Capítulo VI

Modelo, Implementación y Resultados

En este capítulo se describe el modelo, la implementación y los resultados de nuestro exportador de datos. Tomando OpenGIS como estándar, ArcView como fuente de información y UDLASIG como visualizador de base.

6. Modelo

Después del análisis de los modelos, formatos y aplicaciones que se presentan en los capítulos anteriores se concluyó que el exportador de datos junto con el visualizador debían tener las siguientes características: debe estar implementado en Java tanto la aplicación del exportador como el applet, el procesamiento de las operaciones básicas de un SIG (acercamientos, alejamientos y paneos) deben realizarse en el cliente y las consultas a la base de datos deben realizarse solamente al buscar y guardar información en la base de datos. La arquitectura que proponemos se describe en la figura 6.1

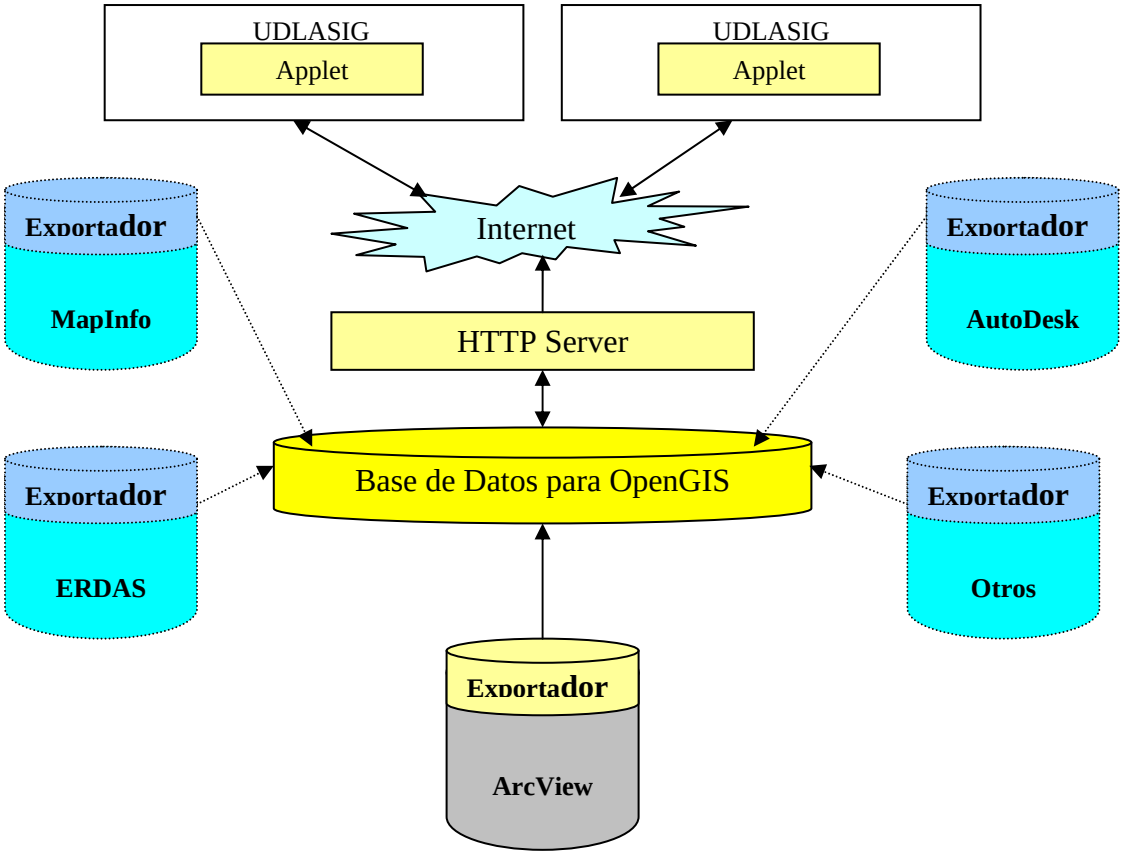

Figura 6.1 Arquitectura propuesta

La arquitectura que se propone como se muestra en la figura 6.1 se compone de las siguientes partes, un exportador junto con una aplicación comercial de un SIG, que en este caso se empleo ArcView y un exportador que lee los “ShapeFiles” y los almacena en la base de datos modelada para el empleo de OpenGIS. Un esquema más general de esta parte lo podemos apreciar en la figura 6.2.

