2.1.4.2 Formato Raster

Los archivos raster generalmente son usados para almacenar información, como mapas escaneados o imágenes aéreas, también se usan para captura de datos por satélite y otros sistemas; algunos formatos son:

| Nombre Formato | Plataforma | Desarrollador | Comentario |
|---|---|---|---|
| Arc Digitized Raster Graphics (ADRG) | Military mapping systems | Agencia Defensa USA | |
| Band Interleaved by Pixel (BIP) | Varios | Estándar para recepción remota | |
| Band Sequential (BSQ) | Varios | Estándar para recepción remota | |
| Digital Elevation Model for (DEM) | Varios | United States Geological Survey (USGS) | Estándar para modelos digitales del terreno. |
| PC Paintbrush Exchange (PCX) | PC Paintbrush | Zsoft | Widely used raster format. |
| Tagged Image File Format (TIFF) | PageMaker | Aldus | Formato ampliamente usado. |

**TABLA 2.2. FORMATOS RASTER DEL MERCADO**

### 2.1.5 Estándares de Información Geográfica

Hoy en día, existe diversa información geoespacial disponible tanto en la web como en archivos off-line, pero es heterogénea e incompatible. Se requiere de mucho trabajo y sistemas especiales para combinar diferentes mapas acerca de la misma región. La conversión de datos consume tiempo, y los resultados no son siempre los mejores, la creación de interfaces comunes es la mejor aproximación a una futura solución.

Uno de los estándares más importantes es el de OGC (Open Gis Consortium). Ésta iniciativa busca a pesar de las diferencias entre los sistemas SIG, la creación de una estructura formal para conseguir interfaces comunes de representación, para tal fin ha reunido a las empresas lideres del mercado:

- ESRI (Enviromental Systems Research Institute,Inc.)
- IBM Corporation
- Informix Software,Inc

- MapInfo Corporation

OGC busca el acceso transparente a geodatos heterogéneos y el geoprocesamiento de recursos en un ambiente de redes. [OGC03]

Dentro de la especificación, OGC adoptó GML, un lenguaje XML desarrollado para el manejo de datos geográficos, el cual puede entregar la información geográfica en forma de características, y controlar como deben ser mostradas dichas características, de esta manera y con la ayuda de un plug_in cualquier browser puede convertirse en un Cliente SIG.

En aras de conseguir una plataforma de integración de SIGs realmente abierta, es primordial la utilización de un formato abierto y extensible para la representación de la información geográfica, el cual debe estar consolidado realmente como un estándar en el mercado para el intercambio de información del sistema.

Se decidió utilizar para nuestro sistema el formato de shapefiles de ESRI por las siguientes razones:

- Este formato creado por ESRI es público, ya que ESRI lo liberó en 1998. Por esta razón todos los servidores de mapas acceden directamente al formato shapefile. [ESR03]
- Los formatos shapefiles están planeados para crecer y en un futuro permitirán a un archivo principal tener diferentes shapes y no solamente uno como contiene actualmente.
- Es el formato más utilizado al interior de los sistemas de información geográficos que se manejan en la región lo cual hace que la integración de éstos con la plataforma de integración sea mas directa.

### 2.1.6 Formato De Un Shapefile

Los shapefiles son formatos creados por ESRI para almacenar por un lado datos geométricos y por otro atributos de la información espacial. La geometría de un objeto es almacenada como un "shape" el cual comprende un conjunto de coordenadas vectoriales. [EDO02]

Los "shapefiles" soportan los tipos básicos de elementos, es decir el punto, la línea y el polígono. Los atributos descriptivos son almacenados en un archivo dBase. Cada atributo tiene una relación de uno a uno con un registro de un shape asociado.

### 2.1.6.1 Tipos de Shape

Los "shape" son las geometrías que puede tener un shapefile. Cada shapefile estará asociado a un objeto geométrico, el cual puede ser cualquiera de los valores que se muestran en la tabla 2.3. El campo "Valor", es el valor con que es representada la geometría cuando se almacena en un archivo "shape".

| Valor | Shape Tipo |
|---|---|
| 0 | Shape Nulo |
| 2 | Punto |
| 3 | PolilInea |
| 5 | Polígono |
| 8 | Multipunto |
| 11 | PuntoZ |
| 13 | PoliLineaZ |
| 15 | PologonoZ |
| 18 | MultipuntoZ |
| 21 | PuntoM |
| 23 | PoliLineaM |
| 25 | PoligonoM |
| 28 | MultiPuntoM |
| 31 | MultiPatch |

**TABLA 2.3. TIPOS DE SHAPES**

Los tipos de "shape" que no se encuentran en la lista, están reservados para usos futuros.

### 2.1.6.2 Elementos de un Shapefile

Un Shapefile de ESRI consta de un archivo principal, un archivo de índices y una tabla de dBase. En el archivo principal se tiene acceso directo a los registros de longitud variable donde cada registro es un "shape" o un objeto con una lista de sus vértices. En el archivo de índices cada registro es un apuntador al registro correspondiente en

el archivo principal. La tabla de dBase almacena los atributos de los objetos con un registro por objeto.

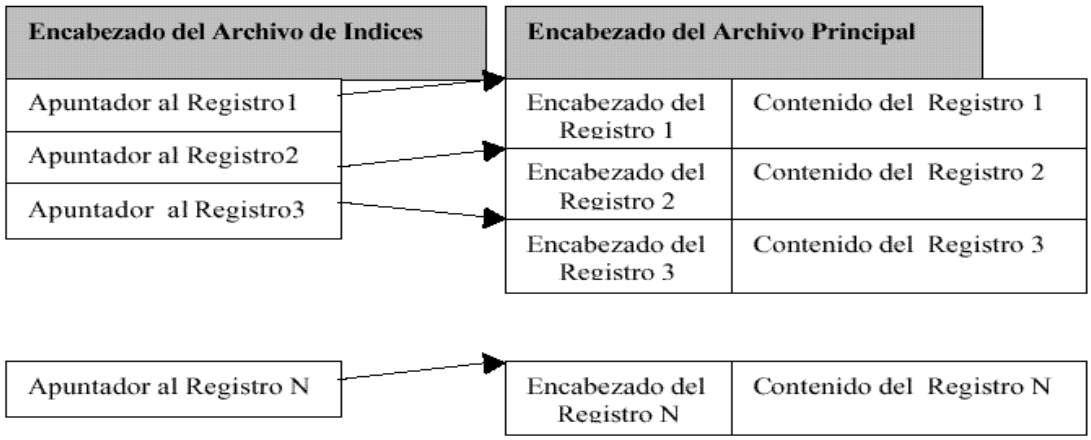



**FIGURA 2.3. RELACIÓN ENTRE ARCHIVOS PRINCIPAL Y DE ÍNDICES**

Todos los archivos siguen la convención de 8 caracteres de nombre más tres de extensión, el archivo principal, el archivo de índices y el archivo dBase tienen el mismo nombre. El nombre del archivo comienza con un carácter alfanumérico (a-X,0-9.-,_ ) seguido de cero a siete caracteres más. La extensión del archivo principal es ".shp", del archivo de índices es ".shx" y de la tabla de dBase es ".dbf".

El shapefile solamente maneja dos tipos de datos numéricos, entero(con signo de 32 bits) y doble (flotante con 64 bits de doble precisión con signo). En un shapefile se mezclan los dos tipos de ordenamiento de bytes LittleIndian y BigIndian.

### 2.1.6.3 Organización de los archivos shape

El archivo principal consta de un encabezado de archivo, encabezado de registro y el registro, el cual es de longitud variable según el tipo de shape. El encabezado de archivo es de un tamaño fijo de 100 bytes.

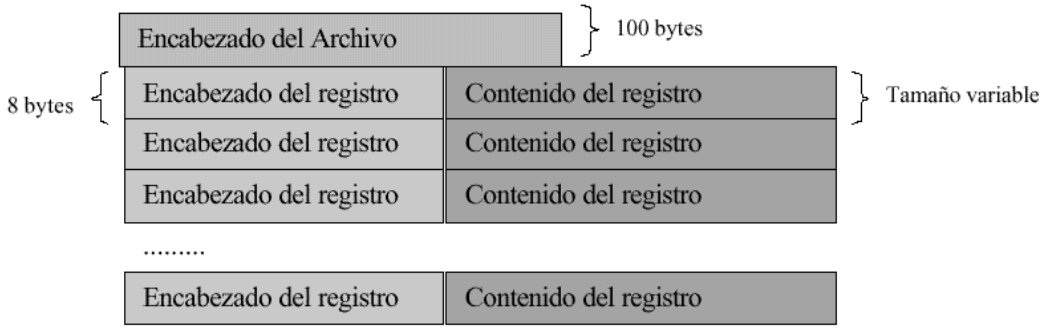



**FIGURA 2.4. ESTRUCTURA DEL ARCHIVO PRINCIPAL**

El campo posición de la tabla, es el byte del archivo donde se encuentra. En todos los archivos shape solamente hay dos tipos de datos numéricos, dobles o enteros, lo que varía entre ellos es el orden del byte más significativo, ya sea BigIndian o LittleIndian. Todos los campos descriptivos del encabezado del archivo principal, encabezado de registro y contenido del registro del archivo principal son de tipo LittleIndian, los demás números del archivo son de tipo BigIndian.

| Posición | Campo | Valor | Tipo | Orden Bytes |
|---|---|---|---|---|
| Byte 0 | Código archivo | 9994 | Entero | Big |
| Byte 4 | Sin uso | 0 | Entero | Big |
| Byte 8 | Sin uso | 0 | Entero | Big |
| Byte 12 | Sin uso | 0 | Entero | Big |
| Byte 16 | Sin uso | 0 | Entero | Big |
| Byte 20 | Sin uso | 0 | Entero | Big |
| Byte 24 | Longitud Archivo | Longitud del archivo | Entero | Big |
| Byte 28 | Versión | 1000 | Entero | Little |
| Byte 32 | Shape Tipo | 1000 | Entero | Little |
| Byte 36 | Boundig box | Tipo de shape | Doble | Little |
| Byte 44 | Boundig box | Xmin | Doble | Little |
| Byte 52 | Boundig box | Y min | Doble | Little |
| Byte 60 | Boundig box | X max | Doble | Little |
| Byte 68 | Boundig box | Y max | Doble | Little |
| Byte 76 | Boundig box | Zmin | Doble | Little |
| Byte 84 | Boundig box | Zmax | Doble | Little |
| Byte 92 | Boundig box | Mmin | Doble | Little |

**TABLA 2.4. CAMPOS DEL ENCABEZADO DEL ARCHIVO PRINCIPAL**

Los bytes 36 al 92 correspondientes al bounding box, representan el rectángulo que rodea una geometría determinada, es decir el espacio suficiente para contener toda la geometría, el boundig box está definido por Xmin, Ymin, Xmax, Ymax. El boundig box

del encabezado del archivo principal, es el rectángulo que contiene a todas las geometrías del archivo.

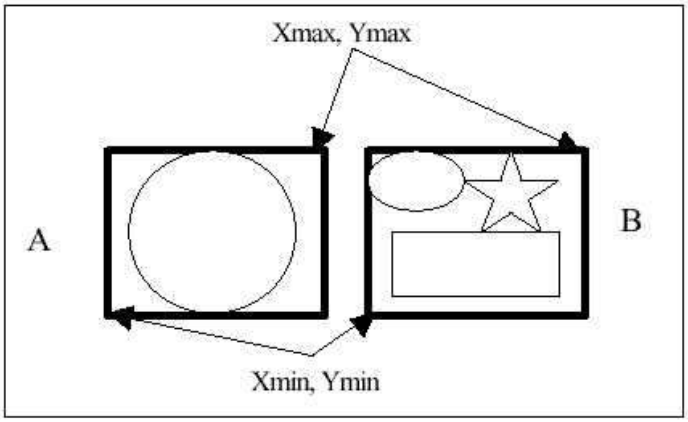



**FIGURA 2.5. BOUNDING BOX**

Actualmente los shapefiles sólo pueden almacenar un solo tipo de geometría, es decir, un archivo está compuesto sólo de shapes punto o líneas pero no de varias. El encabezado de cada registro almacena el número y tamaño de cada registro. Los encabezados de los registros tienen una longitud fija de 8 bytes. Se toma como byte 0 la posición donde comienza el encabezado de registro, el cual es de tipo entero y tiene un tamaño de 4 bytes. El campo de longitud del contenido es de tipo entero y nos indica el tamaño del registro "shape" . El tamaño del registro está dado en palabras de 2 bytes, es decir, cada registro contribuye con (4 + tamaño del registro) palabras de 2 bytes.

Dado que existen varios tipos de "shapes", el tamaño del contenido del registro depende del tipo de objeto geométrico. Es decir, los registros son de tamaño variable. Por ejemplo si el shape contiene valor nulo, nos indica que no existen datos geométricos en el shape. Un tipo nulo se emplea durante la creación de los shapefiles y se les agrega información después de ser creados.

Si el shape del archivo es de tipo punto se necesita especificar su posición a través de dos coordenadas X e Y.

A partir de los puntos se pueden generar multipuntos, estos se representan como un conjunto de puntos los cuales no están conectados en un orden específico. Otro shape importante lo constituye la PoliLinea, la cual es un conjunto ordenado de dos o más vértices. Cada par de vértices forma un segmento. El segmento puede o no interceptarse.

El último "shape" es el polígono, que consiste en uno o más anillos, un anillo es una serie conectada de cuatro o más puntos que se cierran y no existe intersección entre ellos.

#### 2.1.6.4. Organización del archivo de índices

El archivo de índices (.shx) contiene un encabezado de 100 bytes seguido por encabezados de 8 bytes de longitud fija.

El encabezado de este tipo de archivo tiene la misma estructura que el encabezado del archivo principal. Cada registro guarda la posición donde comienza el registro en el archivo principal (offset) y el tamaño del registro. Estos datos están dados en palabras de 2 bytes. El número de registros en el archivo de índices es el mismo número de registros en el archivo principal.

#### 2.1.6.5 Archivos dbf

Para almacenar la información descriptiva se utilizan los archivos dBase, estos archivos contienen las características de los atributos o las claves de las tablas a través de las cuales se pueden efectuar "joins". Este formato es de tipo estándar. Cualquier conjunto de campos presentados en la tabla deben de cumplir con los siguientes requisitos:

- El nombre del archivo debe ser el mismo del archivo principal y el de índices, con excepción de la extensión que debe ser .dbf.
- La tabla debe contener un registro por shape.
- En el encabezado el valor del año debe ser desde 1900.

### 2.1.6.6 Jerarquía de las clases de los shapefiles

Este esquema de clases se construye a partir de las relaciones y atributos de los elementos que conforman un proyecto de ArcView.

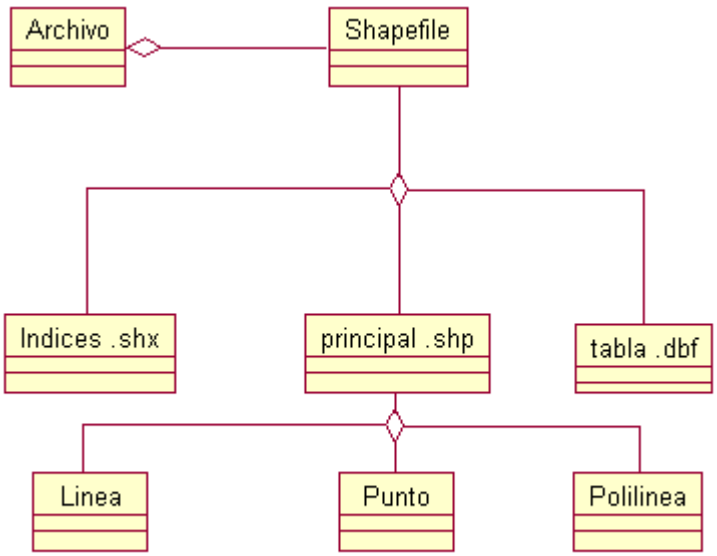



**FIGURA 2.6. JERARQUÍA DE CLASES DEL SHAPEFILE**

## 2.2 INTEGRACIÓN DE DATOS

### 2.2.1 Conceptos básicos

La integración de bases de datos no sólo es necesario en el ámbito científico, los procesos realizados en una gestión eficiente de la información son día a día fundamentales dentro de una empresa, y más aún cuando ésta presenta procesos de fusión o integración debido a la dinámica empresarial que mueve actualmente al mundo. Por lo regular cada organización posee un Sistema de Información que usa un modelo de datos específico y Sistemas de Gestión de Bases de Datos para un propósito bien definido; al presentarse una fusión empresarial bajo esta perspectiva,

implica asumir dos retos, el primero orientado hacia la construcción de una nueva estructura de datos para los requerimientos del nuevo sistema, y el segundo enfocado en la transferencia de datos entre los sistemas existentes. El primero implicaría una alta inversión monetaria, ya que sería echar al traste con los sistemas que regularmente utilizaban las empresas fusionadas; la segunda, implicaría una gran inversión en tiempo y dedicación de personal para poner a punto un intercambio dinámico de información. [FWI03]

Es claro que la opción dos es la más adecuada, pero abordarla implica tener en cuenta algunos problemas que pueden originarse debido a la heterogeneidad de datos. En general, la comunidad que trata los sistemas distribuidos de bases de datos tiene identificados dichos inconvenientes, los cuales son llamados problemas de heterogeneidad y pueden ser divididos en tres categorías:

- Sintaxis
- Estructura
- Semántica

**La sintaxis** hace referencia a la forma de las expresiones, tales como sentencias y unidades de programa. Entre los elementos sintácticos tenemos:

- Conjunto de caracteres
- Identificadores
- Símbolos de operadores
- Palabras claves y reservadas
- Sentencias

**La estructura** depende del modelo de datos seleccionado y del concepto de las diferentes representaciones.

**La semántica** determina el significado de las expresiones sintácticas tales como las sentencias y las unidades de programa.

Ejemplificando las tres categorías de heterogeneidad presentadas con respecto a una base de datos, en la sintaxis se define la heterogeneidad en los formatos de los datos, en la estructura se destacan los homónimos, sinónimos y diferentes atributos de la base de datos, y en la semántica se expresa el significado deseado de los términos en el contexto.

Para la gestión de información los problemas a nivel estructural y semántico tienen una consideración especial ya que terminológicamente son importantes; pues son éstas categorías las que tienen el conocimiento de la organización. En este sentido, un administrador del Sistema de Información de la Organización debe preocuparse por realizar un mapeo entre la terminología del Sistema de Información existente en su organización y el sistema de información de la entidad con la que ha sido fusionada. Una aproximación a la solución de este problema está basado en ontologías formales.

EL objetivo general de todos los esfuerzos para la integración, es permitir a los usuarios responder a una consulta con resultados de múltiples fuentes de información , para lo cual es necesario poder consolidar los datos consultados en una sola respuesta. Dos aspectos importantes para la consecución del objetivo son el uso de metamodelos y ontologías.

### 2.2.2 Ontologías basadas en la Integración de Información

Para alcanzar interoperabilidad semántica entre sistemas de información que usan variadas terminologías, el significado de la información que es intercambiada debe ser entendida por todos los sistemas participantes en el proceso. Los conflictos semánticos ocurren cuando dos contextos no usan la misma interpretación de la información. El uso de ontologías para la clarificación del conocimiento implícito en cada sistema es una alternativa que puede llegar a solucionar los problemas semánticos de heterogeneidad entre sistemas. [FWI03]

Casi todas las ontologías basadas en la integración son usadas para realizar una descripción explícita de la semántica de cada fuente de información que pretende ínter operar. En general son tres los posibles métodos usados en el ámbito de las ontologías, los cuales serán descritos a continuación :

#### 2.2.2.1 Ontología simple

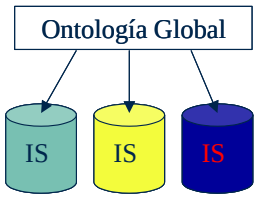



**FIGURA 2.7. MÉTODO DE ONTOLOGÍA GLOBAL[FWI03]**

La figura 2.7 muestra el uso de una ontología global que provee un vocabulario compartido por las diferentes fuentes de información con el objetivo de especificar la semántica de cada una de ellas. Un ejemplo de estas ontologías es SIMS [EDO02] .

#### 2.2.2.2 Ontología Múltiple

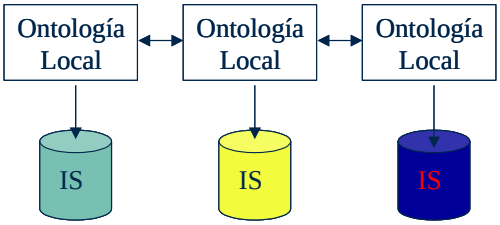



**FIGURA 2.8. MÉTODO DE ONTOLOGÍA MÚLTIPLE[FWI03]**

En esta ontología cada fuente de información es descrita por su propia ontología, una implementación de esta ontología es OBSERVER [MMI96] en el cual la semántica de cada fuente de información es descrita por separado como se muestra en la figura 2.8.

#### 2.2.2.3 Ontología Híbrida

En la figura 2.9, se muestra que este método fue desarrollado para superar las desventajas que se presentan en las Ontologías simple y múltiple. Al igual que la ontología múltiple la semántica de cada fuente es descrita por su misma ontología. Para lograr que cada fuente ontológica pueda ser comparada con las demás fuentes, éstas han sido construidas sobre un vocabulario que es compartido por todas [RRS97].

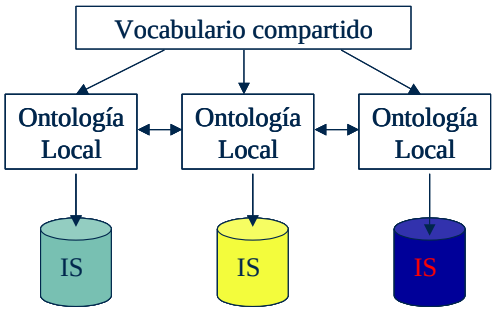


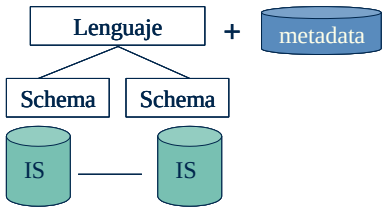

**FIGURA 2.9. MÉTODO DE ONTOLOGÍA HÍBRIDA**

El vocabulario compartido por las ontologías locales contiene los términos básicos del dominio. Algunas veces el vocabulario compartido se convierte en una ontología . Una implementación de esta tecnología es el sistema BÚSTER .

### 2.2.3 Sistemas Multi - Bases de Datos

Varias propuestas se han realizado para la interoperabilidad de Sistemas de Información Heterogéneos. En la figura 2.10 se muestran algunas de ellas. [FWI03]

Un Sistema Multi Bases de Datos (SMulBD) soporta operaciones en múltiples sistemas de bases de datos componentes (SBDC). Cada SBDC es manejado por un sistema manejador de base de datos (SMBD). Un SBDC en un SMulBD puede ser centralizado o distribuido y puede residir en la misma máquina o en múltiples máquinas conectadas por un subsistema de comunicación. Un SMulBD es llamado homogéneo si todos los SMBD componentes son iguales, de lo contrario será llamado SMulBD heterogéneo.




*Multibases de datos*

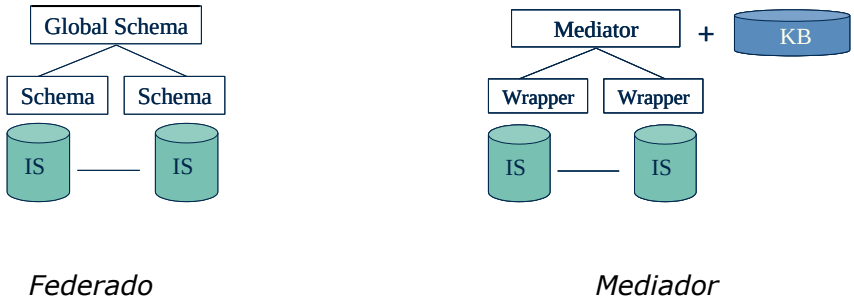



**FIGURA 2.10. APROXIMACIONES PARA INTEROPERABILIDAD DE SISTEMAS DE INFORMACIÓN [FWI03]**

Un SMulBD puede ser clasificado en dos tipos basados en la autonomía de las SBDCs: Sistemas de base de datos no-federado y Sistemas de bases de datos federado.

