3. UMN MapServer

http://mapserver.gis.umn.edu

"Bienvenidos página de MapServer. MapServer es un entorno de desarrollo Open Source para construir aplicaciones de Internet con capacidades espaciales. MapServer no es un sistema GIS completo, tampoco aspira a serlo. En cambio, MapServer sobresale en la interpretación de datos espaciales (mapas, imágenes, y datos vectoriales) para el web".

3.1. Introducción

Los inicios de MapServer provienen de un proyecto de doctorado sobre la construcción de un sistema para planear rutas para el "Boundary Waters Canoe Area Wilderness" (BWCAW) en Minnesota del norte, una especie de parque natural. Se enlazó Arc/Info al web generando AMLs en tiempo real. Entonces en 1995 la NASA subvencionó un proyecto para proporcionar información forestal al personal encargado de gestionar el parque, como imágenes de satélite y datos GIS. Este proyecto se conoce como ForNet (http: //fornet.gis.umn.edu/). Las licencias de Arc/Info resultaban algo caras, lo que obligaron a comenzar a mirar algo nuevo. Se consideró la solución ArcView 1 de ESRI, pero no produjo un resultado satisfactorio. Luego vinieron las librerías shp (shapelib) ideadas por Frank Warmerdam que trabajaban con la librería gráfica de creación de imágenes GD. En aquel tiempo también se había creado a un servidor para imágenes ERDAS basado en plantillas y archivos config. Ambos sistemas (las librerías shp y el servidor de imágenes) derivaron en un nuevo software específico al que se denominó MapServer.
En el siguiente gráfico se puede ver gráficamente esta progresión y las sucesivas versiones del producto final hasta nuestros días.

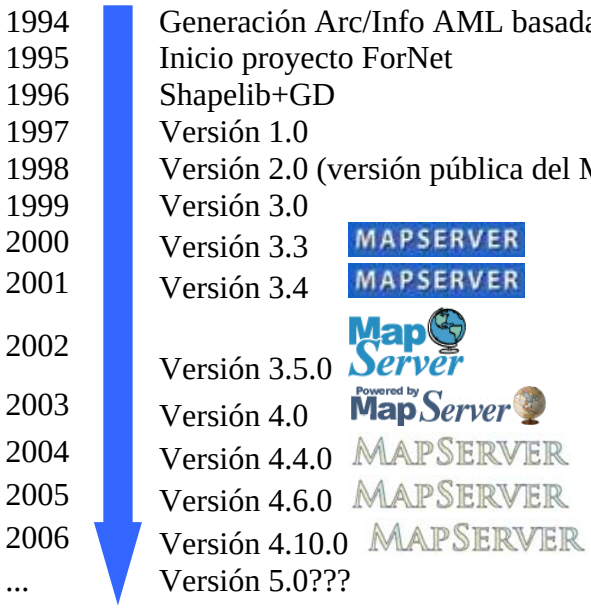


Figura 3.1 Evolución del UMN Mapserver.

3.2. Instalación

Dado el continuo crecimiento de usuarios de MapServer, se han ido creando manuales y ayudas para hacer más rápida y fácil la adaptación de nuevos usuarios al sistema. La más importante y reciente es el paquete MS4W (http://maptools.org/ms4w/) de MapTools.org, un recurso para usuarios y desarrolladores de servidores de mapas Open Source. Este paquete es un instalador de MapServer para plataformas Windows y nos permitirá instalar un entorno de trabajo para MapServer.
El MS4W básico instalará un servidor web preconfigurado que incluirá los siguientes componentes:

- Apache HTTP Server version 2.0.58
- PHP version 5.1.4 or 4.4.3-dev
- MapServer CGI 4.8.4
- MapScript 4.8.4 (CSharp, Java, PHP, Python)
- Includes support for Oracle 10g, and SDE 9.1 data (if you have associated client/dlls)
- MrSID support built-in
- GDAL/OGR Utilities
- MapServer Utilities
- PROJ Utilities
- Shapelib Utilities
- Shp2tile Utility
- OGR/PHP Extension 1.0.0
- OWTChart 1.2.0

La última versión de MS4W es la 2.1 (julio de 2006).