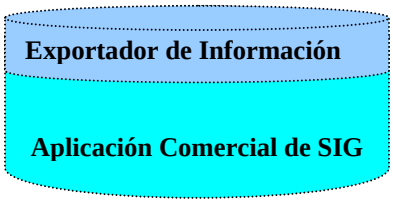

Figura 6.2 Esquema general del Exportador y la Aplicación comercial de un SIG

Esta idea se puede extender para el empleo de otras aplicaciones comerciales como lo son: MapInfo, AutoDesk, Geomedia y otros. La Base de datos es de tipo objeto-relacional y se encuentra modelada para el almacenamiento de datos tipo OpenGIS y se representa con la figura 6.3

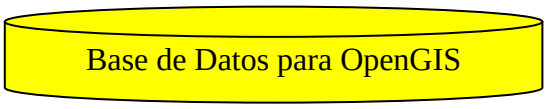

Figura 6.3 Base de datos

Por último el applet de UDLASIG se emplea como visualizador de la información que se almacena en la base de datos y se representa con la figura 6.4

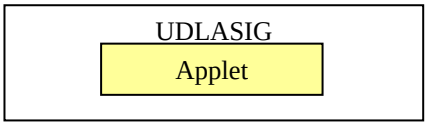

Figura 6.4 Applet del UDLASIG

Para su mejor entendimiento y desarrollo dividimos en tres partes todo el proyecto:

- Exportador
- Base de Datos
- Visualizador

El flujo de datos entre las partes se da, de la siguiente forma:

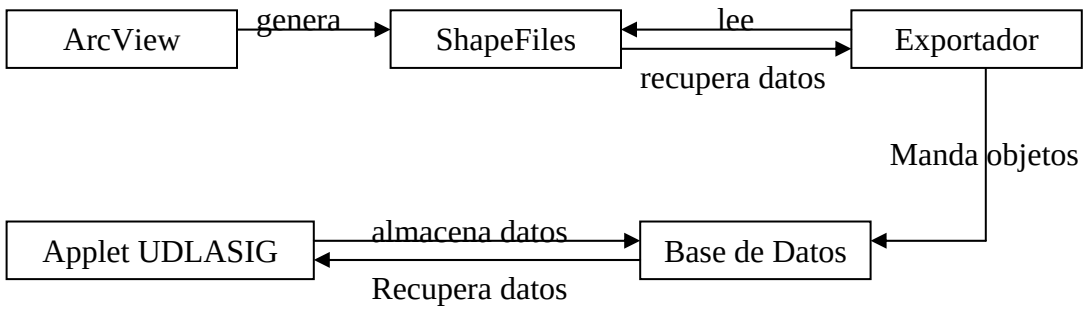

Figura 6.5 Flujo de datos

6.1 Exportador

El exportador es una aplicación de Java la cual se encarga de clasificar, leer los "shapefiles" y extraer la información para mandarla a la base de datos que se encuentra modelada bajo las especificaciones de OpenGIS. El exportador presenta una interfaz sencilla, la cual cuenta con dos ventanas, la primera muestra una ventana de clasificación, la cual se muestra en la figura 6.6, ahí se muestran las opciones que se tienen para clasificar los shapefiles en las tablas propuestas por OpenGis

También se cuenta con una ventana de selección, la cual se muestra en la figura 6.7, en esta ventana se selecciona el archivo con extensión ".shp" para exportar los datos.