#### 2.2.3.1 Sistema de Bases de Datos no Federado

Un sistema de bases de datos no federado es una integración de SMBDs componentes que no son autónomos. Esto significa que los SBDCs al participar en una federación pierden su autonomía y cualquier operación debe hacerse sobre la base de datos global. Un Sistema de este tipo no distingue entre usuarios locales y usuarios no locales.

#### 2.2.3.2 Sistema de Bases de Datos Federados

Un sistema de bases de datos federados (SBDF) consiste de SBDCs que son autónomos, participan en una federación para permitir compartición parcial y controlada de sus datos. El concepto de autonomía implica que los SBDCs tienen control sobre los datos que ellos manejan. Ellos cooperan para permitir diversos grados de integración. No hay control centralizado en una arquitectura federada debido a que los SMBDs controlan el acceso a sus datos.

Para permitir la compartición controlada de datos mientras preserva la autonomía de los SBDCs y continuar con la ejecución de aplicaciones existentes, un SBDF soporta dos tipos de operaciones: local y global (federación). Esta división de operaciones globales y locales es una característica esencial de un SBDF. Las operaciones globales involucran acceso a los datos usando un sistema manejador de bases de datos federado y puede involucrar manejar datos por múltiples SBDCs.

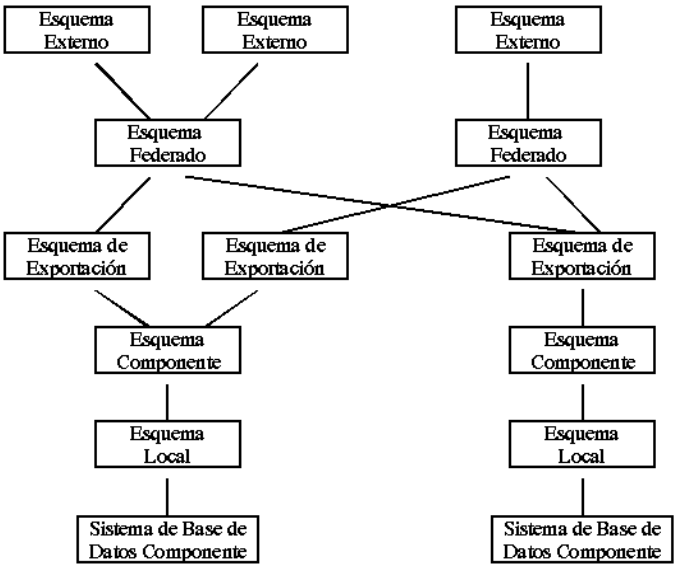



**FIGURA 2.11 ARQUITECTURA DE UN SBDF [FWI03]**

La figura 2.11 propone una arquitectura de 5 niveles de esquema para un SBDF: esquema local, esquema componente, esquema de exportación, esquema federado y esquema externo.

- Esquema local: Esquema conceptual del SBDC.
- Esquema componente: Derivado de trasladar el esquema local en un modelo de datos llamado canónico o modelo de datos común.
- Esquema de exportación: Representa un subconjunto de un esquema componente que esta disponible para el SBDF.
- Esquema federado: Integración de múltiples esquemas de exportación.
- Esquema externo: Define un esquema para un usuario y/o aplicación.

### 2.2.4 Sistema Mediador

Un Sistema Mediador logra interoperabilidad entre sistemas de información, usando módulos inteligentes para procesar las consultas de los usuarios. En general un sistema mediador está compuesto de un mediador, el cual localiza la información

relevante a través de los sistemas de información, resolviendo los conflictos de estructura y semántica de los datos, y, los envoltorios, que son las interfaces de cooperación con los sistemas de información. Bajo este contexto los datos permanecen en las fuentes y es el mediador el responsable de proporcionar a los usuarios la apariencia de estar realizando consultas sobre un único esquema global.[FWI03]

En general estos sistemas utilizan una arquitectura de 3 niveles: aplicación cliente , mediador y wrapper. Los Wrappers sirven como traductores para convertir datos y solicitudes entre el modelo de datos de la información subyacente y el modelo soportado por el mediador, en nuestro sistema aplicamos un modelo de datos basado en XML a nivel del mediador, un ejemplo de otro medidor basado en XML se encuentra en el proyecto MIX (Mediation of Information using XML).

En general el mediador recibe una solicitud del cliente, fragmenta la solicitud de acuerdo a las capacidades de las fuentes y envía los fragmentos a las fuentes apropiadas, dado que las fuentes devuelven los resultados individuales al mediador, éste las integra en una sola y las devuelve al usuario. [IGI00]

De manera general el sistema provee funcionalidades para el acceso a bases de datos tanto de información geográfica como de datos comunes.

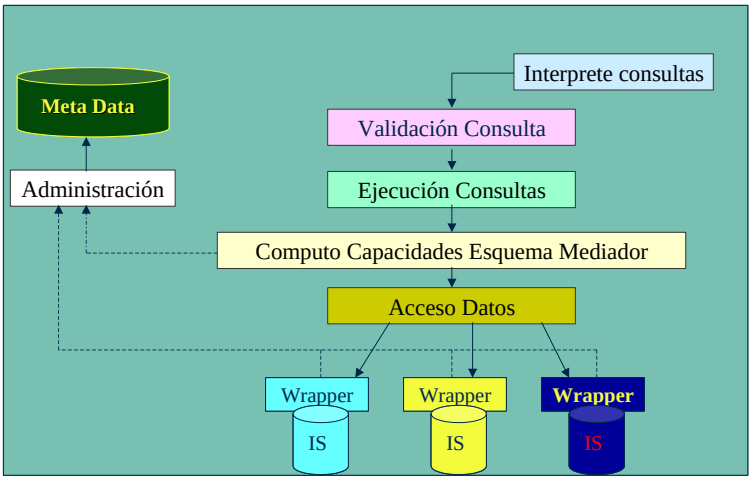



**FIGURA 2.12. ARQUITECTURA DE UN SISTEMA MEDIADOR [FWI03]**

La figura 2.12, muestra la arquitectura de un Sistema Mediador, a continuación se enumeran y describen los distintos módulos generales que la componen.

**Diccionario de datos**: Almacena la definición de todas las estructuras de meta datos: esquemas globales, tipos, operadores, descriptores de las características de las fuentes (envoltorios). Este módulo es importante para la Base de Datos Virtual, ya que los datos están almacenados en lugares remotos, y los únicos datos de la Base de Datos Virtual están almacenado en el diccionario de datos.

**Módulo de Administración:** Facilita la manipulación de los meta datos. Las tareas fundamentales de este módulo son gestionar los procesos en las fuentes de datos y de configurar los envoltorios.

**Módulo Interprete de Consultas**: Traduce la consulta de nivel de usuario a la representación interna de la Base de Datos Virtual.

**Módulo de Cómputo de Capacidades del Esquema Mediador**: Encargado de generación de nuevas vistas (Vistas Globales de la Base de Datos Virtual) en función de la combinación de las capacidades de las fuentes. Las capacidades de las vistas fuente se generan a partir de la especificación de las restricciones de búsqueda que las fuentes encapsulan; en otras palabras las capacidades de las vistas fuentes son generadas por los envoltorios. Estas capacidades son meta datos indispensable para el módulo de ejecución de consultas la Base de Datos Global.

En éste módulo de cómputo, cabe resaltar los procesos llevados a cabo por el módulo minimizador de métodos de búsqueda de las vistas, ya que existen un sinnúmero de métodos implementados por cada fuente.

**Módulo de Validez de Consulta**: Indica si una consulta puede o no ser resuelta por la Base de Datos Virtual.

**Módulo de ejecución de consulta**: Encargado de generar los planes de consulta para las consultas admitidas por el módulo anterior. En éste módulo se resaltan los siguientes sub-módulos:

- Planes de ejecución: Los cuales se forman mediante la unión de los distintos métodos de búsqueda que se activan para la consulta en cada una de las fuentes de datos.

- Optimizador de planes: su función es tener en cuenta la implementación de cada uno de los métodos de búsqueda activados, tal como su complejidad y capacidades de la fuente.

**Módulo de Acceso a Datos**: Encargado de la ejecución de los envoltorios (Wrappers) de cada una de las fuentes de datos.

**Módulo de Seguridad**: Encargado de permitir o denegar el acceso a usuarios a determinadas operaciones sobre la Base de Datos Virtual.

**Planificador de Tareas**: Se puede considerar como un módulo independiente que opera "supervisando" a intervalos de tiempo la ejecución de las consultas sobre la Base de Datos Virtual.

**Manejador de Resultados**: Realiza acciones sobre los resultados obtenidos (enviar correo, etc.)

Además de los módulos que componen la arquitectura del sistema mediador, también es importante conocer las fases a seguir para la creación de un sistema de este tipo.

**Modelado de las relaciones de las fuentes**: Cada fuente tiene un conjunto de relaciones llamadas relaciones base; cada relación está compuesta por un conjunto de atributos, cada uno de los cuales pertenece a un tipo de dato. Estas relaciones presentan limitaciones en la manera en que pueden ser consultadas, las cuales están expresadas en los métodos de búsqueda. Cada método de búsqueda está compuesto por:

- Capacidades negativas: restricciones que deben satisfacer las consultas.
- Capacidades Positivas: Capacidades ofrecidas por la relación.

- Conjunto de atributos de salida: atributos que componen la respuesta a la consulta lanzada contra la relación.

Las capacidades son expresadas por cuatro uplas las cuales representan los atributos, los operadores que pueden ser utilizados en la consulta, la multiplicidad que define cuántos valores pueden consultarse a la vez, y los valores posibles por los que puede consultarse el atributo.

**Generación de Envoltorios**: Se debe proporcionar acceso a una fuente de tal modo que de cara al mediador se comporte similar a una tabla de una Base de Datos Relacional.

**Definición del esquema Global**: Cada relación del esquema global se define mediante una consulta sobre las relaciones base, de forma similar a la definición de las vistas en una base de datos relacional. Las consultas se expresan en un lenguaje próximo a SQL en que se escriben las consultas en fase de ejecución. Después de definir las relaciones globales el mediador debe ser capaz de propagar las consultas a través del árbol de vistas, obteniendo automáticamente las capacidades permitidas por las relaciones del esquema global en forma de una serie de métodos de búsqueda.

## 2.3 TECNOLOGIAS DE COMPUTACIÓN DISTRIBUIDA

La computación de objetos distribuidos extiende el paradigma de programación orientada a objetos permitiendo que los objetos se encuentren distribuidos a través de redes heterogéneas, de forma que cada componente distribuido interopera de manera unificada. Los objetos pueden estar distribuidos en diferentes computadores a lo largo de la red, residiendo dentro de su propio espacio de direcciones fuera de una aplicación, y aun así aparecer como si fueran parte de una aplicación local.

Tres de los paradigmas de objetos distribuidos mas populares son Distributed Component Object Model (DCOM) de Microsoft, Common Object Request Broker Architecture CORBA de la OMG y el Java/Remote Method Invocation (Java/RMI) de JavaSoft.

### 2.3.1 CORBA

CORBA se basa en el protocolo Inter-ORB de Internet para objetos remotos. Todo en la arquitectura CORBA depende del Object Request Broker (ORB). El ORB actúa como un bus de objetos central sobre el cual cada objeto CORBA interactúa transparentemente con otros objetos CORBA localizados tanto local como remotamente. Cada objeto servidor CORBA tiene una interfaz por medio de la cual expone un conjunto de métodos. Para solicitar un servicio, un cliente CORBA adquiere una referencia del objeto al objeto servidor CORBA. El cliente puede entonces hacer invocaciones de métodos en la referencia del objeto como si el objeto servidor CORBA residiera en el espacio de direcciones del cliente. El ORB es responsable de encontrar la implementación de los objetos CORBA, preparándolo para la recepción de solicitudes, comunicando las solicitudes hacia éste y llevando la respuesta de regreso al cliente. Un objeto CORBA interactúa con el ORB ya sea a través de la interfaz del ORB o a través de un adaptador de objetos – el adaptador de objetos básico (BOA) o el adaptador de objetos portable (POA).

Dado que CORBA es una especificación, puede ser usado en diferentes plataformas desde UNIX hasta Windows e incluso dispositivos handheld siempre y cuando haya una implementación del ORB para esa plataforma.

### 2.3.2 COM/DCOM

Con el término COM se conoce tanto la especificación como la implementación de Microsoft que proporciona un marco para la integración de componentes. Este marco soporta la *interoperabilidad* y la *reutilización* de componentes distribuidos, con lo que los desarrolladores pueden construir sistemas a base de ensamblar componentes de diferentes proveedores, que se comunican unos con otros vía COM [DTO96].

COM define un interfaz de programación (API) para la creación de componentes que van a usarse para la construcción de aplicaciones a medida o para permitir que diversos componentes interactúen. Sin embargo, para que esta interacción sea posible, los componentes deben adoptar una estructura binaria especificada por Microsoft. Si lo

hacen, aunque los componentes estén escritos en diferentes lenguajes, pueden ínter operar.

DCOM es una extensión de COM que permite la interacción de componentes en una red. Mientras que los procesos COM pueden funcionar en la misma máquina aunque en diferentes espacios de direcciones, con DCOM estos procesos pueden estar dispersos en diferentes lugares de una red. Con DCOM es posible la interacción de componentes disponibles en diferentes plataformas, siempre que exista DCOM para esos entornos.

Resulta más conveniente considerar COM y DCOM como una sola tecnología que proporciona una serie de servicios para la interacción de componentes, desde la integración de objetos en una sola plataforma hasta la interactividad de componentes a través de redes heterogéneas. De hecho, las extensiones COM y DCOM están mezcladas en un solo 'runtime' que proporciona tanto acceso local como remoto.

Cada objeto servidor DCOM puede soportar múltiples interfaces cada una representando comportamientos diferentes de un objeto. Un cliente DCOM invoca los métodos expuestos por un servidor adquiriendo un puntero a una de las interfaces del objeto servidor. El objeto cliente comienza la invocación a los métodos expuestos del objeto servidor a través del puntero a la interfaz adquirido como si el objeto servidor residiera en el espacio de direcciones del cliente. Debido a que la especificación COM está en el nivel binario permite que los componentes servidores COM sean escritos en diversos lenguajes de programación como C++, Java, Object Pascal (Delphi), Visual Basic e incluso Cobol. Siempre y cuando una plataforma soporte servicios COM / DCOM puede ser usado en esa plataforma.

### 2.3.3 RMI

RMI se basa en el protocolo Java remote method Protocol (JRMP). Java usa la serialización de objetos Java, que permite que los objetos sean transmitidos (marshaled) como un flujo. Debido a que la serialización de objetos es específica de Java, tanto el objeto servidor como el cliente tienen que estar escritos en Java. Cada objeto servidor define una interfaz que puede ser usada para acceder a él fuera de la maquina virtual de Java y en otras maquinas virtuales. La interfaz expone un conjunto de métodos que son un indicativo de los servicios ofrecidos por el objeto servidor. Para

que un cliente localice un objeto servidor por primera vez, RMI depende de un mecanismo de nombrado llamado RMIRegistry que corre en una máquina servidor y contiene información sobre los objetos servidores disponibles. Un cliente adquiere una referencia del objeto servidor haciendo un *"lookup"* en el RMIRegistry. Una vez que obtiene la referencia del objeto hace invocaciones de los métodos en el objeto servidor como si éste residiera en el espacio de direcciones del cliente. Los objetos servidores son nombrados usando URL's y para un cliente adquirir una referencia al objeto servidor , este debe especificar la URL del objeto servidor.

### 2.3.4 Arquitectura RPC Básica

Tanto en DCOM como en CORBA, las interacciones entre un proceso cliente y un objeto servidor están implementadas como comunicaciones estilo RPC orientadas a objetos. La figura 2.13 muestra una estructura RPC típica. Para invocar una función remota, el cliente hace una llamada al stub cliente. El stub empaqueta los parámetros de la llamada en un mensaje de solicitud, e invoca un protocolo de transporte para enviar el mensaje al servidor. En el lado del servidor, el protocolo de transporte entrega el mensaje en el stub servidor, el cual lo desempaqueta y llama la función invocada en el objeto. En DCOM, el stub cliente se denomina proxy y el stub servidor se denomina stub. En contraste, el stub cliente en CORBA se denomina stub y el stub servidor se denomina skeleton. Algunas veces, el termino proxy se usa para referirse a una instancia en ejecución del stub en CORBA. Con respecto a SOAP y Servicios Web, el stub cliente se denomina como service proxy y el stub servidor como el service implementation template.

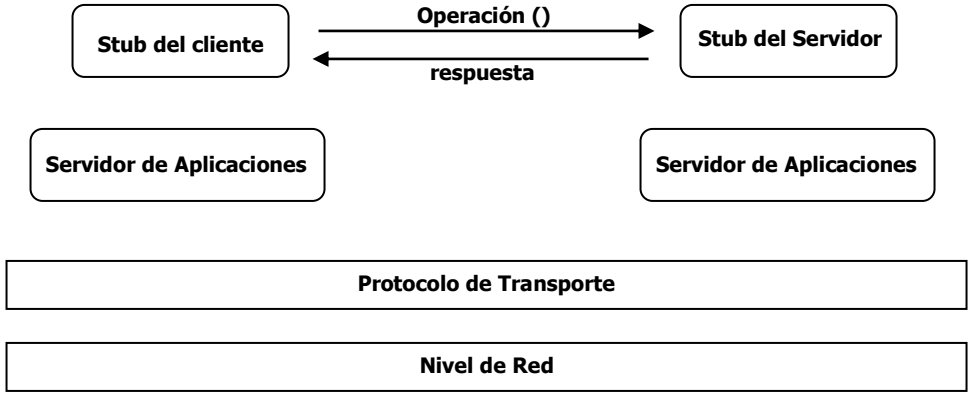



**FIGURA 2.13. ARQUITECTURA RPC BÁSICA**

### 2.3.5 Interoperabilidad

DCOM y CORBA IIOP tienen muchas similitudes; ambos protocolos usan identificadores para ubicar el objeto buscado dentro del middleware y ambos usan identificadores de método para determinar el método a ser invocado.

Sin embargo, con esas similitudes vienen también algunas diferencias que dificultan la interoperabilidad. Las tres diferencias principales son:

**Referencias a objetos:** En CORBA/IIOP, la referencias a objetos se denominan Interoperable Object Reference (IOR). Las IOR's contienen información de direccionamiento en un formato portable el cual cualquier producto CORBA puede resolver a un endpoint objeto. En DCOM, las referencias a objetos se denominan OBJREF, la cual combina un conteo de referencias distribuidas con identificaciones de endpoint/objeto. Desafortunadamente, las IOR's no se correlacionan con las OBJREF's, lo cual resulta en un problema de interoperabilidad entre las aplicaciones CORBA y DCOM.

**Soporte para múltiples interfaces por objeto:** En CORBA, el identificador de interfaz está implícito debido a que sólo una interfaz de objeto se soporta, mientras que DCOM soporta múltiples interfaces por objeto.

**Formato de los parámetros del payload:** En DCOM, el payload está escrito en un formato conocido como Network Data Representation (NDR). En IIOP/GIOP, el payload esta escrito usando el formato Common Data Representation(CDR). Tanto el NDR como el CDR trabajan con representaciones de datos distintas usadas en diversas plataformas. Algunas diferencias existen entre estos dos formatos que los hacen incompatibles el uno con el otro.

### 2.3.6 Desventajas de CORBA y DCOM

A pesar de que CORBA y DCOM han sido implementados en diversas plataformas, la realidad es que cualquier solución construida en esos protocolos será dependiente de la implementación de un proveedor. Así, si se fuera a desarrollar una aplicación DCOM, todos los nodos participantes en la aplicación distribuida deberán estar corriendo en

Windows. En el caso de CORBA, cada nodo en la aplicación necesitaría correr sobre el ORB del mismo proveedor. Ahora hay casos en los que ORB's CORBA de distintos proveedores pueden ínter operar. Sin embargo, la interoperabilidad no se extiende a servicios de alto nivel tales como seguridad y gestión de transacciones.

### 2.3.7 ¿Por Que Servicios Web?

Con el advenimiento de Internet, los negocios tienen una tendencia cada vez más fuerte de integrarse con aplicaciones distribuidas fuera de su empresa. Por lo tanto, se necesita un modelo de computación distribuida independiente de la plataforma, el proveedor y el lenguaje, de manera que ofrezca una verdadera interoperabilidad. Adicionalmente, debe ser simple para los programadores usar el protocolo y hacer el despliegue de las aplicaciones. Esto requiere fácil acceso a las implementaciones del protocolo tanto del lado del cliente como del servidor.

Entre el stack de tecnologías de Servicios Web, hay un conjunto de especificaciones abiertas que son estándares existentes o especificaciones ampliamente aceptadas que están en el proceso normal para convertirse en estándares. El stack básico esta comprendido de HTTP, XML, SOAP, WSDL y UDDI.

En la base del stack encontramos a HTTP, un protocolo similar a RPC que es simple, ampliamente difundido y aceptado por los firewall's. Luego encontramos un lenguaje de representación de datos común en XML, el cual también está ampliamente generalizado. SOAP, un protocolo de mensajes basado en XML, es independiente de la plataforma y el lenguaje. Soporta los modelos de envío de mensajes y de solicitud / respuesta. Como en CORBA y DCOM, requiere de una IDL. En este caso se trata de WSDL, que es una IDL de servicio basado en XML que define la interfaz del servicio así como las características de la implementación. Por ultimo, se tiene a UDDI, que define una forma de publicar y descubrir información sobre Servicios Web [UDD01]. En él encontramos descripciones de las entidades de negocio y sus Servicios Web, así como descripciones técnicas de cada servicio.

En HTTP, los mensajes de solicitud y respuesta pueden contener un payload arbitrario. Los headers HTTP son solo texto plano, lo que lo hace fácil de usar para los

programadores de Internet promedio. Además, HTTP emplea TCP/IP como el protocolo de comunicaciones de red para sus mensajes solicitud / respuesta. Un cliente HTTP se conecta a un servidor HTTP usando TCP. Después de establecer la conexión TCP, el cliente puede enviar un mensaje de solicitud HTTP al servidor. El servidor envía un mensaje de repuesta HTTP de regreso al cliente después de procesar la solicitud. Puesto simplemente, HTTP realiza un transporte independiente del payload, que ofrece la mayoría de las características de gestión de conexión encontradas en CORBA y DCOM. También hace el uso de URL's para las referencias a objetos que coincide con las IORs y los OBJREFs, encontrados en CORBA y DCOM, respectivamente.

Dado que HTTP es independiente del payload, carece de un mecanismo para la representación de los valores de los parámetros en los mensajes RPC. Es aquí donde XML entra en acción. XML es un lenguaje de representación basado en etiquetas, neutral a cualquier plataforma. Permite que los datos sean serializados en un formato de mensaje que es fácilmente decodificado en cualquier plataforma. Sin embargo, a diferencia de NDR y CDR, XML es fácil de usar, ofrece un formato de datos flexible y es soportado prácticamente en todas las plataformas de computación. Una vez más, es abierto y ampliamente adoptado.

El stack de tecnologías de los Servicios Web tiene a SOAP como el estándar abierto ORPC (RPC orientado a objetos), para el mapeo de los objetos de la aplicación a los protocolos de red. A pesar de que SOAP no está atado a un protocolo de transporte especifico, HTTP se ha convertido en el más usado para dicho fin. Usando HTTP, un SOAP envelope usa XML como un esquema de codificación para la solicitud y respuesta de parámetros. Un mensaje SOAP, es básicamente una solicitud y una respuesta que cumplen con las reglas de codificación SOAP.

Todas las características descritas anteriormente denotan claramente algunas de las ventajas de los Servicios Web y de sus virtudes para el trabajo sobre Internet. Entrando a un mayor nivel de detalle podemos decir que entre las ventajas de los Servicios Web encontramos:

- Estándar aceptado por los principales fabricantes de software (incluido Microsoft).

- Los Servicios Web facilitan la interoperabilidad. La interacción entre un proveedor de servicio y un solicitante de servicio está diseñada para ser completamente independiente de la plataforma y el lenguaje.

- Los Servicios Web se comunican a través de HTTP, lo que permite que interactúen a través de Internet.