Aunque no se vaya a utilizar alguno de estos componentes es muchísimo más sencillo instalar el paquete completo que hacerlo por componentes separados. Además, se crea una estructura de directorios que utilizan muchos usuarios, lo que permite intercambiar aplicaciones sin modificar casi nada, y las nuevas versiones que salgan de este paquete se pueden actualizar en un instante.
Vamos pues a seguir el sencillo procedimiento de instalación. Debemos empezar por descargarnos el paquete. Para ello vamos a la página de descargas de maptools.org, que es: http://maptools.org/ms4w/index.phtml?page=downloads.html. Aquí podemos ver las diferentes versiones disponibles así como paquetes adicionales. De momento "cogeremos" la última versión que haya del MS4W. Son aproximadamente 30MB, así que dependiendo de la conexión que se disponga tardará más o menos.

Figura 3.2 Descarga del paquete MS4W.

Elegiremos la opción 'Guardar' y lo dejaremos en la raíz de una unidad de disco. En el caso que nos ocupa, será la C:\.
Una vez acabada la descarga descomprimiremos el contenido en la misma raíz. Con esto se nos creará el siguiente sistema de directorios:

C:\ms4w	Directorio principal
C:\ms4w\Apache	Instalación del Apache
C:\ms4w\apps	Directorio de las aplicaciones
C:\ms4w\gdaldata	Ficheros de soporte para formatos GDAL
C:\ms4w\gdalplugins	Ficheros dll requeridos por plugins GDAL
C:\ms4w\httpd.d	Ficheros de configuración de las aplicaciones instaladas en C:\ms4w\apps
C:\ms4w\proj	Ficheros de definición de proyecciones cartográficas
C:\ms4w\tmp	Ficheros temporales
C:\ms4w\tmp\ms_tmp	Ficheros temporales que necesitan ser accesibles vía web como las imágenes creadas por MapServer

Si es la primera vez que instalamos el paquete, bastará con hacer doble clic en el archivo 'c:\ms4w\apache-install.bat' para activar el servicio Apache Web Server. El servidor web Apache queda instalado en el sistema como un servicio de Windows con lo que al reiniciar el ordenador se volverá a activar automáticamente.
Si algún día queremos cambiar la versión del paquete haremos doble clic en el archivo 'c:\ms4w\apache-uninstall.bat', con lo que desinstalaremos el servicio. Ahora cambiamos el nombre a toda la carpeta 'ms4w' (también se puede borrar, pero es mejor hacer este paso intermedio para poder volver atrás en caso de error) a por ejemplo 'ms4w-old'. Ahora volvemos a repetir el método que hemos utilizado para hacer una instalación nueva.
En ambos casos, si todo ha ido bien, abriendo un navegador y entrando una de las siguientes URLs:

http://localhost/ o http://127.0.0.1/

También funcionaría utilizando el nombre del servidor en la barra de navegación. En el ejemplo que seguiremos en este trabajo el ordenador que utilizamos como servidor se llama 'GIS002' por lo que utilizando la URL http://gis002/ nos direcciona a la misma página principal.

Deberíamos ver la página principal de MS4W en el navegador:

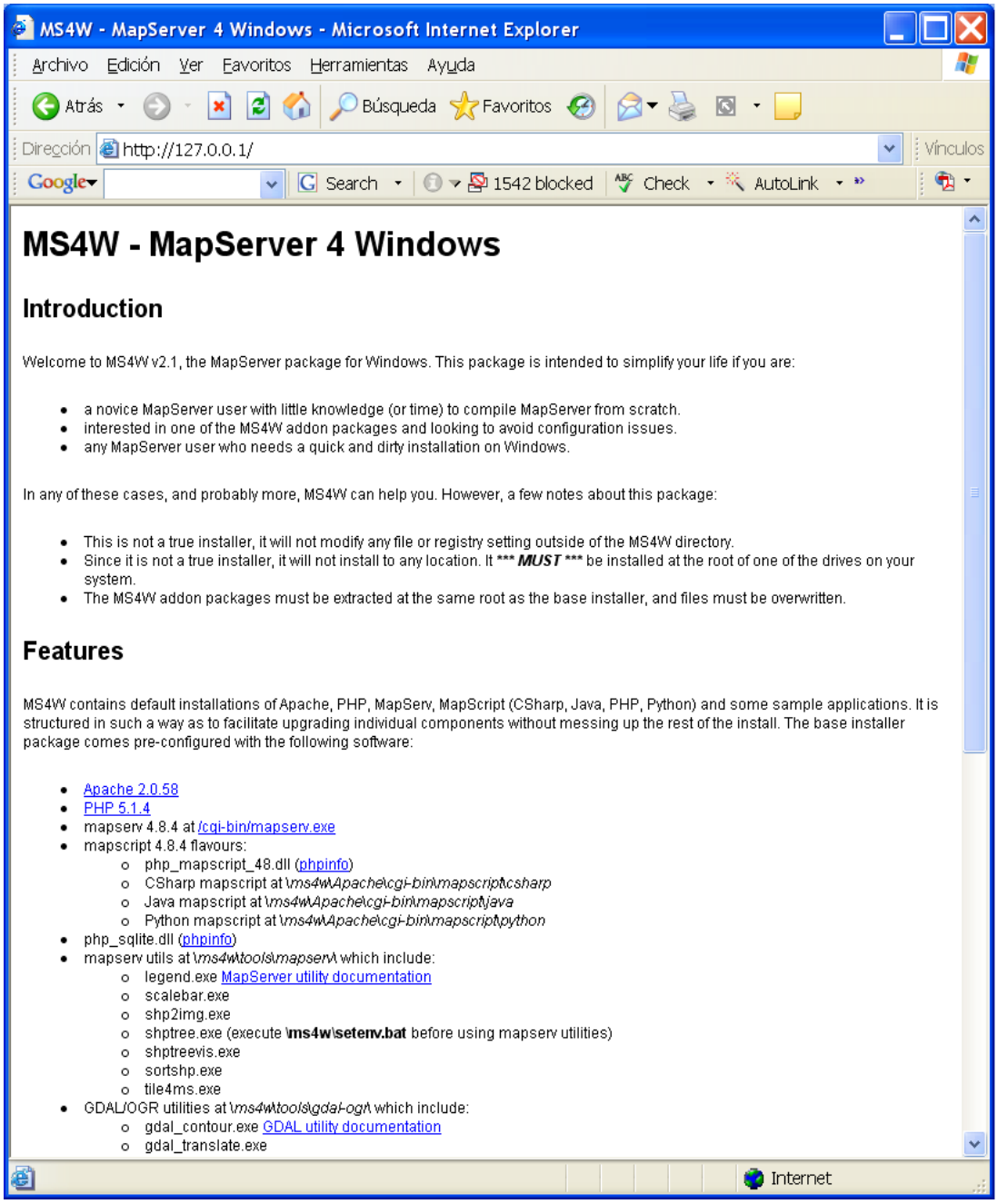


Figura 3.3 Página de bienvenida del paquete MS4W.

Técnicamente en este punto, tendremos instalado el paquete con las características que nos indica la página.
Es posible instalar alguna aplicación que nos ayude a la hora de empezar con MarpServer como el MapLab, que es un editor de ficheros de mapa (.map) que veremos más adelante.

3.3. Componentes

MapServer produce mapas en un entorno CGI en el cual un usuario accede al servidor Apache desde un navegador. Common Gateway Interface (en castellano «Interfaz Común de Pasarela», abreviado CGI) es una importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa.
Las aplicaciones CGI fueron una de las primeras maneras prácticas de crear contenido dinámico para las páginas web. En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.
CGI ha hecho posible la implementación de funciones nuevas y variadas en las páginas web, de tal manera que esta interfaz rápidamente se volvió un estándar, siendo implementada en todo tipo de servidores web.
El CGI de MapServer utiliza generalmente los siguientes recursos:

- Un servidor http como Apache o Internet Information Server.
- El software MapServer.
- Un archivo de inicialización que active la primera vista de la aplicación.
- Un archivo de tipo ‘mapa’ (con extensión .map) que controle los datos a visualizar y/o consultar y la manera de hacerlo.
- Un archivo de tipo plantilla que controle la aplicación MapServer en la ventana del navegador (con extensión .html que puede coincidir con el de inicialización).
- Los datos espaciales.