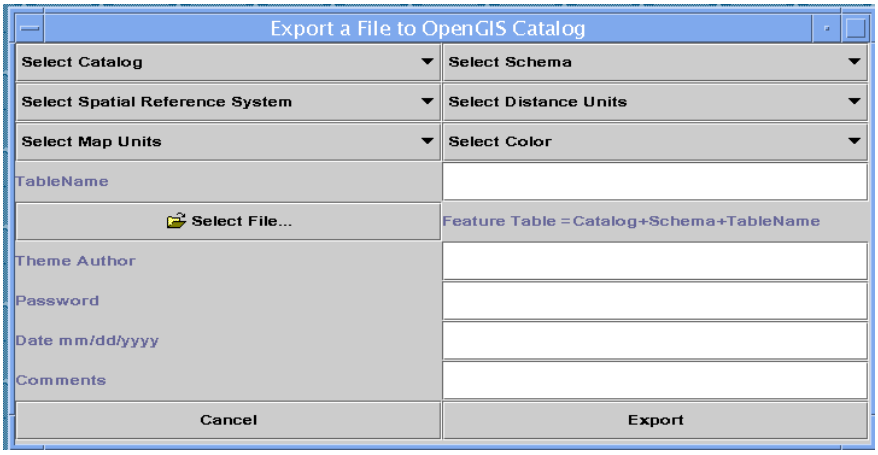

Figura 6.6 Clasificación del archivo a exportar

En la figura 6.6 se muestra la interfaz que nos ayuda a clasificar nuestra información en los catálogos propuestos por OpenGis.

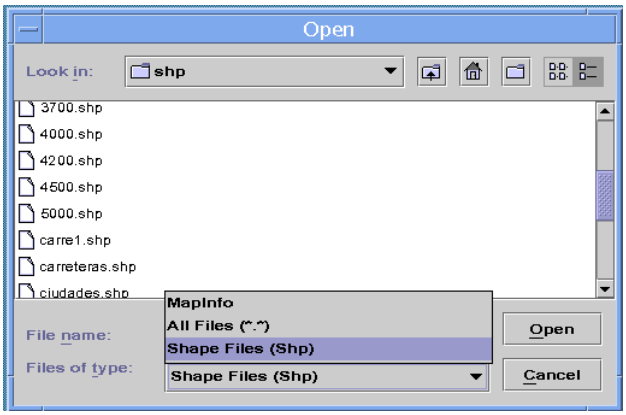

Figura 6.7 Ventana de selección de archivos shp

Cabe resaltar que el exportador de datos solamente emplea los archivos con extensión “shx” y “shp”, que se deben de encontrar en el mismo directorio, de faltar alguno de ellos, la aplicación se cerrará de forma automática.

Debido a que los “ShapeFiles” manejan los dos tipos de orden del byte más significativo (BigIndian y LittleIndian) fue necesario implementar algoritmos para el manejo de bytes, ya que trabajamos con Java y este lee los bytes en orden BigIndian. Los pasos que se siguieron para la recuperación de los datos se basaron en las estructuras de las tablas del capítulo 4 y son los siguientes:

- Leer los datos de una archivo binario (“shp” / ¨shx”) y almacenarlos en una estructura dinámica como un vector
- Crear un método que nos permita leer bytes en orden LittleIndian y pasarlos a BigIndian y después regrese el tipo de número según la cantidad de bytes que reciba (4 bytes es entero, 8 bytes es doble)
- Leer los primeros 100 bytes de los encabezados y obtener el tipo de “shape” y el número de registros (capítulo 4, tabla 4.2)
- Leer el archivo “shp” a partir del byte número 100 y recuperar los datos dependiendo de l tipo de “shape” (ver tablas del capítulo 4)

Los principales algoritmos que se emplearon para el manejo de bytes y un ejemplo de recuperación de datos de un archivo “shp” de tipo polígono se encuentran en el apéndice B. En el apéndice C se muestra una corrida en pantalla de la lectura de un archivo “shape” de tipo punto.