- Los Servicios Web facilitan la integración. Como los solicitantes de servicios usan agentes de servicios (service brokers) para encontrar proveedores de servicios, el proceso de descubrimiento se lleva a cabo dinámicamente. Una vez que el solicitante y el proveedor se han encontrado el uno al otro, el documento WSDL del proveedor del servicio se usa para ligar el solicitante y el servicio. Esto significa que solicitantes, proveedores y agentes trabajan juntos para crear sistemas auto configurables, adaptables y robustos.

- Su desarrollo e implementación son bastante sencillos, comparados con protocolos de integración anteriores (CORBA, RMI, DCOM).

- Los Servicios Web reducen la complejidad por medio del encapsulamiento. Un solicitante del servicio no se preocupa de cómo el proveedor del servicio implementó su servicio, y un proveedor del servicio no se preocupa de como el solicitante del servicio lo usa. Esos detalles son encapsulados por los proveedores y los solicitantes.

### 2.3.8 SERVICIOS WEB

#### 2.3.8.1 Definición

Según [WSA03], un servicio web se define de la siguiente forma:

*"Un Servicio Web es un sistema software identificado por una URI, cuyas interfaces públicas son definidas y descritas usando principalmente el estándar WSDL(Web*

*Services Description Language), el cual cubre todos los detalles necesarios para interactuar con el servicio, incluyendo los formatos de mensaje (que detallan las operaciones), protocolos de transporte y localización*".

La interfaz esconde los detalles de implementación, permitiendo que este se use independientemente de la plataforma hardware o software en la que este implementado y del lenguaje de programación utilizado. Esto permite que las aplicaciones basadas en Servicios Web tengan un alto grado de interoperabilidad entre las distintas tecnologías utilizadas para las implementaciones [SWE01]. Otros sistemas podrían interactuar con el Servicio Web en la forma preestablecida por su definición usando XML basado en mensajes transportados por protocolos de Internet. La forma más difundida para acceder a un Servicio Web es a través de SOAP (Simple Object Access Protocol) sobre HTTP.

### 2.3.8.2 Arquitectura de los Servicios Web

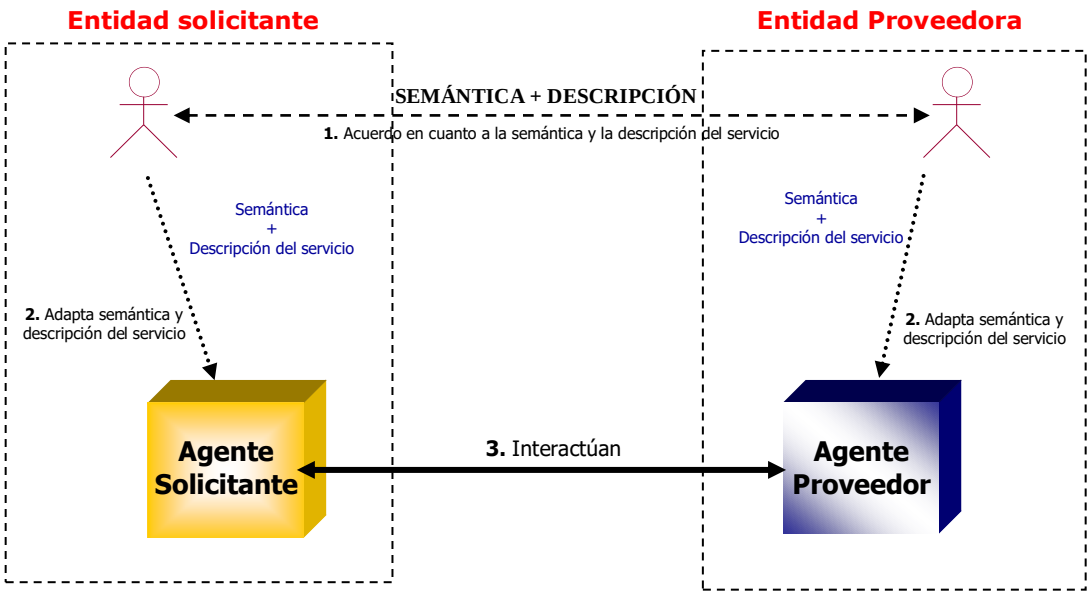



**FIGURA 2.14. ROLES BÁSICOS DE LA ARQUITECTURA DE SERVICIOS WEB**

***Agentes y Servicios***

Un Servicio Web es visto como una noción abstracta que debe ser implementada por un agente concreto (Ver figura 2.14). El agente es la entidad lógica (una pieza de software) que envía y recibe mensajes, mientras que el servicio es el conjunto abstracto de funcionalidad que es provisto. Para ilustrar esta distinción, se podría implementar un Servicio Web particular un día usando un agente (escrito en un lenguaje de programación x), y un agente diferente el día siguiente(escrito en un lenguaje de programación y). Al tiempo que el agente puede cambiar, el Servicio Web permanece igual [WSA03].

***Solicitante y Proveedor***

El propósito de un Servicio Web es proveer alguna funcionalidad en representación de su dueño – una entidad legal, tal como un negocio o un individuo. La entidad proveedor es la entidad legal que provee un agente apropiado para implementar un servicio particular(ver figura 2.14).

La entidad solicitante es una entidad legal que desea hacer uso del Servicio Web de la entidad proveedora. Éste usará un agente solicitante para intercambiar mensajes con el agente proveedor de la entidad proveedora. Con el fin de que haya un intercambio de mensajes exitoso, la entidad solicitante y la entidad proveedora deben ponerse primero de acuerdo en la semántica y los mecanismos del intercambio de mensajes.

***Descripción del servicio***

La descripción del servicio es una especificación procesable de máquina de los formatos de mensaje, tipos de datos y protocolos que deberían ser usados entre el agente solicitante y el agente proveedor. Este también especifica la ubicación de red del agente proveedor, y podría proveer alguna información sobre el patrón de intercambio de mensajes esperado.

***Semántica***

La semántica del intercambio de mensajes representa el "contrato" entre la entidad solicitante y la entidad proveedora relativo al propósito y las consecuencias de la interacción. También incluye algunos detalles adicionales sobre los mecanismos del intercambio de mensajes que no son especificados por la descripción del servicio. Aunque este contrato representa el contrato completo entre la entidad solicitante y la entidad proveedora sobre cómo y por qué sus respectivos agentes interactuarán, esto no está necesariamente escrito o explícitamente negociado. Podría ser explicito o implícito, oral o negociado, procesable por la máquina u orientado a humanos.

Mientras que la descripción del servicio representa un contrato que gobierna los mecanismos de interacción con un servicio particular, la semántica representa un contrato gobernando el significado y propósito de la interacción.

***El rol de las personas:*** Aunque unos de los principales propósitos de los Servicios Web es automatizar procesos que serían de otra manera ejecutados manualmente, las personas todavía juegan un rol en su arquitectura y uso, notable en dos formas:

1. Las personas necesitan ponerse de acuerdo en la semántica y la descripción del servicio. Ya que una persona ( u organización) a fin de cuentas es el dueño legal de cualquier Servicio Web, las personas deben implícita o explícitamente acordar la semántica y la descripción del servicio que gobernara la interacción.

2. Las personas crean los agentes solicitante y proveedor (ya sea directa o indirectamente). Finalmente, las personas deben asegurarse de que esos agentes implementen los términos de la semántica y la descripción del servicio acordados.

### 2.3.8.3 Stack de protocolos de los Servicios Web

La arquitectura de los servicios Web se basa en una serie de estándares que permiten llevar a cabo las roles mencionados y facilitan la interacción entre los componentes de

un Servicio Web y de los Servicios Web entre si [WSO01]. En la figura 2.15 se ilustra la pila de protocolos de Servicios Web:

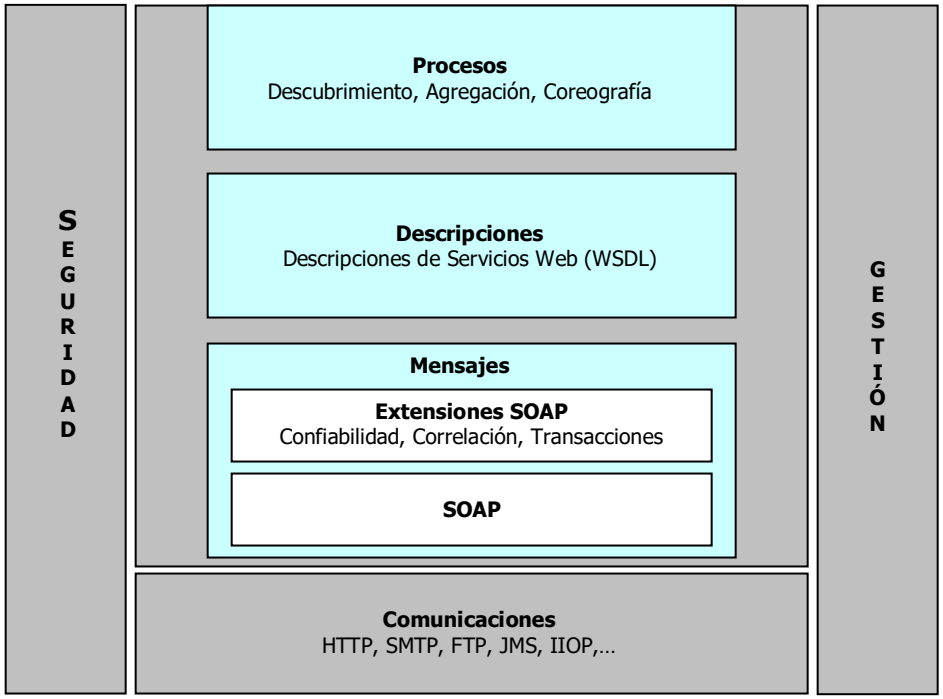



**FIGURA 2.15. STACK DE PROTOCOLOS DE LOS SERVICIOS WEB**

La base de la pila de protocolos de los Servicios Web es el nivel de red. Los Servicios Web deben ser accesibles a través de la red para poder ser invocados por un solicitante del servicio. HTTP es el protocolo de red estándar para Servicios Web disponibles en Internet. Otros protocolos de Internet son soportados, entre los que se encuentran SMTP y FTP. Los dominios de Intranet pueden usar tecnologías como MQSeries, CORBA, etc.

El siguiente nivel , mensajería basada en XML, representa el uso de XML como la base para el protocolo de intercambio de mensajes. SOAP es el protocolo escogido debido a que es simple, posee un formato adecuado para especificar la llamada a procedimientos remotos (RPC), define un mecanismo estándar para incorporar extensiones ortogonales a los mensajes usando las cabeceras y es firewall friendly, ya que el viajar por HTTP utiliza el puerto 80.

Además, las extensiones de SOAP permiten proveer servicios de autenticación, encriptación, control de acceso, procesamiento de transacciones, enrutamiento, confirmación de entrega, etc. Las estructuras SOAP envelope (y attachment) y el modelo de procesamiento de cabecera ha demostrado ser una herramienta robusta para realizar dichas tareas .

WSDL es el estándar escogido para la descripción de servicios basado en XML, pero no es el único, ya que el proceso de descripción también puede ser realizado por medio de descriptores de despliegue (deployment descriptors), que aunque no presentan tanto nivel de detalle como sí lo ofrece WSDL, son una forma bastante sencilla de describir y publicar servicios. WSDL es la descripción del servicio necesaria para soportar la interoperabilidad de los Servicios Web. WSDL define la interfaz y los mecanismos de interacción del servicio.

En el último nivel encontramos la publicación y descubrimiento de Servicios Web. La publicación directa consiste en el envío por parte del proveedor del servicio del documento WSDL al solicitante del servicio. Alternativamente, el proveedor del servicio puede publicar el documento WSDL a un registro UDDI. Debido a que un Servicio Web no puede ser descubierto a menos que haya sido publicado, el descubrimiento de servicios depende de la publicación de servicios. UDDI provee un mecanismo para el almacenamiento de descripciones de Servicios Web. A la vez que UDDI es visto como una especie de directorio , también define una estructura de datos estándar para representar la información de la descripción de servicio en XML.

Desde ya se está trabajando en toda una variedad de descripción de procesos: el proceso de descubrir descripciones de servicios relacionadas por un criterio especifico, el proceso de describir secuencias de mensajes, la agregación de procesos en procesos de mas alto nivel , etc. En esta área encontraremos especificaciones como la Web Service Choreography (que todavía se encuentra en desarrollo), que busca especificar la interacción coordinada entre múltiples Servicios Web en lo que se refiere al intercambio de mensajes, su composición y la secuencia en la cual éstos son transmitidos y recibidos [WSC03].

En adición a las tecnologías de descripción y mensajería , la arquitectura define mecanismos para la gestión y la seguridad de los Servicios Web, involucrados con los diferentes niveles del stack de protocolos por lo que se muestran en la figura 2.15 transversal a ellos.

Veamos detalladamente cada uno de los principales estándares de los Servicios Web.

#### 2.3.8.4 SOAP (Simple Object Access Protocol)

SOAP provee un mecanismo simple y ligero para el intercambio de información estructurada en un ambiente distribuido, por medio de XML. SOAP fue desarrollado para que aplicaciones distribuidas se comunicaran sobre HTTP permitiendo que objetos de cualquier clase, en cualquier plataforma, puedan interoperar de manera simple [SOA00].

El protocolo está compuesto por tres partes:

- El SOAP Envelope que define la estructura general que expresa el contenido, quien debe procesarlo y la naturaleza opcional u obligatoria del mensaje.

- Las reglas de codificación SOAP que definen un mecanismo de serialización que se usa para intercambiar instancias de tipos de datos definidos por la aplicación.

- La representación SOAP RPC que define una convención que se usa para representar llamadas a procedimientos remotos y sus respuestas.

Un mensaje SOAP consiste de un documento XML compuesto varias etiquetas que explicaremos a continuación [ESO00].

##### 2.3.8.4.1 SOAP Envelope

Es la primera parte del documento y es obligatoria. Contiene el nombre del elemento (Envelope), seguida por un espacio de nombre definiendo la versión SOAP que está siendo usada, y el atributo opcional encodingStyle que apunta al enlace donde están

definidas las reglas de codificación y la serialización. El envelope es de la siguiente forma:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">… </SOAP:Envelope>
```

Los espacios de nombre son usados para proveer un contexto y garantizar la unicidad de los elementos.

#### 2.3.8.4.2 SOAP Header

Este es una parte opcional del mensaje SOAP y se encuentra encapsulado en el SOAP envelope. Contiene información para intermediarios, puede contener una o más entradas (entries). Una aplicación SOAP debe incluir espacios de nombres correctos para todos los elementos y atributos definidos en el mensaje. Estos son una URI que apunta a la descripción de la información del mensaje con el fin de garantizar la unicidad del mensaje. No deben ser usadas DTDs.

```
 <SOAP-ENV:Header>
<u:transaction xmlns:u="HTTP://www.unicauca.edu.co/transaccion"
 SOAP-ENV :mustUnderstand="1">
</u:transaction>
</SOAP-ENV:Header>
```

SOAP define tres atributos en el espacio de nombres por defecto ("http://www.w3.org/2001/12/soap-envelope"). Estos atributos son: actor, mustUderstand y encodingStyle. Los atributos definidos en el SOAP Header especifican cómo un nodo debería procesar el mensaje SOAP.

*Atributo actor:* Un mensaje SOAP podría viajar del emisor al receptor pasando por diferentes nodos a lo largo de la ruta del mensaje. No todas las partes del mensaje SOAP están dispuestas para ser procesadas en el nodo final, en cambio, pueden estar dispuestos para uno o más nodos de la ruta del mensaje.

*Atributo mustUnderstand:* El atributo mustUnderstand se usa para indicar si un Header debe ser procesado o no por un nodo específico. Si se fija "mustUnderstand='1'" a un elemento hijo de un elemento Header, esto indica que el receptor al procesar el Header debe reconocer el elemento.

*Atributo encodingStyle:* El atributo SOAP encodingStyle se usa para definir los tipos de datos en el documento. Este atributo puede aparecer en cualquier elemento SOAP, y se aplicará a los contenidos de ese elemento y a todos los elementos hijos.

#### 2.3.8.4.3 SOAP Body

La información a ser procesada por el endpoint se encuentra en el body del mensaje SOAP. Puede contener un conjunto de entradas que se encuentran en la raíz del mensaje body.

```
<soapenv:Body>
   <iniciarSesion xmlns="http://server.messenger">
        <nombre xmlns="">p</nombre>
        <password xmlns="">p</password>
   </iniciarSesion>
 </soapenv:Body>
```

SOAP define un elemento dentro del elemento Body en el espacio de nombres por defecto (("http://www.w3.org/2001/12/soap-envelope"). Este es el elemento Fault, el cual se usa para indicar mensajes de error.

***Elemento SOAP Fault***

Un mensaje de error de un mensaje SOAP se transporta dentro del elemento Fault. Si el elemento Fault está presente, debe aparecer como un elemento hijo del elemento Body. Un elemento Fault solo puede aparecer una vez en el mensaje SOAP. El elemento Fault tiene los siguientes sub - elementos:

| Sub Elemento | Descripción |
|---|---|
| <faultcode> | Un código para la identificación del error. |
| <faultstring> | Una explicación legible del error |
| <faultactor> | Información sobre quien causó el error ocurrido. |
| <detail> | Contiene información del error específica de la aplicación relacionada con el elemento Body. |

**TABLA 2.5 SUB-ELEMENTOS DEL ELEMENTO FAULT**

***Códigos de error SOAP***

Los valores faultCode definidos abajo deben ser usados en el elemento faultCode cuando se describen los errores:

| Error | Descripción |
|---|---|
| VersionMismatch | Espacio de nombre invalido para el elemento SOAP envelope |
| MustUnderstand | Un elemento hijo inmediato del elemento Header, con el atributo mustUnderstand fijado a "1", no fue entendido. |
| Client | El mensaje fue incorrectamente formado o contiene información incorrecta. |
| Server | Hubo un problema con el servidor por lo que el mensaje no pudo proseguir. |

**TABLA 2.6 VALORES FAULTCODE**

Para llevar a cabo el transporte de los mensajes SOAP se usa el protocolo HTTP, debido a que es un protocolo ampliamente aceptado y difundido. A continuación se presenta como se estructura HTTP para el transporte de los mensajes SOAP.

### 2.3.8.4.4 Estructura HTTP

***HTTP Header***

El HTTP header se encuentra justo antes del mensaje SOAP. El protocolo HTTP envía una solicitud POST a través de la red. Se hace de la siguiente manera:

En la primera línea se definen el método HTTP, la URI del endpoint y la versión del protocolo:

```
POST /MSN/services/ServidorSOAP HTTP/1.0
```

La siguiente línea apunta al sitio:

```
Host: sdoo.unicauca.edu.co
```

Las siguientes tres líneas se usan para definir el formato MIME para el despliegue del mensaje, la codificación HTTP y la longitud del mensaje.

```
Content-Type: text/xml;

charset=utf-8

Content-Length: 402
```

Luego, los métodos son añadidos como SOAPAction (SOAPMethodName) que determinan la intención de la solicitud HTTP.

#### 2.3.8.4.5 Estructura SOAP

Abajo encontramos un ejemplo del mensaje de solicitud SOAP, seguido por el mensaje de respuesta, en negrita se encuentran señalados los elementos explicados anteriormente:

```
POST /MSN/services/ServidorSOAP HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.0
Host: sdoo.unicauca.edu.co
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 402
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <iniciarSesion xmlns="http://server.messenger">
   <nombre xmlns="">p</nombre>
   <password xmlns="">p</password>
  </iniciarSesion>
 </soapenv:Body>
</soapenv:Envelope>
```

El servidor procesa la solicitud y envía una respuesta al cliente. La respuesta contiene el código de estado de la solicitud.

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Set-Cookie: JSESSIONID=h667ysrh11;Path=/MSN
Date: Tue, 16 Sep 2003 17:51:40 GMT
Server: Tomcat Web Server/3.3.1 Final ( JSP 1.1; Servlet 2.2 )

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:iniciarSesionResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://server.messenger">
   <iniciarSesionReturn xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[0]"
xmlns:ns2="null" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
  </ns1:iniciarSesionResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

En el ejemplo anterior , el servidor retorna un código de estado de 200. Éste es el código de éxito estándar para HTTP.

#### 2.3.8.4.6 APIS's SOAP

Durante el desarrollo de este proyecto trabajamos con dos implementaciones de la especificaciones SOAP. El API Apache SOAP y el API Apache Axis.

Apache SOAP es una implementación "open source" de las especificaciones en Java del SOAP v1.1 y "Mensajes SOAP con Attachments". Puede ser usado como una librería del cliente para invocar Servicios Web disponible en Internet, o como una herramienta del lado del servidor para implementar servicios SOAP. Como librería cliente provee un API para la invocación de servicios SOAP RPC así como un API para el envió y recepción de mensajes SOAP.

Por el otro lado, Apache Axis es un framework para la construcción de procesos SOAP tales como clientes, servidores, gateways, etc, además de prestar un soporte extensivo para el lenguaje de descripción de Servicios Web WSDL, generar clases java a partir del documento WSDL y monitorear paquetes TCP/IP. Axis es la tercera generación del Apache SOAP. Entre las mejoras que tiene respecto a su predecesor encontramos:

- **Velocidad:** Axis usa el "parseo" SAX, con el que consigue significativas mejoras de velocidad respecto a las primeras versiones del Apache SOAP.
- **Flexibilidad:** La arquitectura Axis ofrece al desarrollador completa libertad para insertar extensiones en el motor para un procesamiento de cabeceras customizado y una mejor gestión del sistema.
- **Soporte WSDL:** Axis soporta WSDL versión 1.1, el cual permite fácilmente construir stubs para acceder a servicios remotos, y también exportar descripciones legibles por maquinas de los servicios desplegados desde Axis.

En los anexos se muestra un ejemplo desarrollado tanto con la implementación del Apache SOAP así como con la implementación del Apache Axis. De esta manera se espera que se adquiera una idea mas clara sobre las diferencias y similitudes entre las dos APIS's.

#### 2.3.8.5 Web Services Description Language (WSDL)

Es el estándar propuesto para la descripción de los Servicios Web, el cual consiste en un lenguaje de definición de interfaz (IDL - Interface Definition Language) basado en XML, que define la interfaz de servicio y sus características de implementación.

Un documento WSDL está compuesto por dos partes, y puede ser dividido en dos documentos (o también estar en el mismo documento):

- La parte de interfaz de servicio , que describe la interfaz de tipo abstracto y su protocolo de enlace.
- La parte de implementación del servicio, que describe la información de acceso al servicio

El documento WSDL de interfaz de servicio incluye los siguientes elementos:

- Port Type
- Message
- Operation
- Binding
- Types

El documento WSDL de implementación del servicio incluye los siguientes elementos:

- Port
- Service

Si los documentos de implementación e interfaz de servicio están almacenados en archivos distintos, el documento de implementación del servicio debe importar el documento de interfaz del servicio.

**Estructura de un documento WSDL**

Un documento WSDL es simplemente un **conjunto de definiciones**. Hay un elemento **definitions** en la raíz y definiciones en su interior. Los servicios se definen utilizando seis elementos principales:

- **types**, que proporciona definiciones del tipo de datos utilizado para describir el intercambio de mensajes.
- **message**, que representa una definición abstracta de los datos que se están transmitiendo. Un mensaje se compone de partes lógicas, cada una de las cuales está asociada a una definición dentro de algún sistema de tipos.
- **portType**, que es un conjunto de operaciones abstractas. Cada operación hace referencia a un mensaje entrante y a mensajes salientes.
- **binding**, que determina las especificaciones del protocolo concreto y del formato de datos para las operaciones y mensajes definidos por un determinado portType.
- **port**, que especifica la dirección para un enlace, definiendo así un único punto final de comunicación.
- **service**, que se utiliza para agregar un conjunto de puertos relacionados.

***Elemento types***

El elemento **types** incluye definiciones de tipos de datos que son relevantes para los mensajes intercambiados. Para lograr una máxima interoperabilidad y neutralidad de la plataforma, WSDL prefiere utilizar XSD como sistema de tipos canónico y lo admite como sistema de tipos intrínseco. La sintaxis para definir un elemento **types** se muestra a continuación.