El sistema de funcionamiento puede resumirse en el siguiente gráfico:

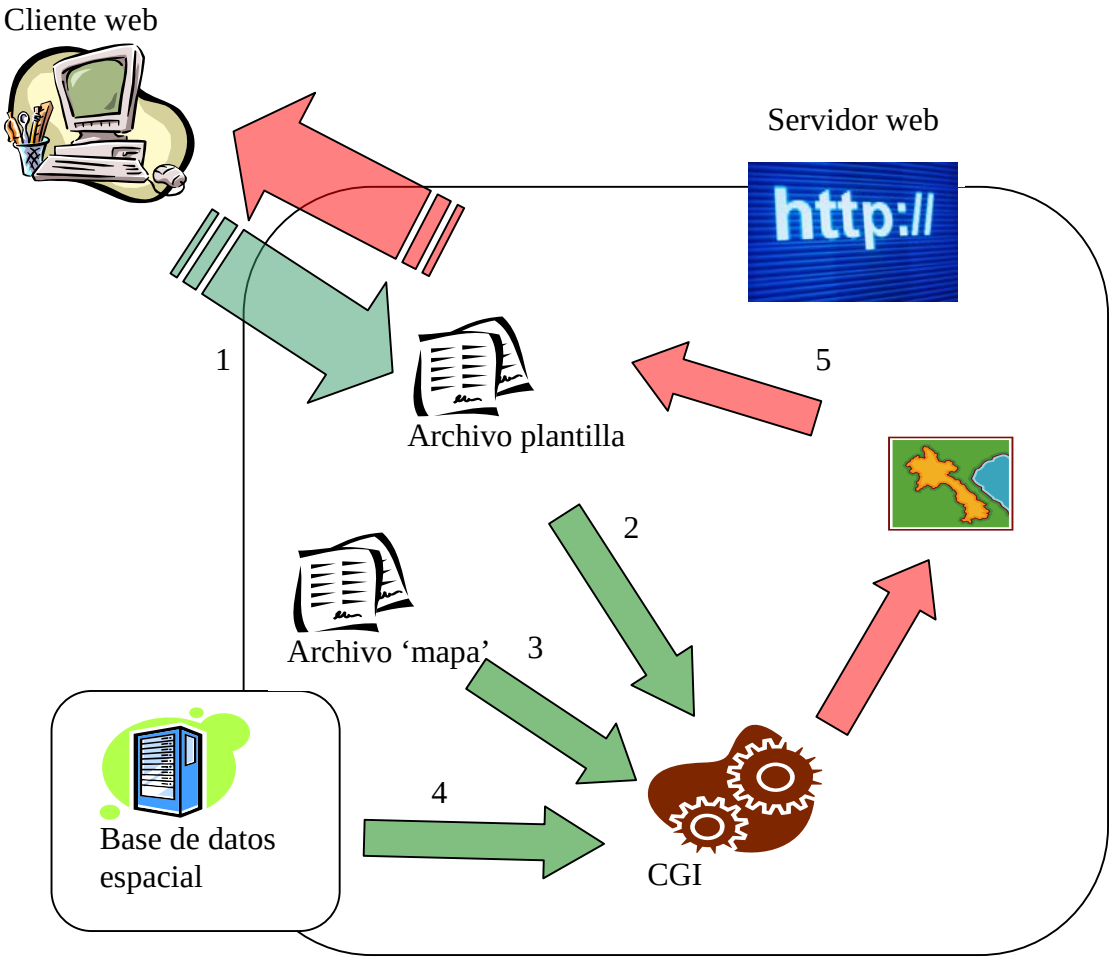


Figura 3.4 Esquema de funcionamiento del UMN Mapserver.