6.2 Base de Datos

El modelado de la base de datos se tomó directamente de la especificación de OpenGIS [OpenGIS, 99], y se describe en el capítulo 5. El tipo de implementación que se escogió de

los existentes fue la implementación de SQL92 con tipos geométricos de la tabla de características (SQL92 Geometry Types Implementation of Feature Tables), la cual consiste de tablas o vistas con tipos de SQL y funciones de SQL. La base de datos objeto-relacional empleada es IUS (Informix Universal Server), tomando como base para su implementación el modelo de Posada [Posada, 99].

A continuación se presenta el esquema de las tablas de características de OpenGIS.

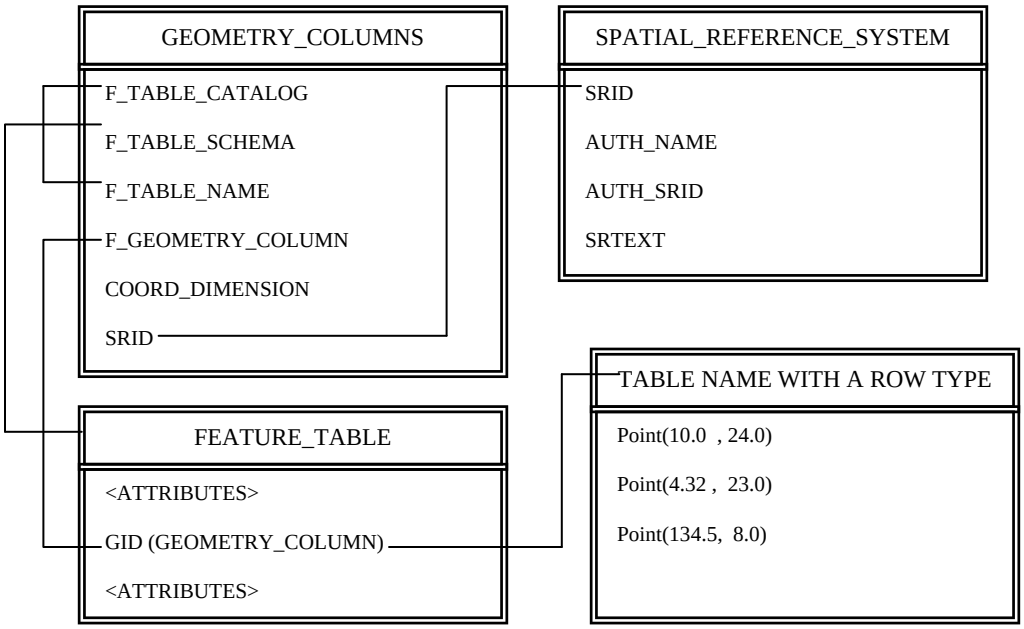

Figura 6.8 Esquema de las tablas bajo la especificación de OpenGIS con tipos geométricos

En la figura 6.8 se puede apreciar que existen tablas son tipos de datos y tablas sin tipos.

Las tablas sin tipos de datos nos ayudan a navegar entre tablas y a organizar la base de datos.

La construcción de las tablas o vistas se crearon bajo las siguientes declaraciones y estructuras:

```
CREATE TABLE SPATIAL_REF_SYS
```

```
{
        SRID            INTEGER NOT NULL PRIMARY KEY,
        AUTH_NAME       VARCHAR(256),
        AUTH_SRID               INTEGER,
        SRTEXT                  VARCHAR (2048)
}
```

La explicación y el significado de la tabla se encuentra en la sección 5.3 del capítulo. Hay que resaltar que el campo SRID es el que nos sirve de llave primaria para hacer referencia a la tabla GEOMETRY_COLUMNS.