```
<definitions...>
   <types>
      <xsd:schema.../>
   </types>
</definitions>
```

Dado que no es razonable esperar una única gramática del sistema de tipos que se utilice para describir todos los tipos abstractos presentes y futuros, WSDL permite la agregación de sistemas de tipos mediante elementos de extensibilidad. Un elemento de extensibilidad puede aparecer bajo el elemento **types** para identificar el sistema de definición de tipos que se está utilizando y para proporcionar un elemento XML contenedor de las definiciones de tipos. La función de este elemento es comparable a la del elemento **schema** del lenguaje de los esquemas XML. A continuación se muestra en negrita la ubicación del elemento de extensibilidad dentro del elemento types.

```
<definitions...>
   <types>
      <-- Elemento de extensibilidad del sistema de tipos -->
   </types>
</definitions>
```

***Elemento message***

Los mensajes se componen de una o más **partes** lógicas. Cada parte está asociada a un tipo procedente de algún sistema de tipos que utiliza un atributo de escritura de mensajes. El conjunto de atributos de escritura de mensajes es extensible. WSDL define varios atributos de este tipo para su uso con XSD:

- **element**. Hace referencia a un elemento XSD que utiliza un QName.
- **type**. Hace referencia a un tipo simpleType o complexType de XSD que utiliza un QName.

Se pueden definir otros atributos de escritura de mensajes siempre que utilicen un espacio de nombres distinto al de WSDL. Los elementos de extensibilidad del enlace también pueden utilizar atributos de escritura de mensajes.

La sintaxis para definir un mensaje se muestra a continuación. Los atributos de escritura de mensajes (que pueden variar según el sistema de tipos utilizado) aparecen en negrita.

```
<definitions...>
   <message name="nmtoken">
```

```
    <part name="nmtoken"? element="qname" type="qname"?/>
  </message>
</definitions>
```

El atributo **name** de message proporciona un nombre único entre todos los mensajes definidos en el documento WSDL. El atributo **name** de part proporciona un nombre único entre todas las partes del mensaje.

***Elemento portType***

Un elemento portType es un conjunto de operaciones abstractas y de mensajes abstractos que recibe un nombre. A continuación se muestra su sintaxis.

```
<wsdl:definitions...>
  <wsdl:portType name="nmtoken">
    <wsdl:operation name="nmtoken".../>
  </wsdl:portType>
</wsdl:definitions>
```

El atributo **name** de portType proporciona un nombre único entre todos los tipos de puerto definidos en el documento WSDL. Mediante el atributo **name** se da nombre a una operación. WSDL tiene cuatro primitivas de transmisión que los puntos finales de red pueden admitir:

- **Unidireccional**. El punto final recibe un mensaje.
- **Petición-respuesta**. El punto final recibe un mensaje y envía otro mensaje correlacionado.
- **Solicitud-respuesta**. El punto final envía un mensaje y recibe otro mensaje correlacionado.
- **Notificación**. El punto final envía un mensaje.

***Elemento binding***

Los elementos binding determinan el formato de mensaje y los detalles de protocolo de las operaciones y mensajes definidos por un portType determinado. Puede haber un número indeterminado de binding para un portType concreto. La gramática para enlaces se muestra a continuación:

```
<wsdl:definitions...>
   <wsdl:binding name="nmtoken" type="qname"> *
      <-- extensibility element (1) --> *
      <wsdl:operation name="nmtoken"> *
        <-- extensibility element (2) --> *
        <wsdl:input name="nmtoken"?> ?
           <-- extensibility element (3) -->
        </wsdl:input>
        <wsdl:output name="nmtoken"?> ?
           <-- extensibility element (4) --> *
        </wsdl:output>
        <wsdl:fault name="nmtoken"> *
           <-- extensibility element (5) --> *
        </wsdl:fault>
      </wsdl:operation>
   </wsdl:binding>
</wsdl:definitions>
```

El atributo **name** proporciona un nombre único entre todos los enlaces definidos en el documento WSDL. Los elementos binding hacen referencia al portType que enlazan utilizando el atributo **type.**

***Elemento port***

Un puerto define un punto final de red individual especificando una dirección única para un enlace.

```
<wsdl:definitions...>
   <wsdl:service...>
```

```
    <wsdl:port name="nmtoken" binding="qname">
      <-- Elemento de extensibiidad (1) -->
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

El atributo **name** proporciona un nombre único entre todos los puertos definidos en el documento WSDL. Los elementos de extensibilidad del enlace (1) suelen especificar la información de dirección para el puerto. Un puerto no puede especificar más de una dirección. Un puerto no puede especificar ningún tipo de información de enlace distinta a la información de dirección.

***Elemento service***

Un servicio agrupa un conjunto de puertos relacionados:

```
<wsdl:definitions...>
  <wsdl:service name="nmtoken">
    <wsdl:port.../>
  </wsdl:service>
</wsdl:definitions>
```

El atributo **name** proporciona un nombre único entre todos los servicios definidos en el documento WSDL. Los puertos de un servicio tienen las siguientes relaciones:

- Ningún puerto se comunica con los otros (por ejemplo, la salida de un puerto no es la entrada de otro).
- Si un servicio tiene varios puertos del mismo tipo, pero emplean enlaces o direcciones distintas, los puertos serán alternativos.
- Cada puerto presenta un comportamiento semánticamente equivalente (dentro de las limitaciones de transporte y formato de mensaje impuestas por cada enlace).

- Esto permite a un cliente de un documento WSDL elegir determinados puertos con los que comunicarse basándose en ciertos criterios (protocolo, distancia, etc.).

Para mejor ilustración, en el anexo F se encuentra el documento WSDL que describe el Servicio Web de mensajería, implementado a manera de ejemplo.

### 2.3.8.6 Universal Description, Discovery And Integration

UDDI (Universal, Description, Discovery and Integration) es una especificación para la creación de registros distribuidos de Servicios Web. Un registro UDDI almacena información del negocio, los servicios ofrecidos por esos negocios, e información técnica sobre los servicios. El modelo de datos y la API de programación usada por UDDI, basada en XML y SOAP, provee una forma de publicar y ubicar toda clase de servicios. Los servicios no tienen que ser necesariamente Servicios Web. De hecho, ni siquiera tienen que estar del todo relacionados con la computación.

Un registro UDDI puede ser comparado a un motor de búsqueda en Internet. El motor de búsqueda contiene información categorizada e indexada sobre paginas disponibles en la World Wide Web. Sin embargo, mientras que un motor de búsqueda de Internet solo retorna una URL a una pagina, un registro UDDI necesita retornar no sólo la ubicación de los servicios, sino también información sobre el servicio, como funciona, que parámetros usa, los valores de retorno, etc.

Específicamente, UDDI está hecho para soportar tres clases de datos de registro: Páginas Blancas (organizando los negocios por nombre), Páginas Amarillas (organizando los negocios por categorías), y Páginas Verdes (organizando los negocios por servicios) [UDD01]. Estos tres tipos de especificación, que hacen parte de la especificación UDDI, dan un vistazo general de los diferentes tipos de datos que pueden ser almacenados dentro del registro.

***Páginas Blancas:*** Las páginas blancas contienen información de un negocio en sí, incluyendo un nombre, detalles de contacto y la ubicación del negocio. Adicionalmente, identificadores únicos como tax Ids o D-U-N-S numbers pueden ser adheridos para identificar unívocamente el negocio. Una vez que esta información es

ingresada en el registro, un cliente está en la capacidad de buscar en el registro basado en la información del negocio. La información de las páginas blancas es facilitada por medio del elemento <businessEntity>, que será explicado más adelante.

***Páginas Amarillas:*** Las páginas amarillas contienen información categorizada sobre los servicios provistos por un negocio. La categorización es hecha por la asignación de una o más taxonomias al negocio. La información de las paginas amarillas también esta asociada con el elemento <businessEntity>.

***Páginas Verdes:*** Las páginas verdes contienen información técnica sobre el servicio ofrecido por un negocio. Esta información incluye toda la información técnica, necesaria para el uso del servicio. Información como ubicación del servicio, la categoría a la cual el servicio pertenece, y la especificación para los servicios, puede ser encontrada en las páginas verdes. La información de las páginas verdes es facilitada por medio de los elementos <businessService> y <bindingTemplate>. Estos elementos serán explicados mas adelante.

#### 2.3.8.6.1 Estructuras De Datos Del UDDI

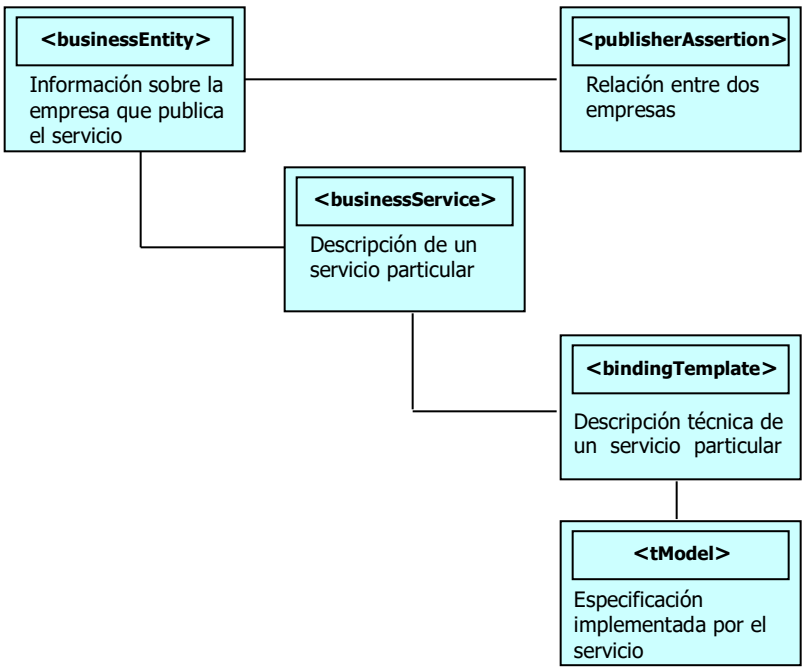



**FIGURA 2.16 ESTRUCTURAS DE DATOS UDDI**

UDDI define 5 estructuras de tipos de datos para especificar un acceso en el registro. Esta división en 5 estructuras hace bastante fácil la búsqueda y entendimiento de los diferentes tipos de información. Cada una de estas estructuras de datos está representada por un documento XML, que contiene tanto la información descriptiva como la técnica.

***BusinessEntity***

La estructura businessEntity contiene toda la información descriptiva sobre el negocio y los servicios que ofrece. La información incluye nombre y descripción del negocio así como información de contacto, categorización y relaciones con otros negocios. Esta estructura puede ser vista como la estructura de más alto nivel de un servicio en el registro.

***BusinessService***

Cada estructura businessEntity contiene una o más estructuras businessService. Una estructura businessService describe un conjunto de servicios categorizados ofrecidos por un negocio. Un elemento businessService no es exclusivo de un elemento businessEntity, puede ser compartido entre múltiples negocios.

***BindingTemplate***

La estructura bindingTemplate contiene la descripción técnica de un servicio. Cada bindingTemplate pertenece únicamente a un elemento businessService.

***tModel***

Uno de los elementos claves de UDDI es el tModel. Un tModel describe la especificación, el comportamiento, el concepto o incluso el diseño compartido, al cual un servicio sigue. Provee información específica sobre cómo interactuar con un servicio. El contenido de una estructura tModel consiste de una clave, un nombre, una

descripción (opcional) y un elemento URL. La URL, en la mayoría de los casos, apunta a la dirección donde puede ser encontrada mayor información sobre el tModel.

***PublisherAssertion***

La estructura publisherAssertion contiene información sobre la relación entre dos partes sostenida por uno o por ambos. Muchos negocios, tales como grandes corporaciones no son efectivamente representados por una businessEntity. Un publisherAssertion puede ser usado para denotar la relación entre los negocios. El contenido de una estructura publisherAssertion consiste de una clave para el primer negocio, una clave para el segundo negocio, y una referencia que especifica la relación sostenida en términos de un par keyName, keyValue dentro de un tModel.
Un identificador único, llamado UUID(Unique Universal Identifier), es generado para cada instancia de datos cuando es insertada en el registro. La forma en que este numero único es generado, no está especificada dentro de la especificación de UDDI. Sin embargo, en la mayoría de los casos el numero es calculado de una combinación de direcciones hardware, time stamps y orígenes aleatorios. El UUID es usado para acceder instancias de datos específicos por demanda y como clave foránea para relacionar múltiples estructuras de datos pertenecientes a una entrada.

#### 2.3.8.6.2 API's UDDI

Las especificaciones de UDDI incluyen definiciones para las interfaces que permiten acceso a la información del registro UDDI. El API se divide en dos partes lógicas: El API de consulta Y el API de publicación. El API de consulta se divide a su vez en dos partes; una parte para la construcción de programas que permitan explorar la información que se encuentra en el registro UDDI, y la otra parte que es usada en el caso de que las invocaciones a servicios Web fallen. Los programadores pueden usar el API de publicación para crear completas interfaces para herramientas que interactúen directamente con el registro UDDI, permitiendo gestionar la información publicada ya sea sobre una businessEntity o sobre una estructura tModel [EUD01].

Entre estas APIs cabe destacar UDDI4J y JAXR, dos APIs utilizadas para la interacción con registros como el UDDI para realizar funciones como la publicación y búsqueda de servicios Web.

#### 2.3.8.6.3 Mapeo WSDL en UDDI

Para facilitar la publicación y búsqueda de descripciones de servicio en un registro UDDI, los documentos WSDL están divididos en 2 tipos: *interfaz de servicio e implementación de servicio.* Observemos esto gráficamente:

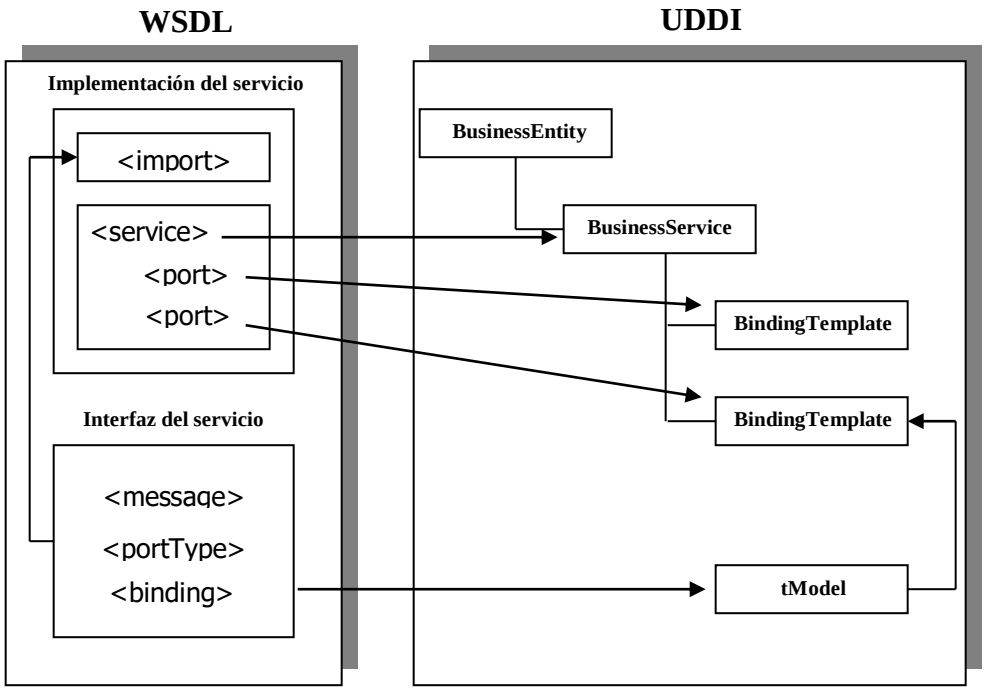



**FIGURA 2.17 MAPEO WSDL A UDDI**

La interfaz de servicio representa una definición reutilizable de un servicio, y es por consiguiente publicada en un registro UDDI como un tModel [HIW02]. La implementación del servicio describe instancias de un servicio. Cada instancia es definida usando un elemento <service>; por consiguiente, cada elemento <service> es usado para publicar un BusinessService UDDI. Cuando publicamos una descripción de servicio WSDL, una interfaz de servicio debe ser publicada como un tModel antes de que una implementación del servicio sea publicada como un BusinessService.

**Mapeo de la interfaz de servicio**

Algunos de los elementos del tModel son construidos usando la información del documento WSDL de interfaz del servicio [HIW02]. La siguiente tabla ilustra los pasos de la creación de un tModel.

| tModel UDDI | Interfaz de servicio WSDL | Descripción |
|---|---|---|
| <name> | Atributo targetNameSpace del elemento <definitions> | El nombre del tModel es fijado usando el targetNameSpace del documento de la interfaz del servicio. Un nombre consistente es necesario para asegurar que el tModel pueda ser localizado usando únicamente la información del documento de implementación del servicio |
| <description> | Elemento hijo <documentation > del elemento <definitions> | El elemento descripción del tModel esta restringido a 256 caracteres y es fijado a los primeros 256 caracteres del elemento hijo <documentation> del elemento <definitions> en el documento de interfaz del servicio. |
| <overviewURL> | Especificación de la URL y enlace del documento de la interfaz de servicio | La ubicación del documento de interfaz de servicio debe ser fijado en el elemento <overviewURL>. SI hay mas de un enlace en el documento de interfaz de servicio, el enlace debe ser codificado en la URL |
| <categoryBag> | No aplicable | El elemento <categoryBag>para el tModel debe contener por lo menos una referencia clave. Esta referencia clave debe contener una referencia al tModel uddi-org:types y el valor de la clave debe ser wsdlSpec. Esta entrada identifica el tModel como una definición de interfaz de servicio WSDL |

**TABLA 2.7 PASOS DE LA CREACIÓN DE UN TMODEL**

**Mapeo de la implementación del servicio**

Una implementación del servicio es publicada en un registro UDDI como un elemento BusinessService con uno o mas elementos BindingTemplate. El elemento BusinessService es publicado por el proveedor del servicio [HIW02].

Un nuevo elemento BusinessService es creado por cada elemento <service> que esté definido en el documento de implementación del servicio. La siguiente tabla describe los elementos BusinessService que pueden ser creados con base en los contenidos del documento de la implementación del servicio.

| UDDI BusinessService | Implementación del servicio WSDL | Descripción |
|---|---|---|
| <name> | Atributo nombre del elemento <service> | El elemento <name> para el BusinessService es fijado a partir del atributo name del elemento <service> en el documento de implementación del servicio |
| <description> | Elemento hijo <documentation> del elemento <service> | El elemento <description> es fijado a partir de los contenidos del elemento <documentation> dentro del elemento <service>. Si no hay elemento <documentation>, el elemento <description> no es fijado |

**TABLA 2.8 ELEMENTOS BUSINESSSERVICE**

Un elemento BindingTemplate es creado dentro de un elemento BusinessService por cada elemento hijo <port> de un elemento <service>.

| UDDI bindingTemplate | Implementación del servicio WSDL | Operación |
|---|---|---|
| <description> | Elemento hijo <documentation> del elemento <port> | Si el elemento <port> tiene un elemento hijo <documentation> en el documento de implementación del servicio, un elemento <description> es fijado con los primeros 256 caracteres del elemento <documentation> |
| <accessPoint> | Atributo "location" del | Para un enlace SOAP, el |

| | | |
|---|---|---|
| | elemento extensión asociado con el elemento <port> | elemento <accessPoint> es fijado al valor del atributo "location" del elemento extensión asociado con el elemento <port>. El elemento contendrá la URL, y el atributo URLType es fijado con base en el protocolo de la URL. |
| <tModelInstanceInfo> | No aplicable | El elemento bindingTemplate contiene por lo menos un elemento <tModelInstanceInfo> por cada tModel que referencie |
| <overviewURL> | No aplicable | El elemento <overviewURL> puede contener una referencia directa al documento de implementación del servicio, el cual es usado solo para proveer acceso a documentación legible por humanos. Mantener una referencia directa al documento original WSDL asegura que el documento publicado es el mismo que es retornado por una operación "find". |

**TABLA 2.9. ELEMENTOS BINDINGTEMPLATE**

### 2.3.8.7 Herramientas

#### 2.3.8.7.1 WSDK

El IBM WebSphere SDK para Web Services, V5 facilita crear y probar servicios Web basados en Java . El kit de desarrollo incluye el IBM SDK para Java, el entorno de ejecución, un registro UDDI, herramientas, ejemplos y documentación [IWS03]. El WSDK está basado en especificaciones abiertas para Servicios Web tales como SOAP, WSDL y UDDI y corre tanto en Windows como en Linux.

El WSDK contiene :

- Un servidor de aplicaciones.
- Un conjunto de herramientas

- Documentación
- Una serie de ejemplos sobre Servicios Web

El servidor de aplicaciones incluye un servidor UDDI, usado para la publicación y el descubrimiento de servicios Web usando los protocolos UDDI. El servidor UDDI corre como una aplicación empresarial (EAR) dentro del servidor de aplicaciones.

El servidor de aplicaciones provisto con el WSDK 5.0 está basado en el servidor de aplicaciones WebSphere V5.0. Es un servidor basado en Java 2 Enterprise Edition (J2EE) que provee las funcionalidades de un servidor Web http, un servlet engine, y un contenedor EJB. Provee capacidades para servicios Web basado en los estándares definidos por J2EE , particularmente JAX-RPC (JSR101) , y Servicios Web para J2EE (JSR 109) [WSJ02]. El servidor de aplicaciones también incluye el IBM Java SDK V1.3.1, el cual es usado para correr el servidor en sí, y para construir y correr aplicaciones Java del lado del cliente.

Dentro de las principales herramientas del WSDK, encontramos:

- *Bean2WebService:* Genera un Servicio Web listo para ser publicado (deploy) a partir de una clase Java.
- *EJB2WebService:* Genera un Servicio Web listo para ser publicado a partir de un EJB stateless session (contenido en un archivo EAR).
- *WSDL2Web Service:* Genera Servicios Web a partir de uno o más documentos WSDL.
- *UDDIPublish:* Agrega ya sea una business entity o una business service a un registro UDDI.
- *UDDIUnpublish:* Remueve ya sea una business entity o una business service de un registro UDDI.
- *Tcpmon:* Monitorea y despliega los mensajes SOAP que son intercambiados entre un Servicio Web y un cliente.

Personalmente, nos pareció un excelente editor, muy fácil de usar, con una documentación bastante buena y ejemplos muy claros, aunque no tiene una interfaz de usuario amigable, ya que es en modo DOS. Además, se debe tener un buen equipo

para usarlo ya que el servidor de aplicaciones consume muchos recursos. También hay que tener en cuenta que trabajamos con una versión *trial* ya que el editor es licenciado.

#### 2.3.8.7.2 JBuilder8

JBuilder está en la capacidad de usar los toolkits Apache Axis, Apache SOAP 2 o WebLogic para el soporte de SOAP. Genera el documento WSDL para la publicación de un Servicio Web a partir de la clase java. También puede tomar el documento WSDL y crear las clases cliente para la invocación del servicio Web [JBW02].

Estas tareas se llevan a cabo por medio de una serie de wizards, entre los que se encuentran:

- Import a Web Service: Este wizard genera las clases java a partir del documento WSDL.
- Export as a Web Service: Este wizard genera el documento WSDL a partir de una clase java exponiendo los métodos de la clase como un Servicio Web.

JBuilder incluye el Explorador de Servicios Web para buscar servicios, así como publicarlos en un registro UDDI.

JBuilder puede ser configurado para trabajar con los servidores:

- Borland Enterprise Server 5.0.2-5.1.x
- webLogic Server 7.0
- WebSphere Application Server 4.0 AES/AE

Para la implementación de la aplicación desarrollada en este trabajo de grado se usó el servidor Jakarta Tomcat, debido a que es libre, da soporte a las necesidades de la aplicación y era el servidor por defecto que tiene configurado el JBuilder para montar los servicios Web a través de sus wizard. No sobra decir que JBuilder es una herramienta que facilita el proceso de creación de un Servicio Web, ya que genera todos los descriptores necesarios, para el montaje del Servicio Web en el servidor, en este caso el servidor Tomcat, y el documento WSDL de descripción del servicio.

# CAPITULO 3. APLICACIÓN PARA LA VALIDACIÓN DE UN SISTEMA MEDIADOR

A pesar de la amplia difusión de los Sistemas de Información Geográficos, promovida principalmente por el descubrimiento de su aplicabilidad en los diversos campos de la gestión, la posibilidad de compartir información de diversas implementaciones de estos sistemas aún es un problema que aqueja a los usuarios, quienes desearían contar con la información y de paso consolidarla en un solo mapa con el fin de explotar al máximo el verdadero propósito de los Sistemas de Información geográficos.