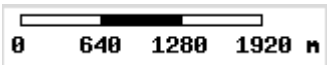
El navegador del usuario visualiza el archivo plantilla (como ya hemos dicho, un html).
El usuario manda una petición al CGI con los parámetros definidos en el archivo plantilla.
El CGI procesa la petición usando estos parámetros y la configuración del archivo 'mapa'.
Se cargan los datos geográficos creando el mapa resultante.
Por último, retorna este mapa como una respuesta al archivo plantilla y llega al navegador.

En cada sesión de trabajo MapServer

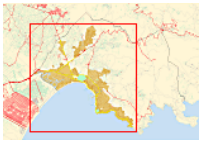
Crea un identificador de la sesión (<id>)
Si se ha especificado crea un mapa (<nombre_mapa><id>.gif)

Si se ha especificado crea una leyenda (<nombre_mapa>leg<id>.gif)

Si se ha especificado crea una barra de escala (<nombre_mapa>sb<id>.gif)



Si se ha especificado crea un mapa de referencia (<nombre_mapa>ref<id>.gif)



Donde <nombre_mapa> es el nombre del fichero de 'mapa' indicado al principio de éste.
Todas estas imágenes las crea en un directorio temporal, y aunque en la explicación son de tipo GIF, se puede definir otro tipo de imagen de salida en el fichero 'mapa'.
Por ejemplo, si el fichero 'mapa' empezara así:

```
01  NAME "ejemplo"
02  SIZE 400 300
03  IMAGECOLOR 245 245 245
04  IMAGETYPE gif
05  EXTENT …
06  …
```

La primera línea nos define el nombre del mapa y la 4 el tipo de imagen de salida. Los ficheros que se nos crearán serán del tipo:

ejemplo52158963254172.gif (el mapa)
ejemploleg52158963254172.gif (la leyenda)
ejemplosb52158963254172.gif (la barra de escala)
ejemploref52158963254172.gif (el mapa de referencia)

3.3.1. El archivo de inicialización

Este archivo puede ser parte de otro archivo html, pero por simplicidad puede ser un archivo separado. El archivo de inicialización se usa para enviar una consulta inicial al servidor http que retorna un resultado del servidor de mapas en el archivo plantilla. Alternativamente, se puede construir un hiperlink al servidor MapServer que pase los parámetros básicos requeridos por la Aplicación CGI MapServer.
Muchas veces se utiliza como una página de presentación y al hacer clic en el botón 'Iniciar' se activa el proceso de creación del mapa. Un ejemplo de este tipo de archivos es el siguiente:

Figura 3.5 Iniciación de la aplicación.

Donde las variables se pasan mediante un formulario, y al activar el botón 'iniciar' envía estos datos para que los procese MapServer. El resultado será:

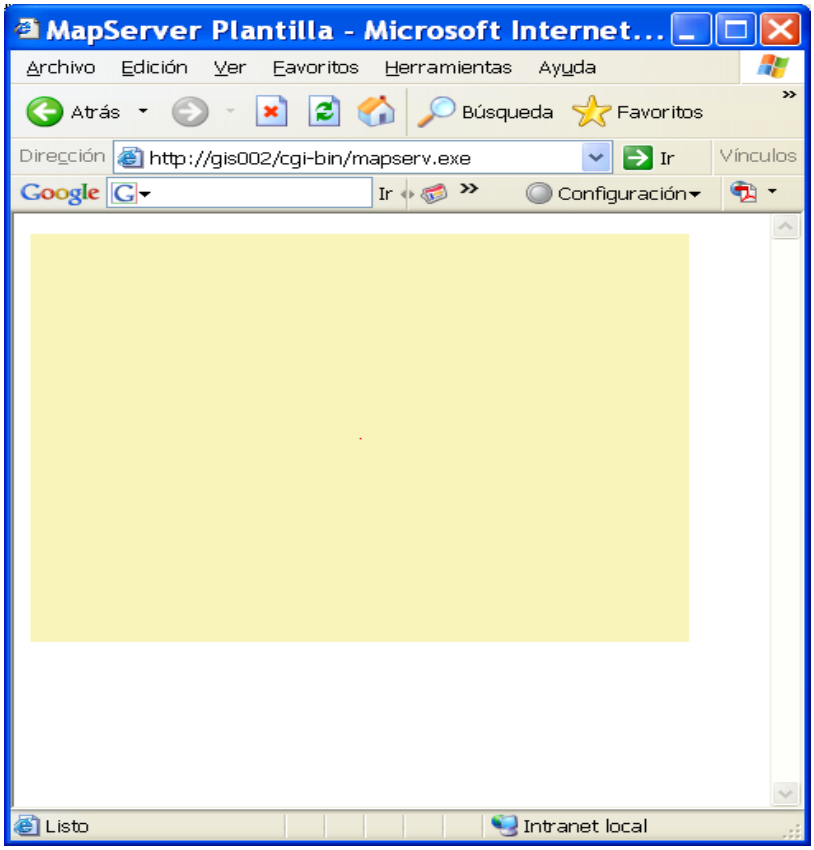


Figura 3.6 Resultado de la aplicación

Puesto que es lo que se le indica en el fichero 'mapa'.

La otra opción es crear un acceso directo que pase las variables necesarias para que MapServer las procese. Para ello se crea el acceso directo y en la dirección se hace la llamada al programa con las variables separadas por el carácter '&'.

http://gis002/cgi-bin/mapserv.exe?map=%2Fms4w%2Fapps%2Ftfc%2Fhola.map&program=%2Fcgi-bin%2Fmapserv.exe&root=%2Ftfc%2F&map_web_template=inicio.html

Escribiendo lo anterior en un navegador obtendríamos directamente el resultado final. Que será el mismo, ya que las variables son las mismas que las que se han utilizado en el fichero de inicio.
En el capítulo 3.4 se explicará con detenimiento el funcionamiento de este primer ejemplo.

3.3.2. El archivo 'mapa'

Este archivo define una colección de objetos del mapa que juntos determinan la apariencia de como será mostrado en el navegador. Un archivo 'mapa' es jerárquico. Cada archivo de este tipo define a un número de otros objetos que incluyen las capas, colores símbolos, escalas de visualización, atributos que pueden ser consultados... Existen muchos otros objetos que se describirán en los próximos capítulos.
Las definiciones en el archivo consisten en pares de palabras clave y valores. Algunos valores son listas con elementos separados por espacios:

```
01  NAME "ejemplo"
02  SIZE 400 300
03  IMAGECOLOR 245 245 245
04  IMAGETYPE gif
05  EXTENT …
```

Queda definido por el usuario como un fichero de texto, que se puede manipular con cualquier editor de texto (notepad, word...). También existen opciones gratuitas más sofisticadas como los editores UltraEdit o TexPad que disponen de ficheros de sintaxis para archivos 'map' en la dirección http://mapserver.gis.umn.edu/docs. Otra solución todavía más avanzada son los programas que crean ficheros 'map' a través de formularios como el MapEdit de MapLab. Este último es como los anteriores, totalmente gratuito y puede obtenerse en la página de Maptools: http://www.maptools.org/maplab/.