La tabla GEOMETRY_COLUMNS maneja una estructura adecuada para que nos permite clasificar la información que tenemos sobre proyectos y temas específicos

```
CREATE TABLE GEOMETRY_COLUMNS {
        F_TABLE_CATALOG    VARCHAR(256) NOT NULL,
        F_TABLE_SCHEMA     VARCHAR(256) NOT NULL,
        F_TABLE_NAME               VARCHAR(256) NOT NULL,
        F_GEOMETRY_COLUMN          VARCHAR(256) NOT NULL,
        COORD_DIMENSION    INTEGER,
        SRID                       INTEGER REFERENCES SPATIAL_REF_SYS,
    CONSTRAINT GC_PK PRIMARY KEY
    (F_TABLE_CATALOG,          F_TABLE_SCHEMA,          F_TABLE_NAME,
    F_GEOMETRY_COLUMN)
}
```

La descripción de cada uno de los elementos de la tabla y la tabla misma se encuentra en la sección 5.4.2 del capítulo 5, aquí vale la pena resaltar que los campos de la tabla de GEOMETRY_COLUMNS son los que nos ayudan a clasificar a la tabla FEATURE TABLE/VIEW.

La tabla Feature Table/View (Tabla de características o vistas) almacena las características de un proyecto determinado y se crea una tabla de estas por cada nuevo proyecto.

Nota: la creación de las tablas se llevó acabo mediante el uso de la interfaz de IUS, con las características mencionadas en cada declaración con las siguientes modificaciones:

Todos los VARCHAR(256) de la tabla GEOMETRY_COLUMNS se declararon con un tamaño de VARCHAR(200) debido a que la CONSTRAINT GC_PK PRIMARY KEY, solamente puede tener sumando los VARCHAR de sus elementos un tamaño de 400 por ser llave primaria.

El campo SRTEXT de la tabla SPATIAL_REF_SYS fue declarado como TEXT debido a que el tamaño máximo de un VARCHAR es de 255 (tomando en cuenta que comienza en cero tenemos un total de 256 caracteres) y por consiguiente el campo AUTH_NAME tiene un tamaño de 255.

Para la creación de los tipos geométricos se buscó que estos cumplieran con la siguiente estructura:

Punto:= (X,Y)

Donde X, Y = números de punto flotante de doble presición

LíneaString> := Punto {,Punto}*

Polígono := LíneaString {,LíneaString}*

MultiPunto := Punto {,Punto}*

MultiLíneaString := LíneaString {, LíneaString }*

MultiPolígono := Polígono {,LíneaString}*

La notación {}* denota 0 o más repeticiones de los elementos que se encuentran entre los paréntesis.

Estos tipos de datos fueron creados en IUS (apéndice B sección 9), pero esta herremienta no acepta directamente la recuperación de datos de una lista como esta:

LíneaString(10 10, 10 20, 20 20, 30 40)

No se puede hacer directamente desde IUS, por ese motivo se almacena el objeto geométrico de la siguiente forma:

LíneaString(1,10 10)

LíneaString(1,10 20)

LíneaString(1,20 30)

LíneaString(1,30 40)

donde LíneaString(Indice,X Y)

todos los elementos del tipo LíneaString con el mismo índice, forman un mismo elemento geográfico.

En la base de datos se crearon los nuevos tipos de datos que manejan coordenadas de tipo número de doble precisión, hay que resaltar que IUS le da el mismo tratamiento a números del tipo punto flotante y punto flotante de doble precisión.

Toda la información que guardan los “shapefiles” es recuperada, pero no toda es de utilidad para el uso del estándar de OpenGIS. El mapeo de datos que se realiza es el de las tablas 3.2 y 3.3 del capítulo 5 de OpenGIS. Todos los datos son almacenados en la base de datos según la especificación de OpenGIS [OpenGIS,99] explicada en el capítulo 5 .

6.3 Visualizador

El visualizador para el Web, es trabajo realizado por Briones [Briones, 98] y Posada [Posada,99]. Se le hicieron algunas adaptaciones como los son pasar todas las coordenadas de los objetos a números de punto flotante, con sus respectivas funciones. Las opciones de análisis como lo es la intersección de líneas y/o capas quedarán por el momento deshabilitadas.