Con el fin de dar una solución al problema planteado, a continuación se presenta el modelado de la aplicación, la cual se va a desarrollar como un sistema mediador. Dicha arquitectura fue escogida porque nos permite definir el dominio (Información geográfica) por medio de meta datos los cuales usamos como marco de referencia para el intercambio de información entre las múltiples fuentes de datos adscritas al sistema. Como se mencionó en el capitulo 2, el sistema mediador utiliza una arquitectura de 3 niveles: aplicación cliente, mediador y wrapper. Para la integración de dichos módulos, se va a hacer uso de la plataforma de Servicios Web, como middleware de comunicación entre los diferentes componentes de la aplicación. Se decidió usar Servicios Web debido a la facilidad que tiene para transportarse sobre Internet, además de que permite un desarrollo rápido de aplicaciones distribuidas, haciendo completamente transparente para el desarrollador los procesos de comunicación y permitiendo que éste se preocupe únicamente de la lógica de la aplicación.

Es importante aclarar, que la aplicación fue desarrollada para el intercambio de información geográfica, sin embargo la arquitectura puede ser aplicada a cualquier dominio de información.

Desde la perspectiva del usuario final del sistema, podemos visualizar el diagrama de casos de uso de la siguiente manera:

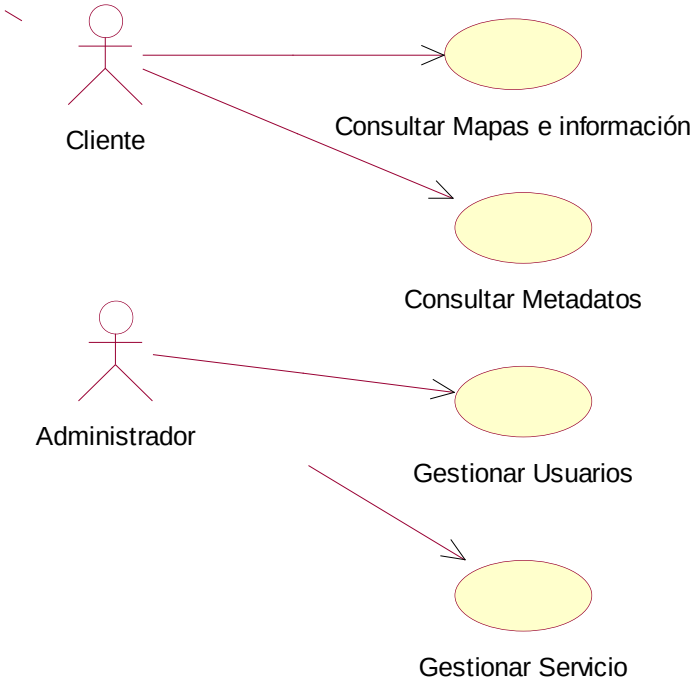



**FIGURA 3.1 DIAGRAMA DE CASOS DE USO DE LA APLICACIÓN**

Sin embargo, el llevar a cabo dichos casos de uso, implica dividir el sistema en tres subsistemas:

- Cliente
- Mediador
- Wrapper

En el presente capítulo se va a presentar el modelado del mediador y el wrapper, aunque de una manera simplificada, se deja para los anexos la documentación completa del mismo y el modelado del cliente.

**Arquitectura Propuesta**

La figura muestra la arquitectura de la aplicación y la ubicación de cada uno de los elementos dentro del entorno.

**FIGURA 3.2 ARQUITECTURA DE LA APLICACIÓN**

Los componentes principales de esta arquitectura son:

**Mediador:** El mediador es un software implementado como Servicio Web encargado de analizar las peticiones de los clientes y determinar que SIG o SIG's pueden ofrecer una respuesta a la solicitud. Se encarga de la validación de usuarios, validación de las peticiones, la estructuración y ejecución de consultas, y la estructuración y retorno de respuestas al cliente. Posee también funciones que le permiten gestionar los wrappers adscritos al sistema.

**Internet:** La red Internet es una colección de redes soportadas en el protocolo TCP/IP, el cual hace posible que muchos sistemas compartan recursos e información entre sí, independientemente del sistema operativo, de la tecnología, las plataformas de aplicación o del medio de transmisión, haciendo posible a la vez la prestación de un gran número de servicios. Internet permite que muchos sistemas informáticos implementados sobre diferentes arquitecturas aprovechen el procesamiento distribuido y las tecnologías de la información para incrementar su potencialidad.

**Wrapper:** Es un software implementado como Servicio Web que permite la extracción de información descriptiva de las bases de datos de los sistemas de información geográfica, mapas en formato shp, y meta datos sobre la información disponible de un SIG.

**Cliente:** Con el fin de validar las potencialidades de la arquitectura propuesta, se desarrolló una aplicación cliente que accede al sistema mediador con el fin de consultar los mapas y la información disponibles en las distintas fuentes de datos adscritas al sistema. Fue desarrollado como una aplicación Web a base de JSPs con una interfaz SOAP que le permite acceder al mediador implementado como servicio Web.

## 3.1 MODELADO SISTEMA MEDIADOR

### 3.1.1 Descripción Narrativa de la Aplicación

Se va a desarrollar un sistema mediador que permita la consulta de información a multiples fuentes de información geográfica. Dentro de la información a consultar encontramos mapas en formato shp, tablas de datos en formato dbf y en el caso de

que existan, datos contenidos en bases de datos descriptivas. Cada usuario puede desarrollar los clientes de acuerdo a sus necesidades específicas, ya que el sistema está en la capacidad de ofrecer la informacion necesaria para la construccion de los mismos a traves del documento WSDL, que estara disponible en un servidor de registro UDDI o en un Servidor Web.

Para el sistema existen cuatro tipos de actores:

**U_Administrador:** Se encarga de la gestión del sistema mediador realizando tareas como la actualización del repositorio de metadata y la agregación de nuevas fuentes de datos al sistema.
**U_cliente:** Usuario que sólo puede realizar consultas de información, ya sea de datos o de metadatos.
**Wrapper:** Es el encargado de responder a las consultas realizadas por el sistema mediador. Es un sistema independiente que le presta servicios al sistema mediador, entre los que encontramos la consulta de información y la consulta de metadatos.
**UDDI:** Es un servidor de registro en el que se almacenan los documentos WSDL con la descripción técnica de la implementación de las fuentes de datos. El sistema mediador lo consulta para obtener los documentos WSDL. Sin embargo , los documentos WSDL pueden tambien ser obtenidos de un Servidor Web, no necesariamente de un registro UDDI.

Cuando un usuario U_Administrador accede a la URL del sistema mediador, se le presenta una interfaz que le pide su login y password para el ingreso al sistema. Una vez que el usuario administrador sea validado se le presenta otra interfaz que le permite realizar las funcionalidades para la **Gestión Fuentes** y la **Gestión de usuarios.**

Dadas las características de Servicio Web del sistema , no se tendrá en cuenta para el presente modelado al U_Cliente, solo se dirá que éste hace uso de los servicios ofrecidos por el sistema, independientemente de la forma en que lo haga, ya que esta en la libertad de implementarse de acuerdo a sus necesidades. Los actores Wrapper y UDDI no inician ningun servicio del sistema sino que son involucrados ante peticiones de los otros dos actores (U_Administrador y U_Cliente).

### 3.1.2 Modelo de Casos de Uso del Sistema Mediador

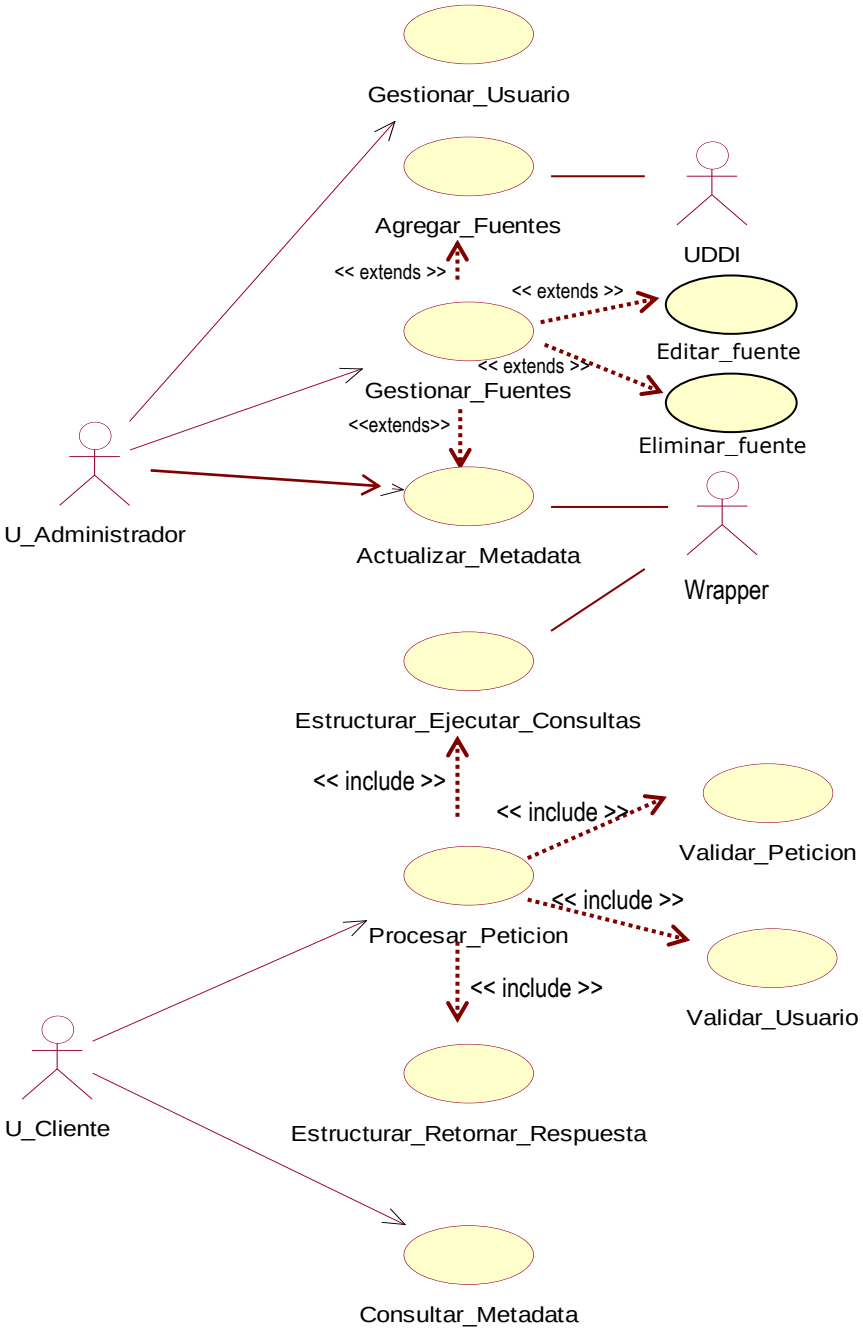



**FIGURA 3.3 DIAGRAMA DE CASOS DE USO DEL SISTEMA MEDIADOR**

**Caso de Uso:** Gestionar Usuario.

**Actores:** U_Administrador (Inicador).

**Propósito:** Gestionar la información de los usuarios del sistema.

**Resumen:** El caso de uso se inicializa cuando el U_Administrador elige la opción Gestionar usuario. El sistema le presenta las opciones Crear usuarios, Actualizar Información de Usuario, Definir Privilegios y Restricciones y Eliminar usuarios.

**Prototipo de GUI**

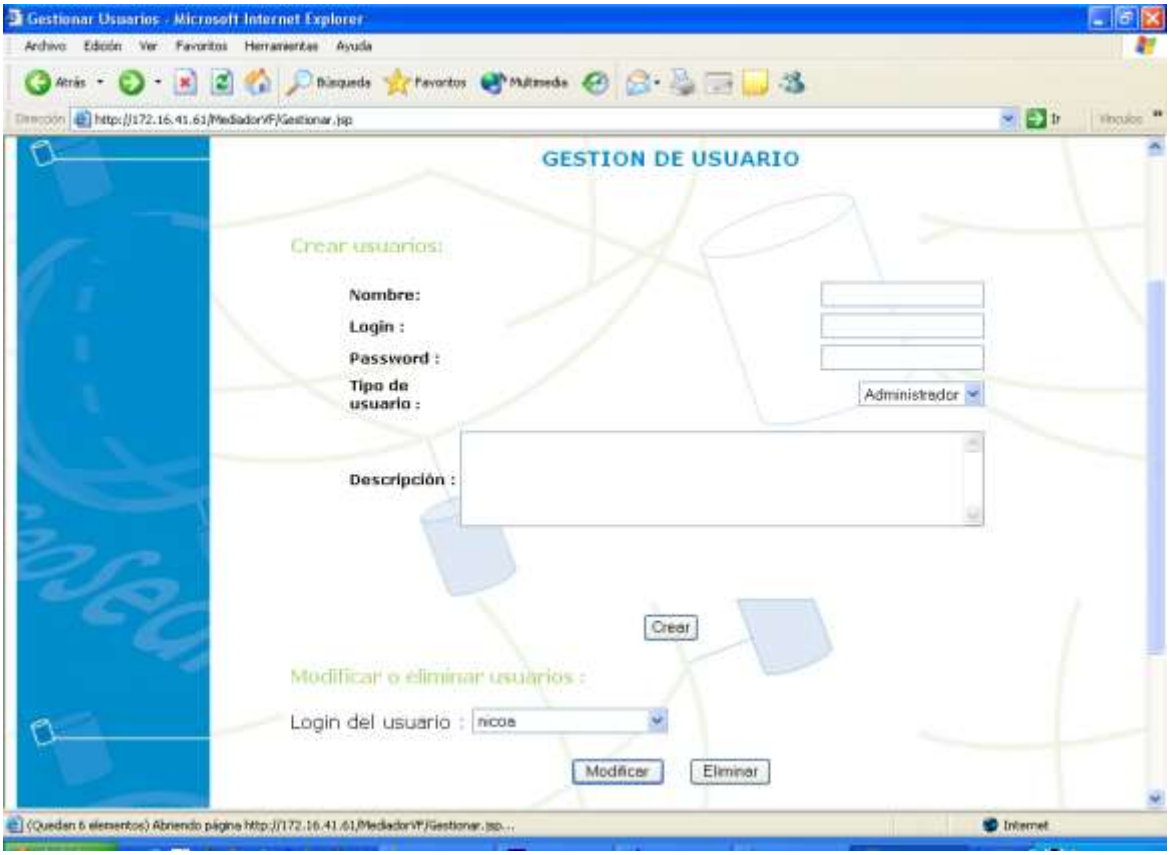



**FIGURA 3.4 INTERFAZ DE GESTIÓN DE USUARIOS**

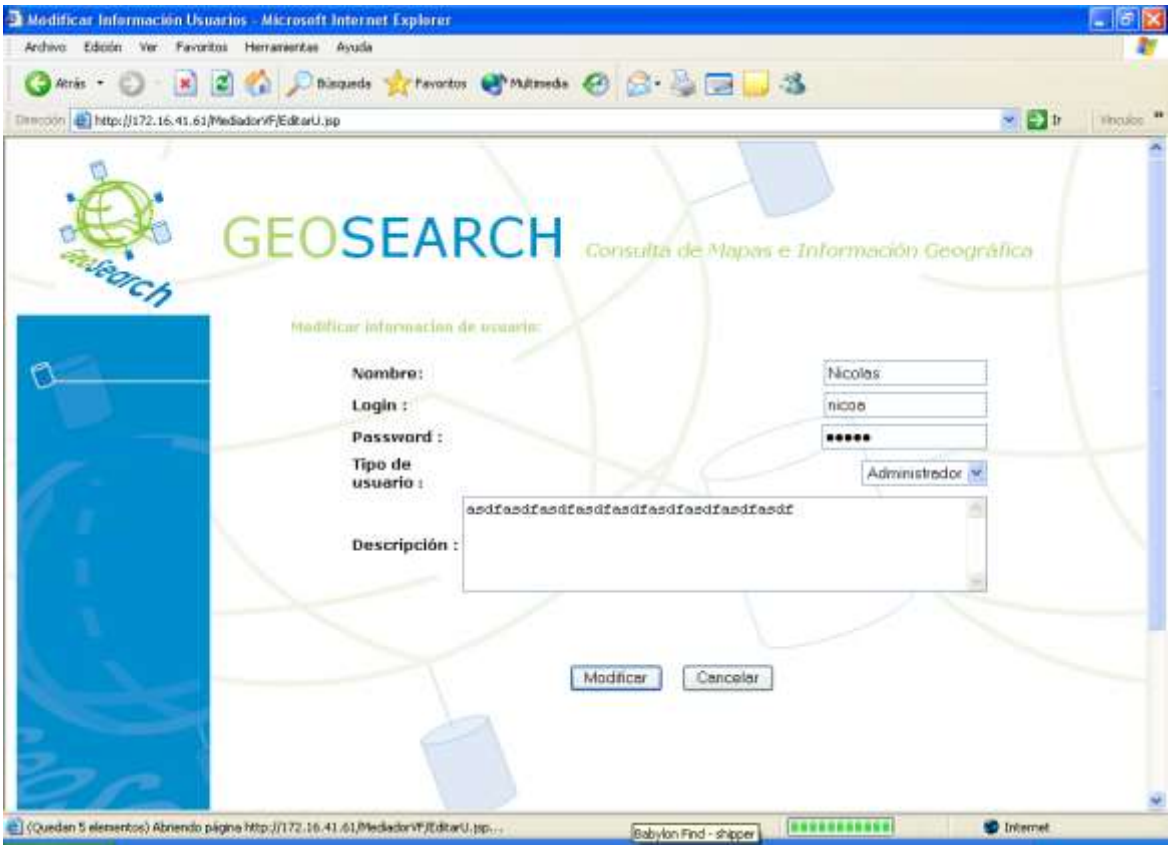



**FIGURA 3.5 INTERFAZ MODIFICAR USUARIO**

**Caso de Uso:** Gestionar fuentes

**Actores:** U_Administrador (Inicador).

**Propósito:** Gestionar las fuentes de datos adscritas al sistema.

**Resumen:** El caso de uso se inicializa cuando el U_Administrador elige Gestionar Fuentes. El U_Administrador tiene las de opciones de Agregar Fuente, Editar Fuentes, Eliminar Fuente, Actualizar Metadata.

**Prototipo de GUI**


**FIGURA 3.6 INTERFAZ DE GESTIÓN DE FUENTES**

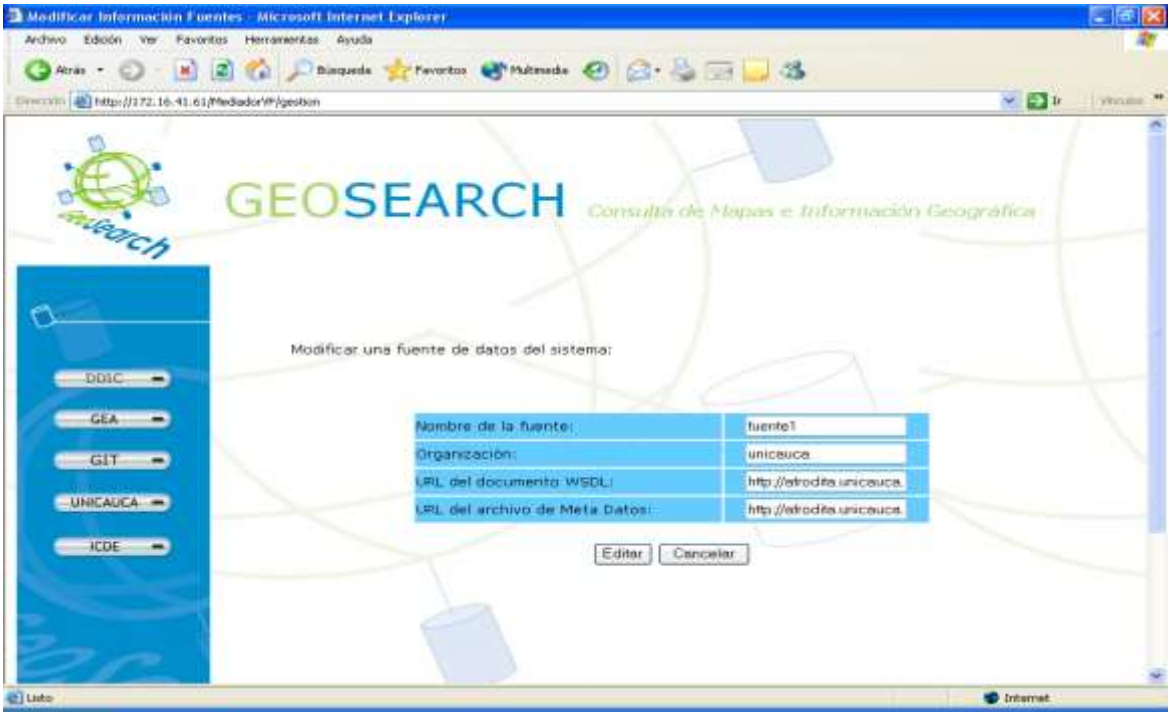



**FIGURA 3.7 INTERFAZ EDITAR FUENTE**

**Caso de Uso:** Actualizar Metadata (Extiende del caso de uso Gestionar Fuentes).

**Actores:** U_Administrador (Iniciador).
**Propósito:** Actualizar el repositorio de metadata.
**Resumen:** El caso de uso se inicializa cuando el U_Administrador selecciona una fuente de datos y elige Actualizar Metadata. Entonces, el sistema se comunica con la fuente seleccionada y se realiza la actualización.

**Caso de Uso:** Procesar petición

**Actores:** U_Cliente (Iniciador).
**Propósito:** Procesar las peticiones realizadas por los clientes.
**Resumen:** El caso de uso se inicializa cuando un cliente realiza una petición al sistema mediador. Se realizan los procesos de validación del usuario y de la petición , la estructuración y ejecución de consultas y la estructuración de respuestas con el fin de poder entregarle al cliente un formato que este pueda interpretar. Incluye los casos de uso validar usuario, validar petición, Estructurar y ejecutar consultas y estructurar y retornar respuesta.

**Prototipo de GUI**
*Este caso de uso no presenta una interfaz gráfica ya que ésta depende de la implementación del cliente.*

**Caso de Uso:** Consultar Metadata

**Actores:** U_Cliente (Inicador).
**Propósito:** Consultar los metadatos de las fuentes adscritas al sistema.
**Resumen:** Los clientes tienen la opción de consultar los metadatos de las fuentes adscritas al sistema mediador con el fin de tener un conocimiento anticipado de la información disponible.

**Prototipo de GUI**
*Este caso de uso no presenta una interfaz gráfica ya que ésta depende de la implementación del cliente.*

### 3.1.3 Diagrama de clases del Sistema Mediador

Por el tamaño del diagrama , este se encuentra en el documento diagramas modelado, esto con el fin de no dañar la paginación y formato del documento

**FIGURA 3.8 DIAGRAMA DE CLASES DEL SISTEMA MEDIADOR**

A continuación hacemos una descripción de las clases del diagrama de clases:

**Control_Servicio:** Clase encargada de controlar el procesamiento de las peticiones realizadas por los clientes. Invoca los procesos de validación de usuarios, validación de peticiones, ejecución de peticiones y estructuración de respuestas.

**Gestion_Metadata:** Esta clase se encarga de todos los procesos de gestión referentes a los metadatos del sistema. Entre estos procesos encontramos el parseo de los documentos de metadatos de cada fuente, así como del documento WSDL que describe los servicios de cada una de ellas. También genera los documentos de metadatos para los clientes.

**Gestion_Petición**: Se encarga de gestionar las peticiones de los clientes. Realiza la validación de las mismas determinando que fuentes contienen información que la puedan resolver.

**BeanFuente:** Este bean permite la gestión la información de las fuentes adscritas al sistema.

**BeanConector:** Este bean permite la conexión con la base de datos del sistema y la ejecución de consultas sobre ella.

**BeanUsuario:** Este bean permite la gestión de la información relacionada con los usuarios del sistema.

**Consultor:** Permite la consulta de las diferentes fuentes adscritas al sistema y formatea las respuestas de acuerdo a la solicitud del cliente.

**ClienteWS:** Clase que realiza las peticiones a los Servicios Web de las fuentes de datos.

**CreadorDBF:** Clase encargada de la creación de un dbf a partir de varias fuentes formateadas en HashMaps.

**Archiver:** Clase encargada del manejo de archivos a alto nivel.

**Stub:** Clase Stub de los servicios web de las fuentes de datos sirve para hacer peticiones a cualquier fuente de datos.

**Compresor:** Clase que permite la descompresión de un archivo en formato Zip.

Para un mayor detalle de las clases, sus atributos y métodos, remítase al anexo B, Implementación del software – Sistema Mediador.

### 3.1.4 Diagrama de Paquetes de Diseño del Sistema Mediador

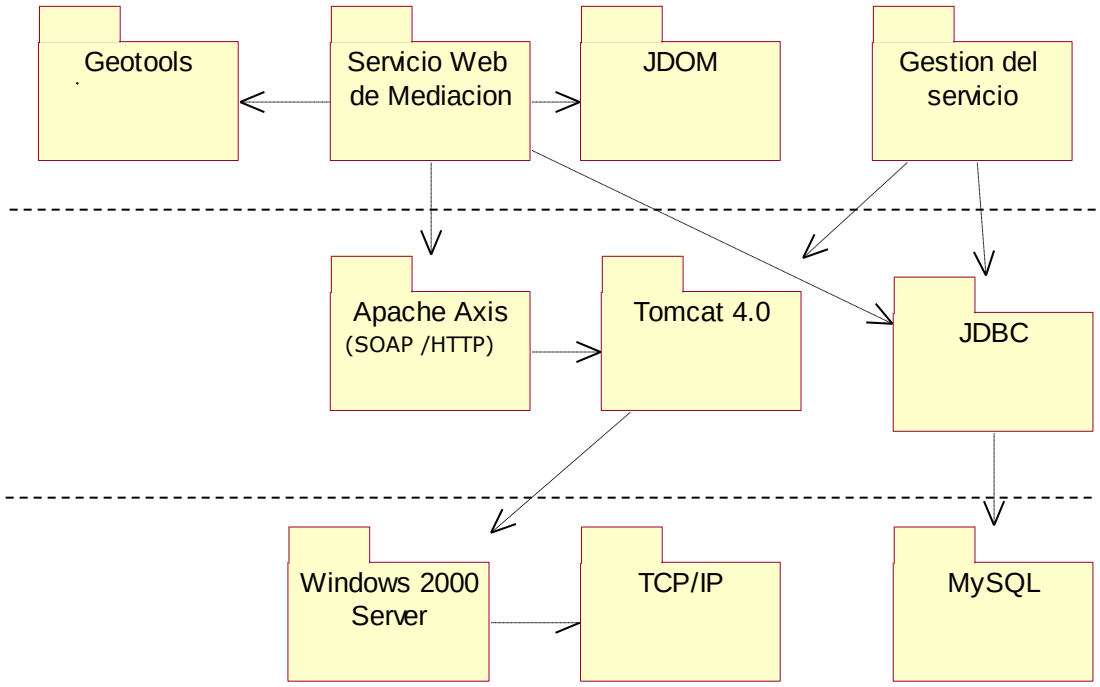



**FIGURA 3.9 DIAGRAMA DE PAQUETES DE DISEÑO**

EL paquete *Servicio Web de mediación* reúne las clases que contienen la lógica del sistema mediador .

El paquete *Gestión del servicio* contiene las interfaces y lógica que permite la gestión del sistema mediador a través de la Web. Esta construido a base de servlets y JSPs.

*GeoTools* es un conjunto de herramientas de java accesible a código fuente, que sirve para desarrollar mapas geográficos interactivos. Para el caso especifico de esta aplicación el API es usada para modificar los archivos de extensión dbf con el fin de adicionarle la información obtenida a partir de las múltiples fuentes de datos (para mayor información remítase al anexo G - Geotools).

Apache Axis es una implementación de la especificación SOAP. Es usada para la construcción y el despliegue del sistema mediador como un Servicio Web.

MySQL es el motor de base de datos donde se encuentra almacenada la información asociada al sistema mediador, como la información de las fuentes de datos (general y la obtenida a partir de los documentos XML) y la información sobre los usuarios del sistema.

JDOM es un API java que permite la creación, lectura y modificación de documentos XML. En esta aplicación es usada para la lectura de los documentos XML obtenidos de las fuentes de datos para su posterior mapeo en la base de datos y para la creación de los documentos XML que se entregan al cliente, que contienen la información que puede ser solicitada por estos.

Tomcat es un servidor Web que incluye un contenedor de servlets y JSPs. Es usado para el despliegue del sistema mediador como servicio Web y para montar la aplicación de Gestión de dicho sistema.

JDBC es el API java que permite el acceso al motor de base de datos MySql (para el caso especifico de nuestra aplicación). Es usado tanto por el Servicio Web de mediación para la validación de los clientes y las peticiones y por la Gestión del servicio para ejecutar las consultas de actualización de la información del sistema en la base de datos.

### 3.1.6 Diagrama Lógico de Datos – Sistema Mediador

Para la definición de las entidades del diagrama entidad relación se partió del documento XML que describe los meta datos obtenidos de cada una de las fuentes de datos adscritas al sistema. Este documento tiene la siguiente estructura:

**Estructura XML**

```
<?xml version="1.0" encoding="UTF-8"?>
<Fuente>
 <Mapas>
  <zona nombre="Purace">
   <Tema nombre="cartografia">
    <Mapa nombre="cuencas" indice="Layer" />
   </Tema>
   <Tema nombre="zonas_vida">
    <Mapa nombre="Zonas_de_vida" indice="Zonas-vida" />
   </Tema>
   <Tema nombre="vias">
    <Mapa nombre="vias" indice="Layer" />
   </Tema>
   <Tema nombre="vereda">
    <Mapa nombre="Veredas" indice="Nom_vereda" />
   </Tema>
  </zona>
 </Mapas>
 <Informacion>
  <zonaI nombre="Purace">
   <TemaI nombre="vereda">
    <EntradaI nombre="rios" parametro="nombre">
     <salida>Nombre_Zona_Vida</salida>
     <salida>nombre</salida>
    </EntradaI>
   </TemaI>
   <TemaI nombre="zonas_vida">
    <EntradaI nombre="id”    nformaci=”Nombre”>
     <salida>Nombre</salida>
     <salida>Noanimales</salida>
     <salida>rios</salida>
    </EntradaI>
   </TemaI>
  </zonaI>
 </Información>
</Fuente>
```

Tomando como referencia este documento, y teniendo en cuenta que la información contenida en él debía ser mapeada en la base de datos se definió una entidad por cada uno de los elementos contenidos en el documento y se definieron las relaciones entre las entidades:

- Una fuente posee varias zonas.
- Una zona posee varios temas
- Un tema posee varios mapas
- Un tema contiene varias entradas de información
- Una entrada de información contiene varias salidas de información

Finalmente, se añadió una entidad usuario que representa la información relacionada a éste. De esta manera, y teniendo en cuenta las anteriores relaciones, se obtuvo el siguiente el siguiente Diagrama Entidad Relación.

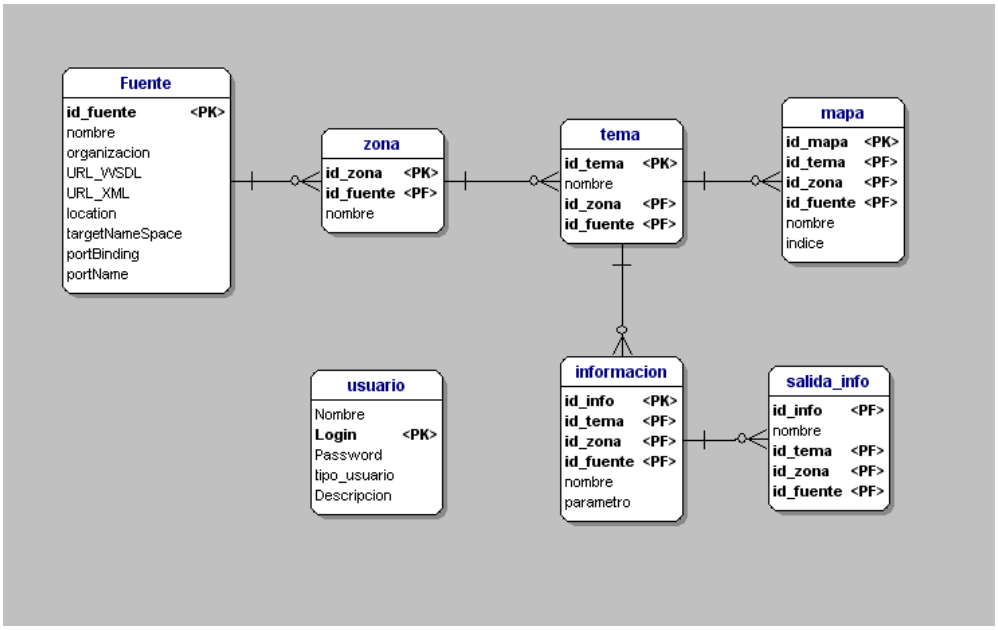



**FIGURA 3.10 DIAGRAMA LÓGICO DE DATOS**

## 3.2 MODELADO DEL WRAPPER

### Descripción narrativa

El sistema de encapsulamiento de fuente de datos o Wrapper, realiza la traducción entre las solicitudes hechas en el formato del mediador al formato propietario de la fuente de datos que encapsula. La información expuesta se categoriza en divisiones que faciliten su interoperabilidad con el sistema mediador en zonas, y cada zona se divide en temas, dentro de cada uno de los temas el administrador del Wrapper puede publicar los campos de su base de datos con nombres representativos del tema; de esta forma el administrador puede seleccionar los campos y tablas que se desea poner a disposición de los usuarios del mediador. Por último el sistema permite configurar los permisos de acceso a cada uno de los temas del Wrapper.

El sistema puede responder a solicitudes del mediador a la base de datos, así como a solicitudes de los metadatos de la fuente de datos, los resultados se entregan al mediador en el formato establecido previamente por el protocolo del sistema mediador. Por otro lado el sistema también responde a solicitudes para obtener los mapas que se encuentren guardados localmente.

Una vez que se realice una petición de los metadatos por parte del sistema mediador se devuelve un documento que describe la estructura de los datos del sistema de información geográfica. Por otro lado cada vez que se envía una sentencia para su resolución el wrapper verifica que el usuario tenga los permisos necesarios para el acceso a las tablas involucradas y devuelve los datos que sean posibles y los mensajes de error en el caso de que existan.

Los actores del sistema son los siguientes:

**Sistema Mediador:** Es el sistema que realiza las solicitudes tanto de datos como de metadatos al sistema de encapsulamiento de fuentes de datos.

**Administrador:** Usuario del sistema encargado de la configuración del wrapper y de la publicación de la base de datos en el mismo.

### 3.2.1 Diagrama de casos de Uso del Sistema Wrapper

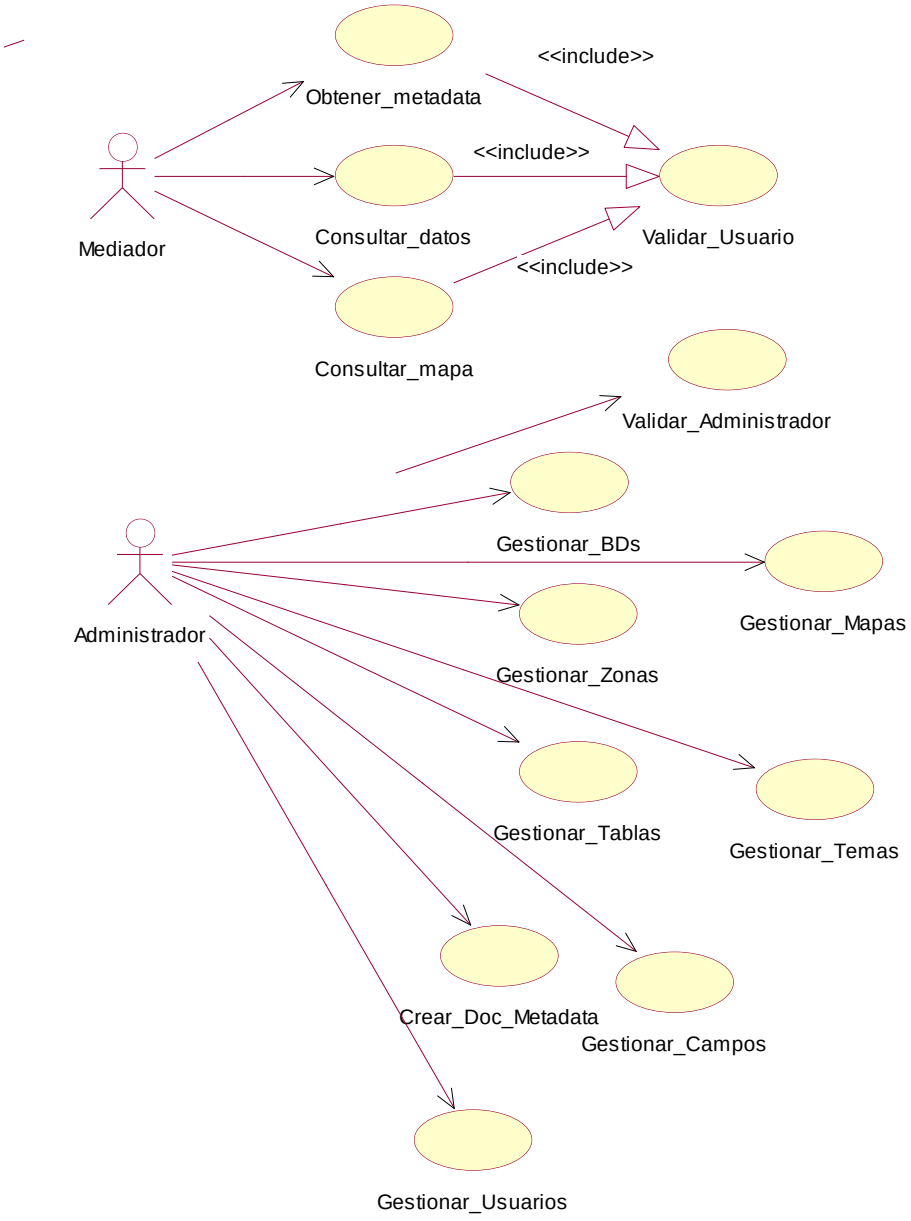



**FIGURA 3.11 DIAGRAMA DE CASOS DE USO DEL SISTEMA WRAPPER**

### 3.2.2 Descripción de escenarios

El sistema Wrapper o Sistema encapsulador de bases de datos se puede dividir para fines de su comprensión general en dos subsistemas, el primero corresponde a la lógica que da soporte a la prestación del Servicio Web y que se ha denominado Subsistema de interacción con el mediador, mientras que el segundo subsistema corresponde a la parte de administración que se constituye como una aplicación Web independiente.

Para la descripción de los casos de uso de la aplicación para el primer subsistema se utilizaron; diagramas de secuencia y de clases, los cuales especifican adecuadamente las aplicaciones estándar, mientras que para la aplicación web de administración se realizaron los diagramas de clase utilizando los estereotipos web, además de mostrar las interfaces de los casos de uso.

#### 3.2.2.1 Subsistema de interacción con el mediador

***Funciones del sistema***

1. **Obtener meta datos**: Brinda al mediador la opción de conocer de antemano los datos con los que cuenta la fuente de datos a través de un documento que los describe.

2. **Realizar Consulta**: es una función por medio de la cual se ejecuta una petición en el sistema de gestión de base de datos y se devuelven los resultados o los mensajes que indican error.

***Casos de uso***

**Caso de Uso:** Obtener metadatos
**Actores:** Mediador (iniciador)
**Propósito:** Permite realizar la actualización de los metadatos en el mediador.
**Resumen:** El caso de uso se inicializa cuando el mediador envía una solicitud de que se envíe el archivo de metadatos, El Wrapper realiza la consulta de dicho archivo actualizado y lo retorna al mediador.

**Caso de Uso:** Ejecutar Consulta

**Actores:** Mediador (iniciador)

**Propósito:** Permite realizar la ejecución de las solicitudes.

**Resumen:** El caso de uso se inicializa cuando el mediador envía una solicitud compuesta por una lista de los mapas o los parámetros que se quieren obtener; el Wrapper realiza la búsqueda de acuerdo con los datos de los usuarios validos en las bases de datos de la fuente y devuelve los resultados en un formato adecuado para el mediador.

#### 3.2.2.2 Subsistema de Administración

***Funciones del sistema***

1. **Verificar permisos**: de acuerdo con el usuario que esté accediendo al wrapper se verifican los permisos para el acceso a las tablas.
2. **Gestionar Bases de datos a Publicar**: Cada una de las bases de datos que componen la fuente de datos que encapsula el Wrapper, puede publicarse independientemente, y dentro de cada base de datos se pueden publicar ciertos campos o ciertas tablas, lo que hace esta función es permitir seleccionar que elementos de las bases de datos se publicarán
3. **Publicar Mapas**: Además de la información de las bases de datos se pueden publicar los mapas para su compartición.
4. **Crear Zonas y temas:** El sistema permite crear estas divisiones generales para su posterior consulta. Las zonas están directamente relacionadas con las zonas geográficas sobre las que se tengan datos en la base de datos, los temas son subdivisiones dentro de las zonas que indican un eje temático dentro de la zona ej: dentro de la zona Colombia puede haber un tema ríos y otro división política.
5. **Administrar usuarios:** El sistema permite crear, eliminar y asignar permisos a los usuarios a determinados temas.
6. **Creación documento Meta Datos:** El administrador del sistema puede determinar cual información será contenida en el documento de meta datos de la fuente que será el que se exporte al mediador para que valide sus consultas y conozca de sus capacidades.

***Casos de Uso***

**Caso de Uso:** Validar Usuario

**Actores:** Casos de uso ejecutar consulta y obtener metadatos (iniciador)

**Propósito:** valida las solicitudes.

**Resumen:** El caso de uso se inicializa cuando se recibe una solicitud para verificar permisos sobre una tabla o un campo y se verifica de acuerdo a los permisos si todos los campos están en tablas para las cuales tiene acceso el usuario determinado.

**Caso de Uso:** Validar Administrador Wrapper

**Actores:** Administrador (iniciador)

**Propósito:** Permite la validación del administrador en el sistema

**Resumen:** El caso de uso se inicia cuando el administrador intenta entrar al sistema. El sistema permite ingresar login y password y verifica si puede acceder o no, en caso de tener permisos muestra la interfaz principal y en otro caso, muestra el mensaje de error.

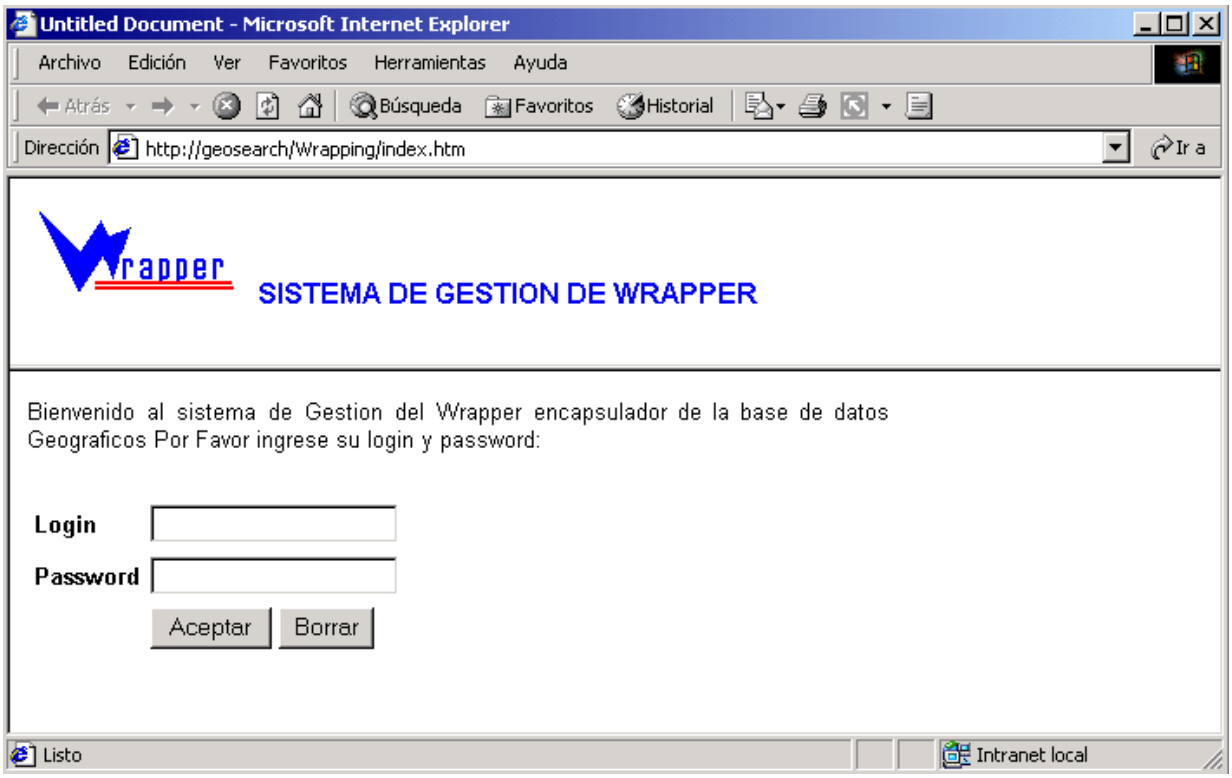



**FIGURA 3.12 VALIDAR ADMINISTRADOR**

**Caso de Uso:** Gestionar usuario Wrapper

**Actores:** Administrador Wrapper (iniciador)

**Propósito:** Permite gestionar los usuarios del sistema Wrapper.

**Resumen:** El caso de uso se inicia cuando El administrador selecciona Gestionar Usuario. El sistema le presenta las opciones de Crear Usuarios, Editar información de los usuarios, y eliminarlos del sistema.

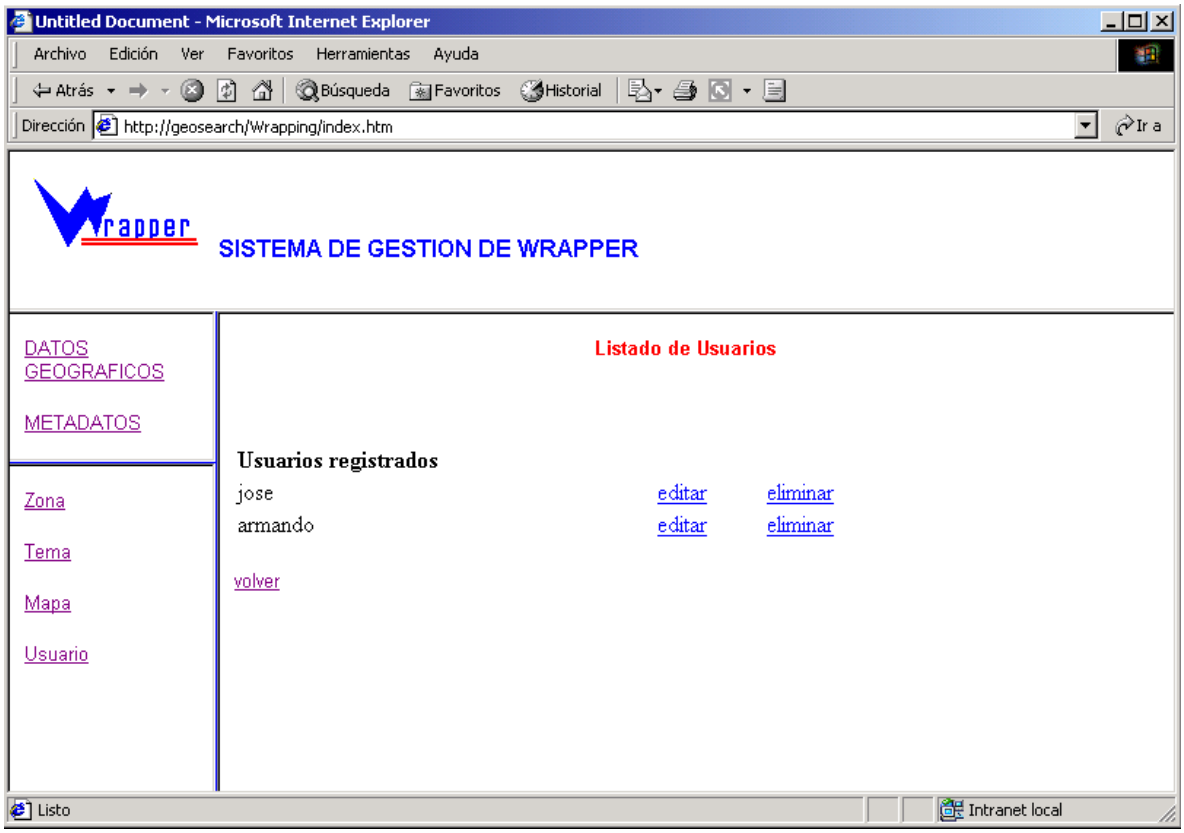



**FIGURA 3.13 GESTIONAR USUARIOS DEL SISTEMA**

**Caso de Uso:** Gestionar BDs

**Actores:** Administrador Wrapper (iniciador)

**Propósito:** Permite gestionar las bases de datos que se van a publicar.

**Resumen:** El caso de uso se inicia cuando El administrador selecciona Gestionar Bases de datos. El sistema le presenta las opciones de Agregar Bases de datos, Editar parámetros de la conexión a las bases de datos, y eliminarlas del sistema.