El visualizador tiene las siguientes características:

- Applet de java que emplea Java 2 con sus respectivas clases de Java Swing y Java2D
- Operaciones de Visualización como acercamientos, alejamientos y paneos.
- Despliegue de coordenadas y elementos geográficos en 2D, esto es debido a que la especificación de OpenGIS [OpenGIS, 99] así lo estipula.
- Procesamiento de las operaciones de visualización en el cliente.

El visualizador no contaba con la posibilidad de manejar coordenadas de números de punto flotante de doble precisión. De hecho para el manejo de los objetos básicos de un SIG, punto, línea, polilínea y polígono, se heredaban las características de la clase Polygon de

java.awt. incluyendo sus métodos de pintado, es decir que en la interfaz solamente se dibujaban polígonos de tipo entero. Dentro de la clase java.awt.Graphics2, la cual es la extensión del java.awt.Graphics, no existe el tipo “Polygon”, de tal manera que hubo que reestructurar todos los objetos básicos, así como sus métodos de pintado. Para ello se emplearon las clases del java2D que cuentan con objetos tipo punto, línea que permiten el manejo de coordenadas de tipo número con punto flotante de doble precisión y poseen sus propios métodos de dibujo que realizan las conversiones necesarias entre coordenadas enteras y coordenadas de números flotantes de doble precisión a la hora de mostrar los objetos geográficos en un contenedor, el cual en este caso es un panel donde se muestran las vistas. Esto quiere decir que todas las conversiones de coordenadas de número flotante con respecto al dispositivo donde se muestran los datos geográficos, son realizados por los métodos de las clases de java2D.

Sobre el visualizador podemos comentar que actualmente no todos los navegadores están habilitados para ejecutar los applets, que emplean Java Swing y Java2D, pero ya se encuentra disponible el plug-in 1.2 de java para estaciones de trabajo y pc´s, los cuales fueron instalados en nuestras máquinas de trabajo. En un corto plazo se dispondrá de un plug-in para Macintosh, lo cual nos permitirá usar nuestro applet sin ningún problema en cualquier plataforma.. Lo anterior se puede considerar como una limitante temporal. Otro factor que se considera en el empleo de JavaSwing y Java2D es el tener dos tipos de archivo html para invocar un applet. El primero se emplea para visualizar cualquier applet que no utilice las Java Foundation Classes (JFC), apéndice B sección 7 y el segundo que emplea tags especiales para usarse con el plugin java1.2 (apéndice B sección 7).

Finalmente mostramos el resultado del presente trabajo, primeramente mostramos tres archivos “shape” con los tipos geométricos básicos (línea, punto y polígono) en la figura 6.9. Estos shapes fueron generados a partir de planos generados en AutoCad.

Figura 6.9 Interfaz de ArcView con los tipos básicos de shape.

Y en la figura 6.10 mostramos el resultado de la combinación del exportador y el UDLASIG con nuevas características.

Como se puede apreciar en la figura 6.10 la aplicación ya se puede utilizar a través del Web, despliega la información de los shapefiles con coordenadas de números de punto flotante de doble precisión y emplea una base de datos bajo el modelo de OpenGIS.