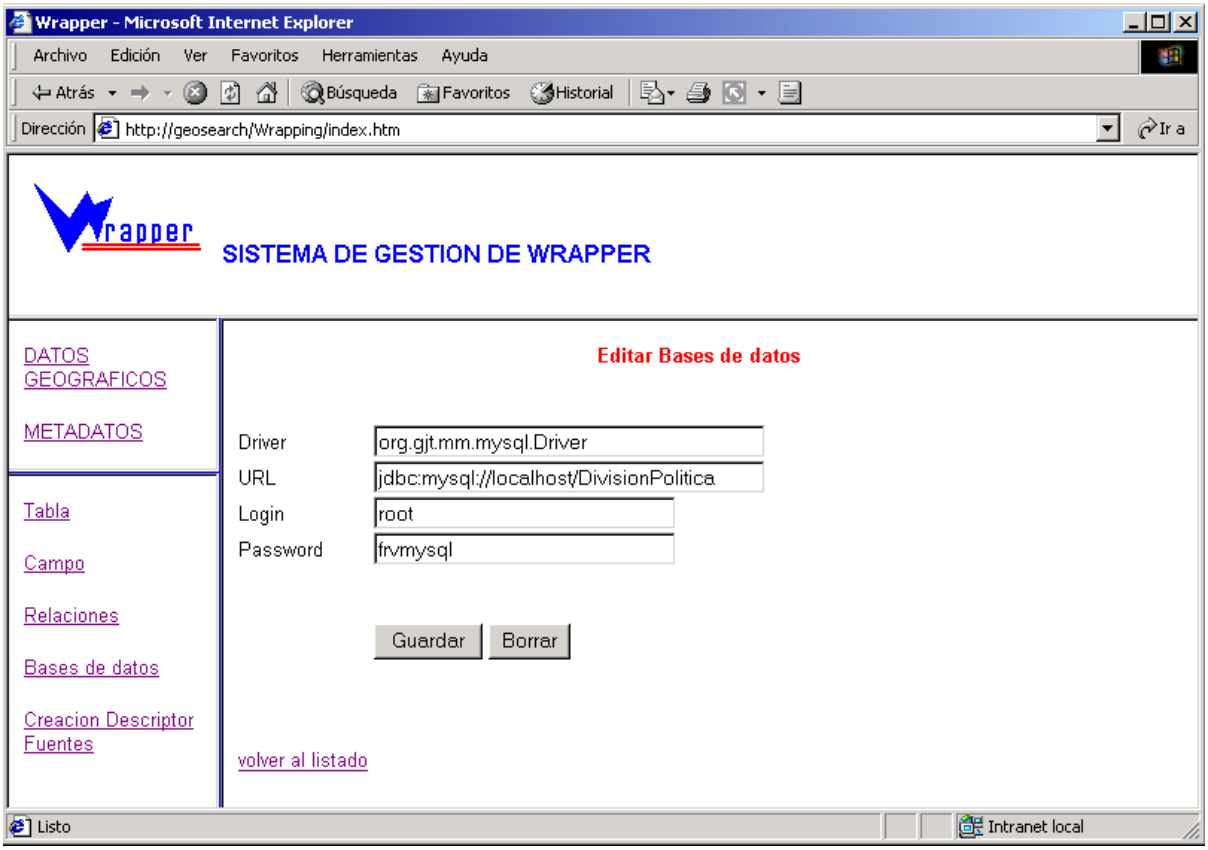



**FIGURA 3.14 GESTIONAR BASES DE DATOS**

**Caso de Uso:** Gestionar Zonas

**Actores:** Administrador Wrapper (iniciador)

**Propósito:** Permite gestionar las zonas definidas para la información geográfica del sistema Wrapper.

**Resumen:** El caso de uso se inicia cuando El administrador selecciona Gestionar Zonas. El sistema le presenta las opciones de Agregar Zonas, Editar información de la zona, y eliminarlas del sistema.

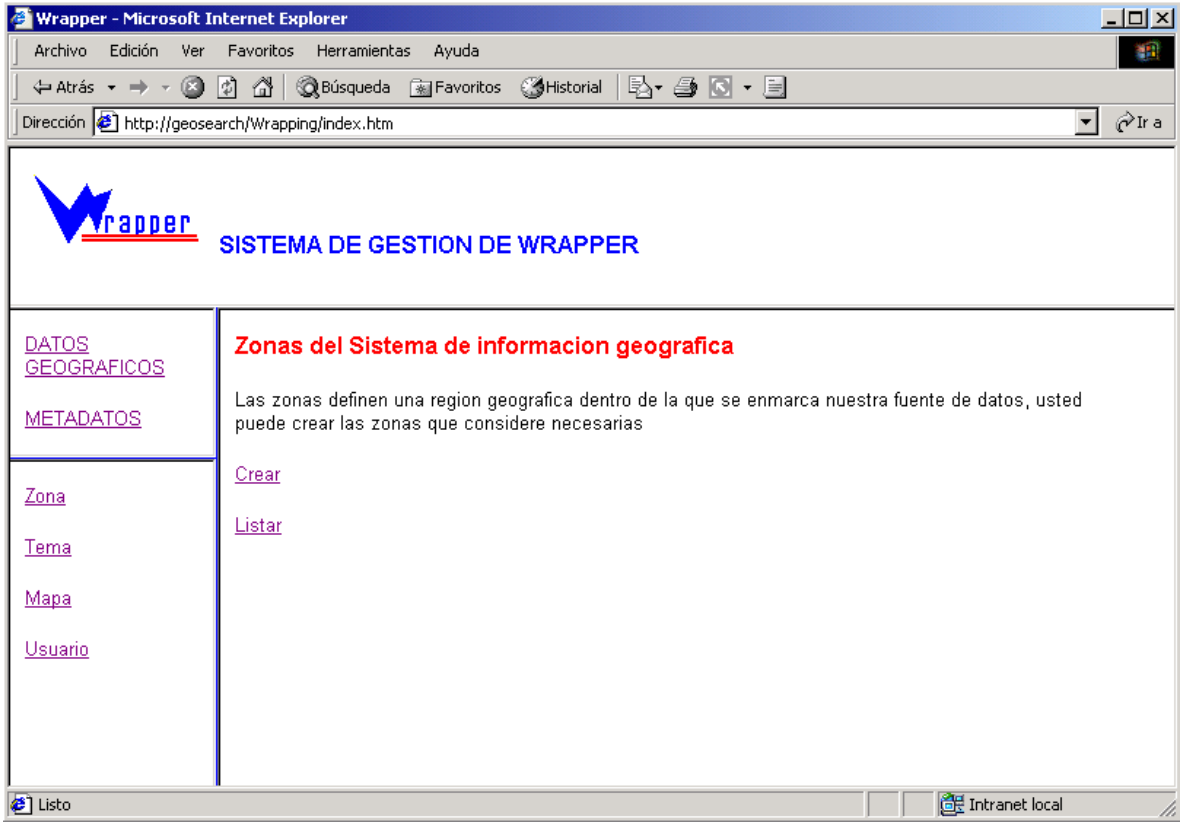



**FIGURA 3.15 GESTIONAR ZONAS**

**Caso de Uso:** Gestionar Temas

**Actores:** Administrador  Wrapper (iniciador)

**Propósito:** Permite gestionar las zonas definidas para la información geográfica del sistema Wrapper.

**Resumen:** El caso de uso se inicia cuando El administrador selecciona  Gestionar Temas. El sistema le presenta las opciones  de Agregar Temas, Editar información del tema,  y eliminarlas del sistema.

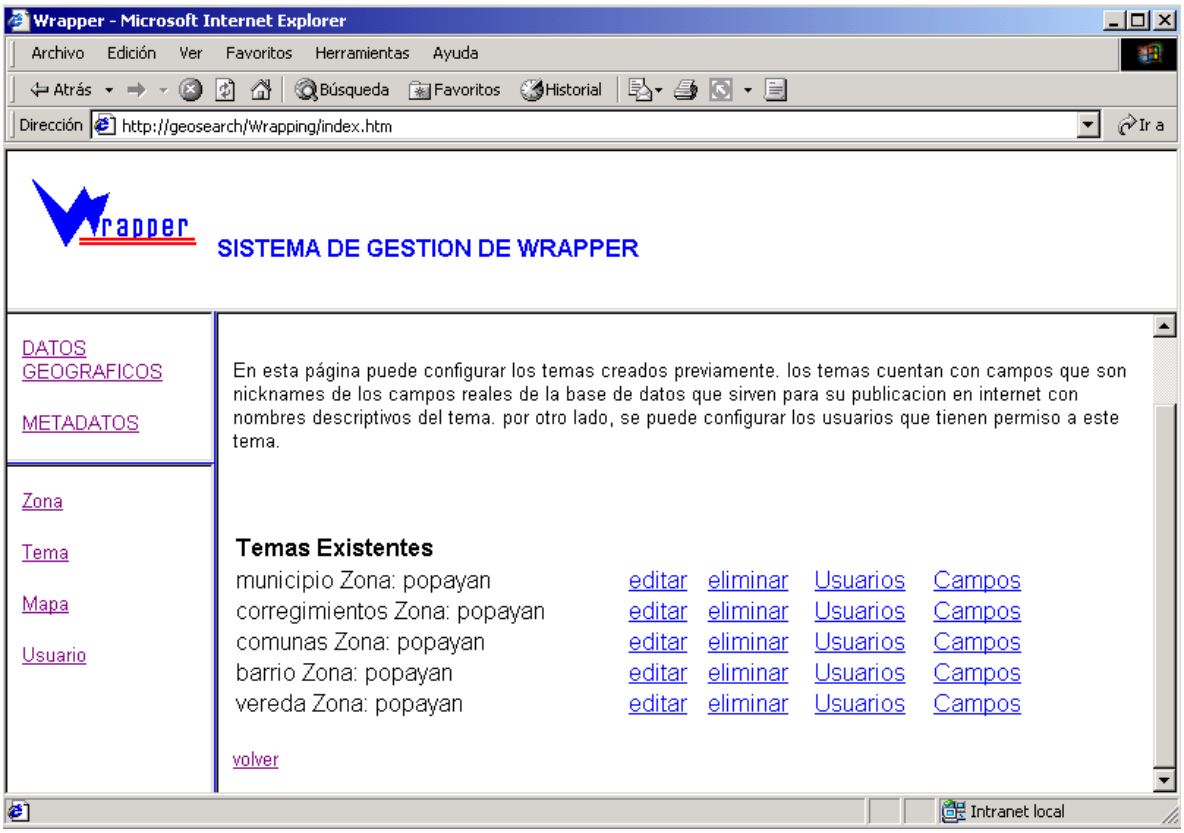



**FIGURA 3.16 GESTIONAR TEMAS**

**Caso de Uso:** Crear_Doc_Metadata

**Actores:** Administrador del wrapper(iniciador)

**Propósito:** Permite seleccionar dentro de los elementos registrados en el Wrapper, cuales serán publicados en el documento de metadatos que describirá la fuente en el mediador

**Resumen:** El caso de uso se inicia cuando el Administrador selecciona crear documento de Metadatos, El sistema le permite seleccionar dentro de los elementos registrados en el Wrapper, cuales serán publicados en el documento de metadatos que describirá la fuente en el mediador, y le muestra al administrador las opciones para agregar al documento descriptor los mapas y los campos de las bases de datos que desea publicar.

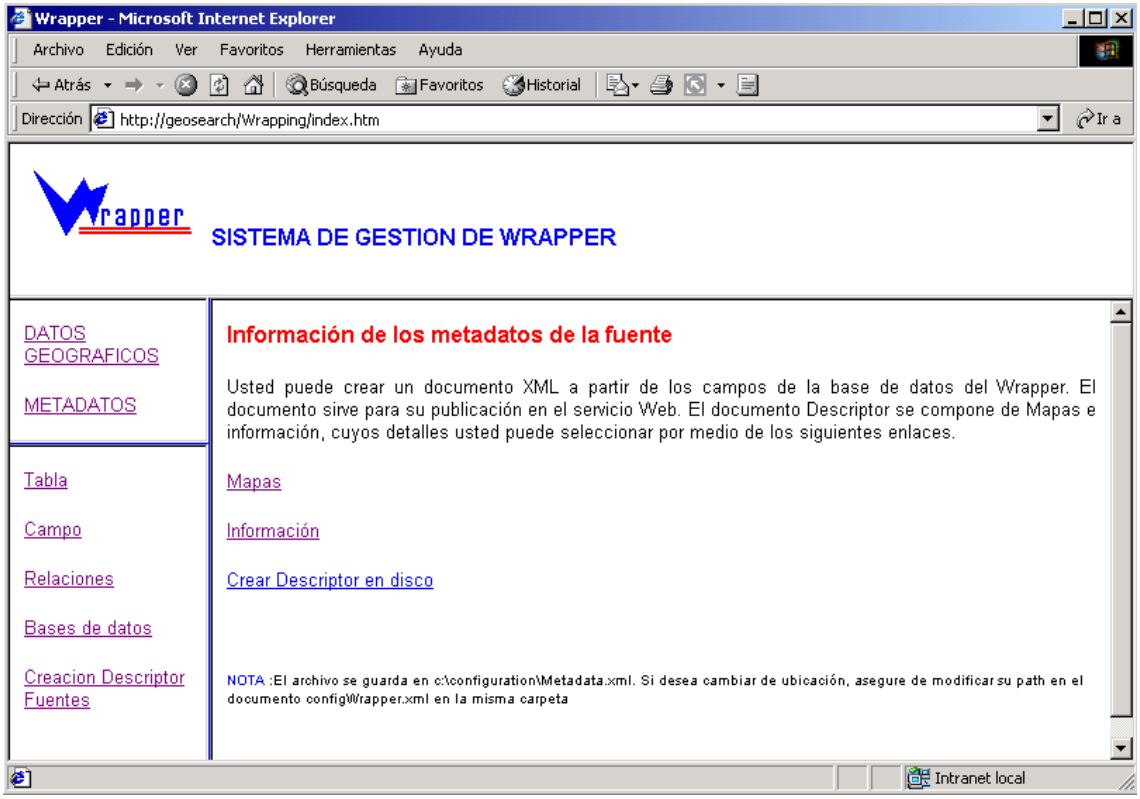



**FIGURA 3.17 CREAR DOCUMENTO DE METADATOS**

**Caso de Uso:** Gestionar Tablas

**Actores:** Administrador  Wrapper (iniciador)

**Propósito:** Permite gestionar las tablas definidas para la información geográfica del sistema Wrapper.

**Resumen:** El caso de uso se inicia cuando El administrador selecciona  Gestionar Tablas. El sistema le presenta las opciones  de Agregar Tablas, Editar información de la tabla,  y eliminarlas del sistema.

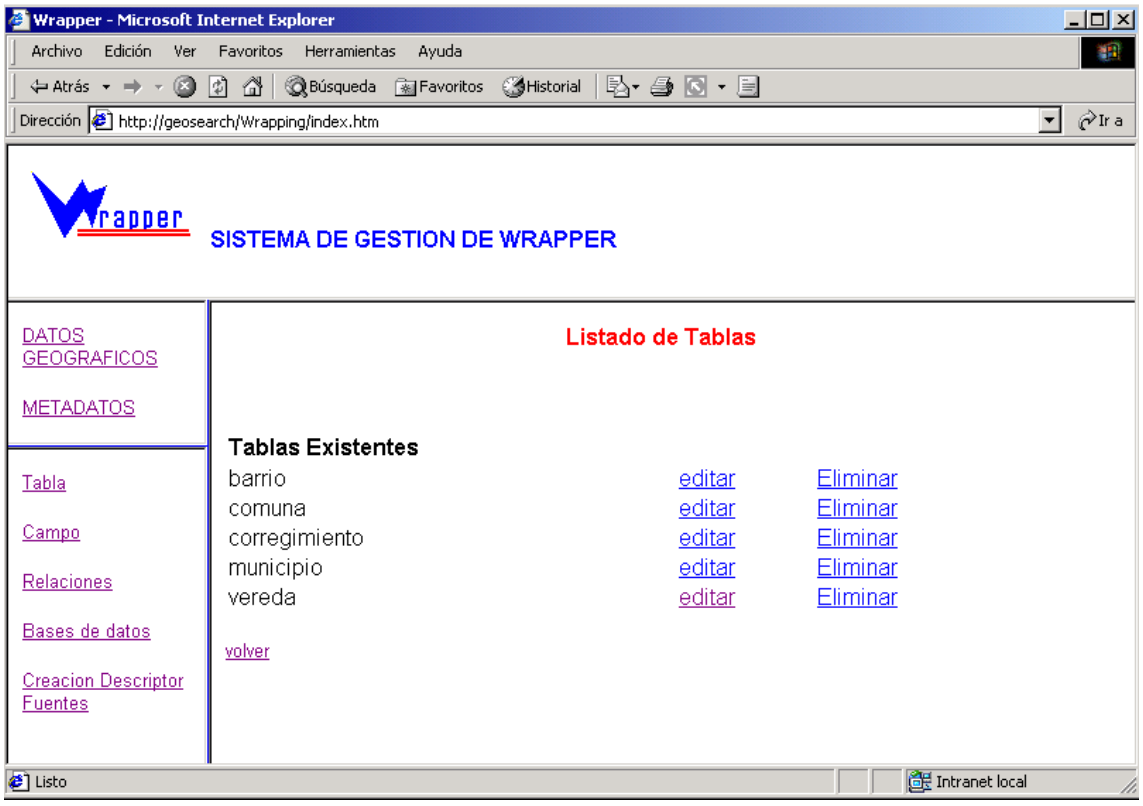



**FIGURA 3.18 GESTIONAR TABLAS**

**Caso de Uso:** Gestionar Campos

**Actores:** Administrador Wrapper (iniciador)

**Propósito:** Permite gestionar los campos definidos para la información geográfica del sistema Wrapper.

**Resumen:** El caso de uso se inicia cuando El administrador selecciona Gestionar Campos. El sistema le presenta las opciones de Agregar Campos, Editar información del campo, y eliminarlos del sistema.

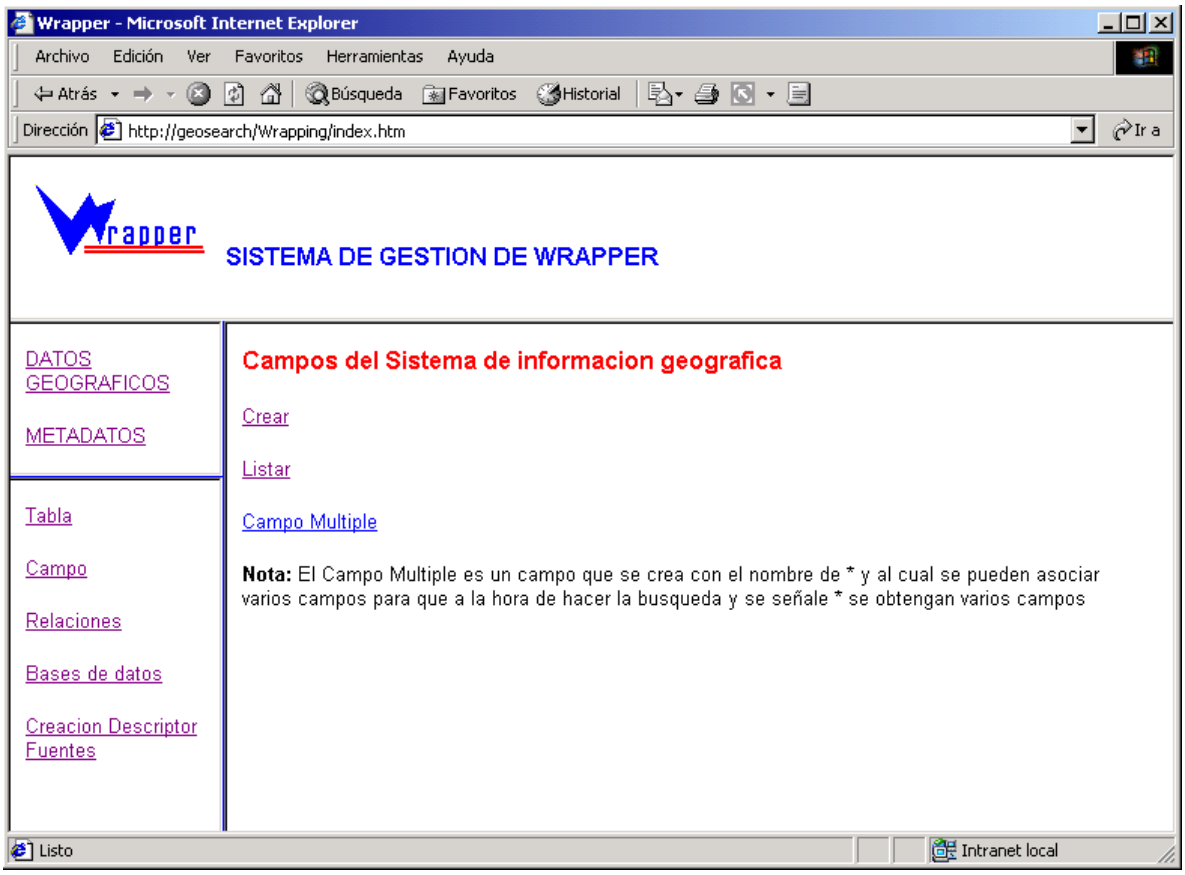



**FIGURA 3.19 GESTIONAR CAMPOS**

### 3.2.3. Diagramas de clases del Sistema Wrapper

Por el tamaño del diagrama , este se encuentra en el documento diagramas modelado, esto con el fin de no dañar la paginación y formato del documento

**Figura 3.20 DIAGRAMA DE CLASES DEL SISTEMA WRAPPER**

En el diagrama de clases se puede apreciar mas detalladamente las clases que intervienen en el sistema Wrapper:

- **WrapperWS:** Clase encargada de la publicación del servicio Web
- **Wrapper:** Clase que realiza los procedimientos de encapsulamiento y formatea las respuestas agregando errores, esta clase recibe en las consultas los parámetros necesarios, para cada método los parámetros son diferentes:

  HashMap **Método getData**:

  String Usuario: usuario del sistema

  String Password : password del usuario

  String tema: Tema en el que se realiza la consulta

  Sring Zona: Zona en la que busca los datos

  Vector paramOut: Parámetros que se quieren buscar en la base de datos

  HashMap paramIn: Parámetros de búsqueda de la consulta

  HashMap **Método getMapa**:

  String Usuario: usuario del sistema

  String Password : password del usuario

  String tema: Tema en el que se realiza la consulta

  Sring Zona: Zona en la que busca los datos

  String mapa: nombre del mapa que se quiere buscar

  HashMap **Método getMapa**:

  String Usuario: usuario del sistema

  String Password : password del usuario

  Por su parte la respuesta se compone de un HashMap cuyas entradas(keys) dependen del **metodo**

  HashMap **Método getData**:

  "Metadata": Vector con un conjunto de HashMaps los cuales llevan la información de los campos devueltos

  Cada Hashmap tiene las siguientes entradas : Nombre, tipo, decimales, longitud

"Data" : String[][] con los datos devueltos de la base de datos, están organizados por columnas y filas

"Error" : Vector con cadenas describiendo cada uno de los errores presentados en el proceso

HashMap **Método getMapa**:

"Mapa": byte[] bytes del archivo del mapa en formato zip conteniendo : archivo shx, index(shx) y dbf.

"Error" : Vector con cadenas describiendo cada uno de los errores presentados en el proceso

HashMap **Método getMetadata**:

"Metadata": byte[] bytes del archivo de metadatos en formato xml.

"Error" : Vector con cadenas describiendo cada uno de los errores presentados en el proceso.

Si se presentan errores que impidan la terminación del proceso, todas las entradas se agregan al HashMap de respuesta con un valor null.

- **XML_RW:** Clase que se encarga de procesar los documentos XML, es útil para la clase Wrapper que lo usa para obtener los parámetros de configuración desde un archivo que es del sistema y que debe ser configurado antes del inicio del sistema wrapper.
- **Archiver:** Clase relacionada con todos los procedimientos de acceso a archivos locales, como para leer los bytes de los mapas, leer bytes del documento de metadatos, crear directorios temporales para comprimir y descomprimir archivos para la respuesta y eliminación de los mismos cuando son temporales.
- **Mapa:** Clase que permite la búsqueda de los mapas en el sistema, realiza la validación de los mapas en el tema y zona, obtiene los archivos copiándolos en un directorio temporal que luego es comprimido y constituye la respuesta.
- **Data:** Clase encargada de convertir los parámetros de búsqueda que vienen en el vector y el HashMap en una sentencia SQL que se ejecutará en la base de datos que según los datos almacenados en la base de datos del Wrapper tiene esta información. Los nombres de los campos que se exponen en el servicio web son seudónimos de los de la base de datos por lo tanto hay que realizar

una validación, conversión y consulta de las tablas previo a la ejecución de la misma sobre la fuente de datos. Por último la clase Data se encarga de dar formato al Resultset según los datos que maneja el mediador.

- **Zipper:** se encarga de los procesos de compresión y descompresión de archivos
- **Usuario:** se encarga de los procesos relacionados con los usuarios especialmente la validación que corresponde a revisar si el usuario tiene permiso al tema que desea consultar.
- **Campo:** Clase que representa cada uno de los campos que se están consultando, permite la validación del mismo en el tema.
- **ConectorBD:** Clase que se ocupa de todo lo concerniente a las bases de datos, conexión, desconexion y consulta.

### 3.2.4 Diagrama de Paquetes del Wrapper

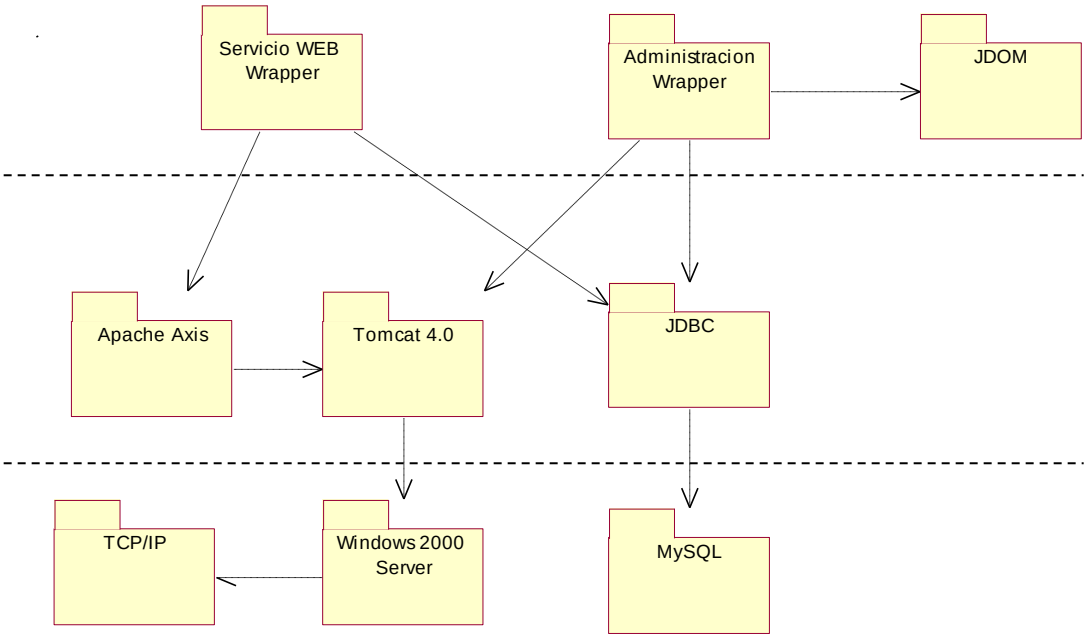



**FIGURA 3.21. DIAGRAMA DE PAQUETES DEL SISTEMA WRAPPER**

El paquete Servicio WEB Wrapper, incluye toda la lógica para el encapsulamiento de las fuentes de datos, atiende las peticiones y las convierte a un formato que la fuente de datos pueda gestionar y devuelve las respuestas tanto de mapas como de datos de las bases de datos.

El paquete de administración Wrapper brinda una interfaz para que el administrador configure los parámetros del Wrapper y pueda publicar las fuentes de datos en el servicio Web, el sistema esta construido utilizando JSP, de allí su relación con el servidor para J2EE Tomcat 4.0. Apache Axis brinda soporte a la publicación de los servicios web, mientras que JDOM es un API en Java para el manejo de documentos XML, tanto lectura como escritura y creación.

### 3.2.5 Diagrama lógico de datos Wrapper

Por el tamaño del diagrama , este se encuentra en el documento diagramas modelado, esto con el fin de no dañar la paginación y formato del documento

**FIGURA 3.22 DIAGRAMA LÓGICO DE DATOS WRAPPER**

**Descripción de las tablas**

1. **basedatos:** Guarda toda la información de las bases de datos a las que accede el sistema.
    a. **URL:** URL de la base de datos (VARCHAR 50)
    b. **Driver:** Driver de la base de datos (VARCHAR 50)
    c. **Login:** Login de la base de datos (VARCHAR 40)
    d. **Password:** Password de la base de datos (VARCHAR 40)
    e. **IDBD :** (Autonumérico)*

2. **Zona: Z**onas del sistema.
    a. **NombreZona:** Nombre de la zona (VARCHAR 50)
    b. **IDZona:** (Autonumérico)*

3. **Tema:** Temas del sistema.
    a. **NombreTema:** Nombre del tema (VARCHAR 50)
    b. **IDZona:** (int)
    c. **DTema:** (Autonumérico)*

4. **Usuario:** Usuarios del sistema.
    a. **LoginUsuario:** Nombre de un Usuario del tema (VARCHAR 40)
    b. **PasswordUsuario:** Password de un Usuario del tema (VARCHAR 40)
    c. **IDUsuario:** (Autonumérico)*

5. **Campo:** Campos del sistema.
    a. **NombreCpo:** Nombre de un Usuario del tema (VARCHAR 40)
    b. **LongCpo:** Longitud máxima del campo (int) **Necesario para crear DBF**
    c. **DecimCpo:** decimales máximos del campo (int) **Necesario para crear DBF**
    d. **IDCpo:** (int)
    e. **IDTabla:** (int)
    f. **IDTipo:** (Autonumérico)*

6. **Usuariotema:** tabla intermedia entre usuario y tema, Un usuario puede tener permiso a uno o mas temas.

   a. **IDUsuario:** (int)
   b. **IDTema:**(int)
   c. **ID** (Autonumérico) *

7. **campotema:** Tabla intermedia; Un campo puede publicarse en diferentes temas y para cada tema puede tener un nickname diferente que lo caracterice dentro del mismo.
   a. **IDCampo:** (int)*
   b. **IDTema:**(int)*
   c. **NickName:**Nombre dado al campo en un tema en particular (VARCHAR 50)

8. **Tipo:** Tipo del campo entero, fecha, cadena. Estos campos son necesarios para determinar el tipo de campo para la respuesta al mediador, y sirve para construir la capa del mapa.
   a. **IDTipo:** (Autonumérico)*
   b. **NombreTipo:**(VARCHAR 30)

9. **Tabla:** Tabla de la base de datos a exportar.
   a. **IDTabla:** (Autonumérico)*
   b. **NombreTabla:**(VARCHAR 30)

10. **entradametadatos:** Representan las entradas de datos del documento Descriptor.
   a. **IDEntrada:** (Autonumérico)*
   b. **IDCpoIn:** (int) Campo de entrada para la búsqueda en la entrada
   c. **IDTema:**(int)
   d. **NombreEntrada:**(VARCHAR 30)

11. **entradacampoout:** Tabla intermedia, Una entrada puede tener varios campos de salida, a la hora de la consulta.
   a. **IDEntrada:** (int) entrada*
   b. **IDCpoOut:** (int) Campo de salida para la búsqueda en la entrada*

12. **campocampo:** representa los campo múltiples. Un campo múltiple puede tener varios campos asociados.
    a. **IDEntrada:** (int) entrada
    b. **IDCpoOut:** (int) Campo de salida para la búsqueda en la entrada
    c. **ID** (Autonumérico)*

13. **mapa:** datos de los mapas del sistema.
    a. **IDMapa:** (autonumérico) *
    b. **IDTema:** (int) tema del mapa
    c. **NombreMapa** (VARCHAR 30)
    d. **URLMapa** (VARCHAR 30)
    e. **NickMapa** (VARCHAR 30)

14. **mapametadatos:** mapas dentro del documento descriptor.
    a. **IDMapa:** (int) *
    b. **IDMapaMetadatos:** (int) *

15. **Usuariomapa:** tabla intermedia entre usuario y mapa, Un usuario puede tener permiso a uno o mas mapas.
    a. **IDUsuario:** (int)*
    b. **IDMapa:**(int)*

16. **Relacion:** Relaciones entre las tablas.
    a. **IDRelacion:** (int)*
    b. **IDTipoRelacion:** (int)
    c. **IDTabla1:**(int)
    d. **IDTabla2:**(int)
    e. **StringRelacion:**(int) Cadena que describe la relación

17. **TipoRelacion:** Relaciones entre las tablas.
    a. **IDTipoRelacion:** (autonumérico)*
    b. **NombreTipoRelacion:**(int)* los tipos pueden ser directa, mucho a muchos y lejana.

18. **RelacionTabla:** Tabla intermedia entre relación y tabla. Las relaciones lejanas y muchos a muchos tienen tablas asociadas entre ellas.
    a. **ID:** (autonumérico)*
    b. **IDRelacion:** (int)
    c. **IDTabla:** (int)

## 3.3 Diagrama de Despliegue del Sistema

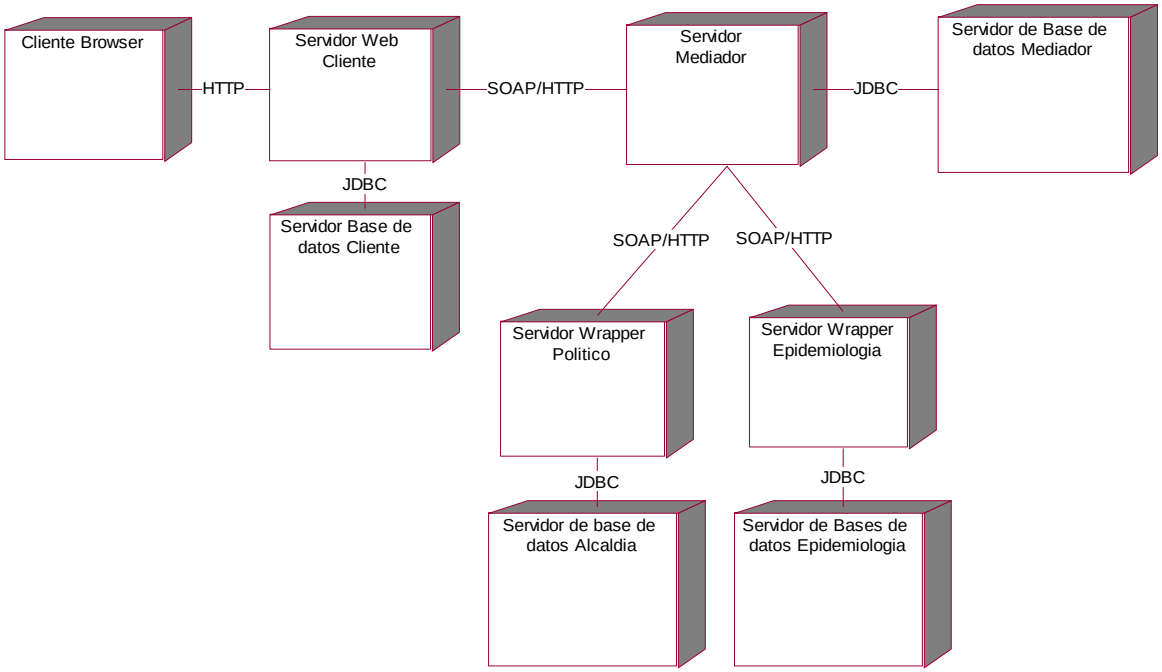



**FIGURA 3.23 DIAGRAMA DE DESPLIEGUE DEL SISTEMA**

**Cliente Browser:** El nodo cliente debe presentar los siguientes requerimientos mínimos:

- Memoria RAM mayor o igual a 128 MB.
- Se recomienda el uso en IE. Por tener un mejor desempeño.
- Navegador IE 5.5 en adelante o Navigator 4.06 en adelante. se requiere instalar el Plug in de Applets java v 1.4 o superior.
- Procesador de 266 MHz o superior.
- Tarjeta de video.

- Tarjeta de red.

**Servidor Cliente web:** Mecanismo principal para ejecutar la lógica residente en el servidor de páginas web que muestra la interfaz al cliente Browser y realiza las peticiones al servicio Web del mediador por medio de una interfaz SOAP. El servidor de Cliente Web es responsable de ejecutar el código de las páginas de servidor y Beans de Java. El servidor es Apache Tomcat 4.1.

Software de soporte :

- Especificaciones Java Servlet 2.2 y JavaServer Pages 1.1.
- Windows 2000 Server con Service Pack 6.0

El ambiente que se tiene configurado es una alternativa, pero es posible correr el servicio en cualquier servidor de aplicaciones que cumpla con el estándar J2EE.

**Servidor de Base de datos Cliente:** Contiene la información persistente del sistema, esta incluye, información sobre usuarios y datos de acceso al Servicio Web del mediador. La base de Datos escogida para el despliegue del servicio es Mysql 3.23.22. Soporta mecanismo de comunicación JDBC, a través del driver de sun para mysql.

Software de Soporte :

- Windows 2000 Server Service Pack 6, J2SE.

**Servidor Mediador:** Contiene la lógica que presta el Servicio Web de Mediación, además de la aplicación web de gestión de la misma. El servidor utilizado es Apache Tomcat 4.1.

Software de soporte :

- Apache Axis.
- Especificaciones Java Servlet 2.2 y JavaServer Pages 1.1.
- Windows 2000 Server Service Pack 6.0

Nuevamente, el ambiente que se tiene configurado es una alternativa, pero es posible correr el servicio en cualquier servidor de aplicaciones que cumpla con el estandar J2EE.

**Servidor de Base de datos Mediador:** Contiene la información persistente del sistema, esta incluye, información sobre usuarios, acceso a las fuentes de datos y potencialidades de las mismas. La base de Datos escogida para el despliegue del servicio es Mysql 3.23.22. Soporta mecanismo de comunicación JDBC, a través del driver de Sun para Mysql.

Software de Soporte :

- Windows 2000 Server Service Pack 6, J2SE.

**Servidor de aplicaciones Wrapper:** Contiene la lógica que presta el servicio Web Wrapper, además de la aplicación web de gestión del mismo. El servidor utilizado es Apache Tomcat 4.1.

Software de soporte :

- Apache Axis.
- Especificaciones Java Servlet 2.2 y JavaServer Pages 1.1.
- Windows 2000 Server Service Pack 6.0.

**Servidor de Base de datos Wrapper:** Contiene la información persistente del sistema wrapper, esta incluye, información sobre usuarios, acceso a las bases de datos publicadas, opciones de acceso y potencialidades de las mismas, información de acceso a los mapas de las fuentes de datos, y los archivos de metadatos para la publicación de sus potencialidades. La Base de Datos escogida para el despliegue del servicio es Mysql 3.23.22. Soporta mecanismo de comunicación JDBC, a través del driver de Sun para Mysql.

Software de Soporte :

- Windows 2000 Server Service Pack 6, J2SE.

# CAPITULO 5. CONCLUSIONES

- En una era como la actual donde el poder de gestión de las grandes empresas está representado por la información , encontramos que no sólo es necesario contar con los datos , sino que es indispensable contar con las herramientas adecuadas para su procesamiento con el fin de obtener resultados óptimos.

- El sistema mediador basado en Servicios Web desarrollado en el presente trabajo de grado, para la validación de la arquitectura, permite multiplicar la potencialidad de los sistemas de información geográficos al facilitar la obtención de información procedente de diferentes fuentes, con diferentes formatos y sobre diferentes plataformas, con el fin de realizar análisis más especializados.

- La tecnología de los servicios web revoluciona de manera avasallante la computación distribuida por sus características de transparencia, reusabilidad y modularidad, lo que la hace propicia para su uso en los sistemas que involucran comunicación de procesos vía web, tales como la interconexión de sistemas de información remotos.

  Sin embargo es importante aclarar que no consideramos que los servicios web vengan a reemplazar las tecnologías de computación distribuida tradicionales como CORBA o COM/DCOM, sino que viene como un complemento para el trabajo sobre Internet , pues como bien se sabe dichas tecnologías tienen un excelente desempeño en ambientes corporativos y dominios específicos como la gestión de redes en donde encontramos implementaciones bastante estables y robustas.

- La arquitectura de sistema mediador, permite que las fuentes de datos trabajen de forma federada, es decir tengan su sistema de gestión propio, y les provee además control total sobre la forma y la cantidad de datos que van a compartir en el sistema mediador por medio de la publicación de las fuentes.

- Los servicios Web son una plataforma propicia para el desarrollo de aplicaciones con arquitectura de sistema mediador dada la similitud de los procesos de publicación y acceso de las mismas.

- En una arquitectura como la del mediador, las peticiones de los clientes al mediador deben ser convertidas al formato de cada una de las fuentes de datos y nuevamente formateadas para devolverlas a los clientes, siendo evidente que el procesamiento del sistema es bastante complejo y por lo tanto causa retardos que para el desarrollo de aplicaciones empresariales deben ser tenidos en cuenta como desventaja de un sistema de este tipo.

- La arquitectura planteada en el presente trabajo de grado, fue validada en el dominio de los sistemas de información geográficos, para tal efecto se desarrolló la plataforma como tal y varios clientes que validaran su funcionalidad. La concepción, diseño e implementación de la plataforma consumió el 95% del tiempo del proyecto y los demás elementos, que sin embargo, son los mas visibles para el cliente, se realizaron en el tiempo restante.

**REFERENCIAS**

[ASD02] Apache SOAP documentation
http://ws.apache.org/soap/docs/index.html

[DEP99] Department of Environmental and Heritage. *ERIN- Environmental Resources Information Network*.1999
http://www.erin.gov.au/erin.html

[DTO96] DCOM Technical Overview
HTTP://www.microsoft.com/ntserver/docs/DCOMTEC.doc

[EDO02] Exportación de datos usando OpenGis, García Ontiveros. 2000 Universidad de las Ameritas-Puebla

[EDW01] A. Cuenca; M. Hurtado, J. Samos. "Extracción de información de fuentes de datos heterogéneas e incorporación al data warehouse", Universidad de Granada, 2001.

[ESO00] Especificaciones SOAP. 2000.
HTTP://www.w3.org/TR/SOAP

[ESR03] Formato de shapefiles en la Pagina principal de ESRI
http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[EUD01] Especificaciones UDDI. 2001.
HTTP://www.uddi.org/specification.html

[FGDC99] *Federal Geographic Data Committee* (FGDC). 1999
http://www.fgdc.gov

[FWI03] J. C. Corrales, "Framework de interoperabilidad". Informe técnico, Universidad del Cauca, Colombia. 2003.

[GID03] Tutorial de SIG: GIS development.
www.GISdevelopment.net

[GIS02] Articulo: "Los sistemas de información Geográfica" por Agrim. María Cristina Luque Master en Planificación Territorial Medioambiental y Urbana.
http://www.cpa.org.ar/InfoTec/Gis.htm

[GMO00] GIS and Markets Opportunities Report . Daratech Inc. 2000

[IGI00] A. Gupta, R. Marciano, I. Zaslavsky, C. Baru, "Integrating GIS and Imagery through XML-based Information Mediation", San Diego Supercomputer Center.2000

[IWS03] IBM WebSphere SDK for Web Services (WSDK)
http://www-106.ibm.com/developerworks/webservices/wsdk/

[JAX03] JSR-000101 Java™ APIs for XML based RPC
http://www.jcp.org/aboutJava/communityprocess/first/jsr101/

[JBW02] JBuilder and web Services
http://info.borland.com/techpubs/jbuilder/jbuilder8/websvcs/webservices_overview.htm

[MME01] M. de C. Moura, H. de Mello P. ,A. Kiyoshi Tanaka, "A Metadata Model for Supporting Data Extraction from Environmental Information Systems". Instituto Militar de Engenharia y Universidade do Rio de Janeiro, 2001

[MMI96] Mena, E., Kashyab, V., Sheth, A., Illarramendi, A. "Managing Multiple Information Sources through Ontologies: Relationship between Vocabulary Heterogeneity and Loss of Information", in Proceedings of the 3rd Workshop Knowledge Representation Meets Databases (KRDB '96), F. Baader, et al., Editors. 1996.

[MST00] S. Ram, J. Park, "Modeling Spatial and Temporal Semantics in a Large Heterogeneous GIS Database Environment". Department of Management Information Systems College of Business and Public Administration. 2000.

[NATB99] *National Aeronautics & Space Administration*. Earth Pages.1999 http://starsky.hitc.com/earth/earth.html

[NOAA99] U.S. Department of Commerce – *National Oceanic and Atmospheri Administration*- NOAA. 1999 http://www.esdim.noaa.gov/NOAA – Catalog.

[OGC03] Pagina principal de Open Gis consortioun OGC www.opengis.org

[RRS97] Goh, C.H.: Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources, in Sloan School of Management. 1997, MIT: Boston. p. 112

[SOA00] SOAP (Simple Object Acces Protocol). 2000. HTTP://www.techmetrix.com/trendmarkers/publi.php?P=12003

[SWE01] Servicios Web. 2001 HTTP://www.fisica.uson.mx/carlos/WebServices/WSRevolution.htm

[TuG03] Tutorial SIG. Página Ciudad de Fort Collins Colorado. http://www.ci.fort-collins.co.us/

[TWP01] UDDI Technical White Paper. 2001 HTTP://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[UDD01] Understanding UDDI. 2001. HTTP://www-106.ibm.com/developerworks/webservices/library/ws-featuddi/?dwzone=webservices

[UDK99] UmwelDatemKatalog. 1999
http://www.mu.niedersachsen.de/udk

[UMA00] M.N. Terrasse, M. Savonnet, G. Becker, "An UML-metamodeling Architecture for Interoperability of Information Systems". Laboratoire LE2I, Université de Bourgogne, B.P. 47870, 21078 Dijon Cedex, France. 2000.

[WIS00] What is a SIG? U.S. Geological Survey (USGS) .
http://webgis.wr.usgs.gov/globalgis/

[WSA03] Web Services Architecture. 2003.
HTTP://www.w3c.org/TR/ws-arch

[WSC01] Web Services Conceptual Architecture. 2001
HTTP://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

[WSC03] Web Services Choreography. 2003
HTTP://www.w3.org/TR/2003/WD-ws-chor-reqs-20030812/

[WSD01] Especificaciones WSDL. 2001.
HTTP://www.w3.org/TR/wsdl

[WSJ02] Web Services for J2EE, Version 1.0
http://www-106.ibm.com/developerworks/webservices/library/ws- jsr109-proposed/

[WSO01] Web Services – Technical overview. 2001
HTTP://dcb.sun.com/practices/webservices/overviews/overview_wsdl.jsp

**INDICE ANALÍTICO**