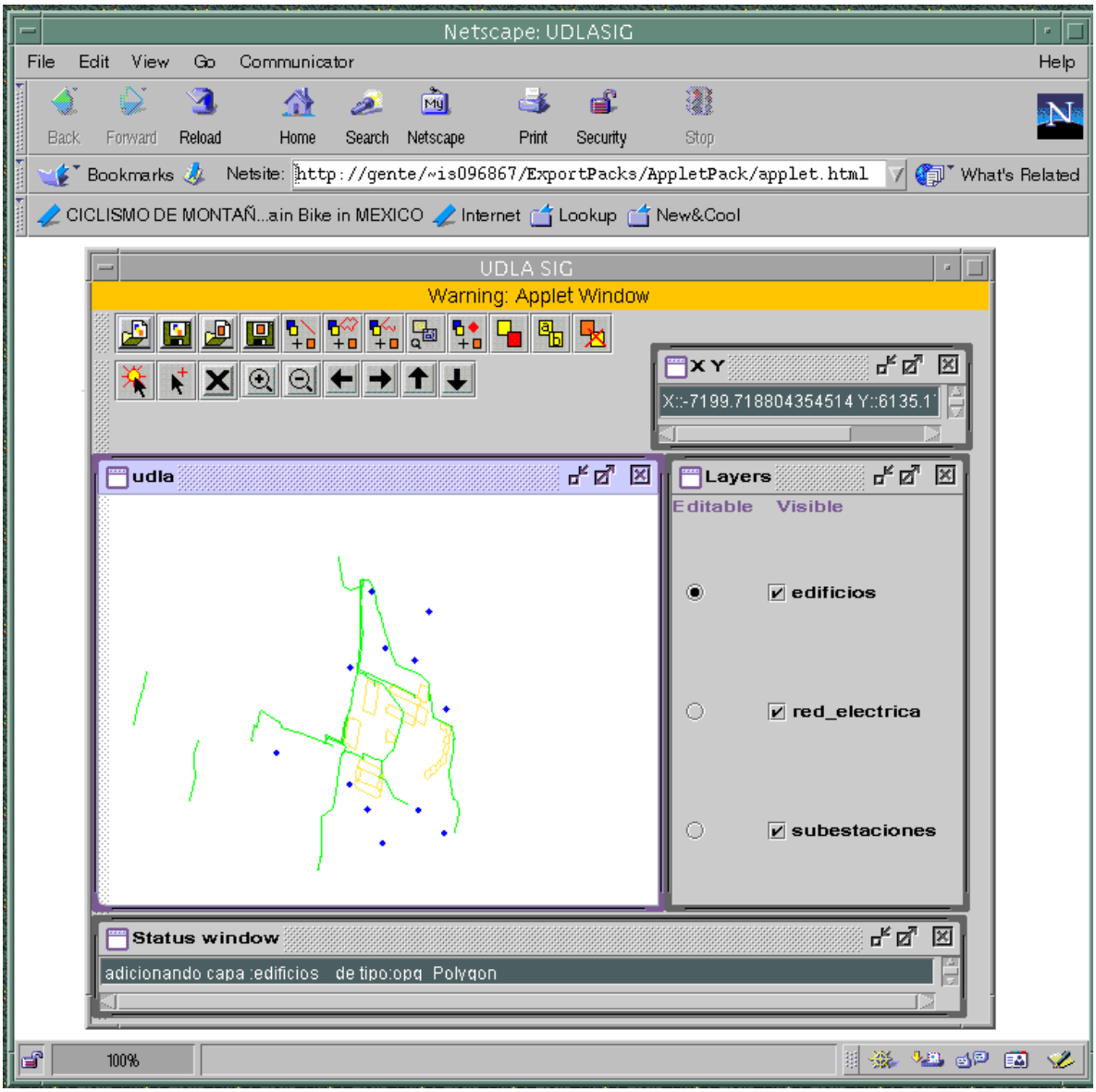

Figura 6.10 Interfaz del UDLASIG

6.4 Conclusiones

La interpretación del modelo de datos de OpenGIS fue algo complejo ya que la especificación no es muy clara, todo lo contrario a la especificación de los archivos “shp” la

cual es muy sencilla de entender. Los resultados que se obtuvieron son aceptables y se logró el cometido, desarrollar una interfaz que visualice la información de los archivos “shp” y que almacene los datos bajo la especificación de OpenGIS. El exportador puede extenderse más mediante la incorporación de otros formatos propietarios y de la definición de un catálogo, sin embargo no toda la información la proveen los archivos “shp”, lo cual es una deficiencia por cubrir.