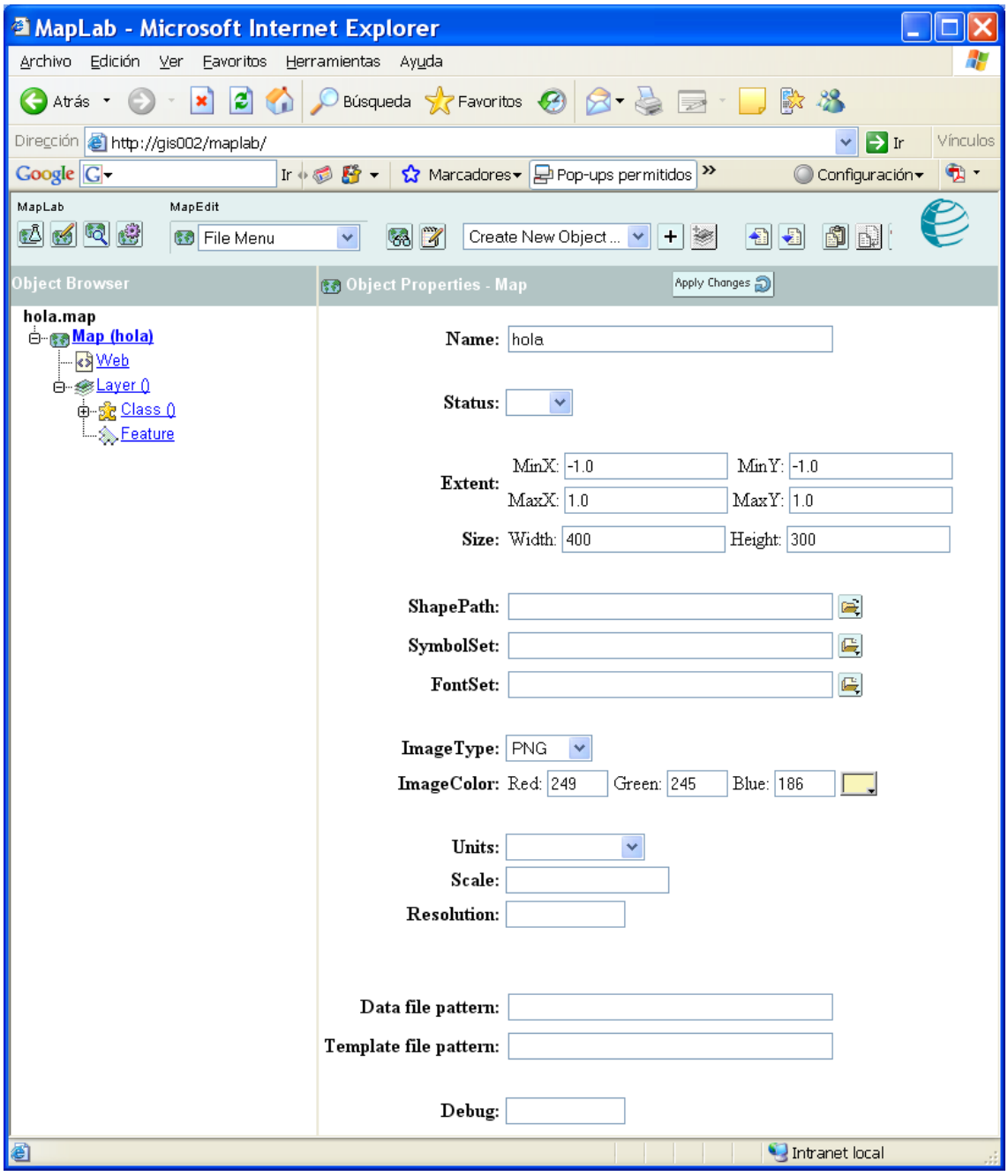


Figura 3.7 Vista del software Maptools

La estructura de objetos es bastante sencilla:

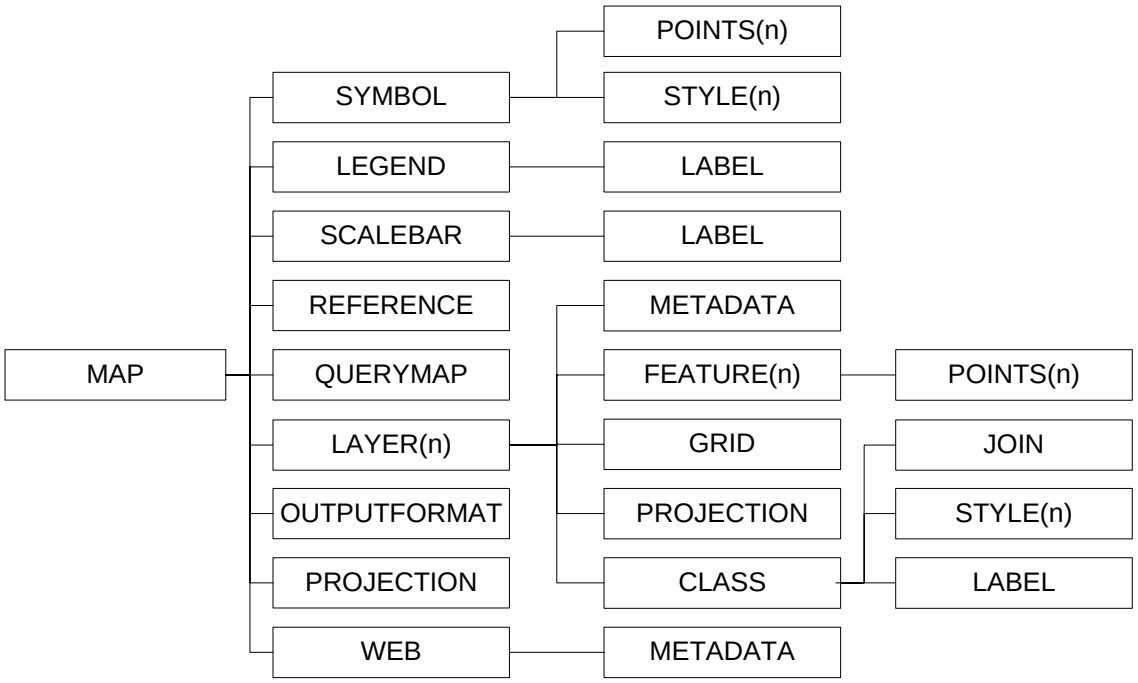


Figura 3.8 Estructura del fichero mapa

En las casillas que aparece el nombre del objeto seguido de (n) indica que en el objeto superior puede haber varios de estos objetos (en una aplicación "normal"). Es decir, en un objeto MAP puede haber 'n' objetos LAYER pero un solo objeto WEB.
El asunto se complica cuando empezamos a ver la gran cantidad de parámetros que definen cada uno de estos objetos.
El objeto MAP, por ejemplo, puede tener los siguientes parámetros:

ANLGLE
CONFIG
DATAPATTERN
DEBUG
EXTENT
FONTSET
IMAGECOLOR
IMAGETYPE
LAYER (objeto)
LEGEND (objeto)
MAXSIZE
NAME
OUTPUTFORMAT (objeto)
PROJECTION (objeto)
QUERYMAP (objeto)
REFERENCE (objeto)
RESOLUTION
SCALE
SCALEBAR (objeto)
SHAPEPATH
SIZE
STATUS
SYMBOLSET
SYMBOL (objeto)
TEMPLATEPATTERN
UNITS
WEB (objeto)

No todos son obligatorios. Dependiendo del resultado que se desee habrá que definir unos u otros. En el ejemplo introducido en el apartado anterior, el archivo 'mapa' que como vemos en la línea 10 del código de la página de inicio es:

<input type="hidden" name="map" value="/ms4w/apps/tfc/hola.map">

Si abrimos este archivo veremos que utiliza solo unos pocos de los parámetros y objetos posibles ya que el resultado que se pretende es muy simple.

```
01  #  Ejemplo-01
02  NAME "hola"
03  SIZE 400 300
04  IMAGECOLOR 249 245 186
05  IMAGETYPE png
06  EXTENT -1.0 -1.0 1.0 1.0
07  WEB
08      TEMPLATE "/ms4w/apps/tfc/plantilla.html"
09      IMAGEPATH "/ms4w/tmp/ms_tmp/"
10      IMAGEURL "/ms_tmp/"
11  END
12  LAYER
13      STATUS default
14      TYPE point
15      FEATURE
16          POINTS  0 0 END
17      END
18      CLASS
19          STYLE
20              COLOR 255 0 0
21          END
22      END
23  END
24  END
```

De momento lo más interesante que hay que entender es que hemos llegado a este fichero a través de un html de inicio (inicio.html). MapServer con las definiciones que se dan aquí crea la imagen temporal ‘hola11608213373208.png’ en el directorio ‘/ms4w/tmp/ms_tmp/’ y la envía a la plantilla ‘/ms4w/apps/tfc/plantilla.html’.

3.3.3. El archivo plantilla

El archivo plantilla es un elemento de comunicación para interactuar con el usuario vía web. Este elemento suele ser una página HTML que contiene los parámetros y las variables apropiadas que el servidor web sustituye en cada sesión de trabajo con los datos introducidos por el usuario (como datos de entrada) y los resultados del proceso CGI (como datos de salida). Lo que distingue a un archivo plantilla de una página web convencional es la presencia de variables que la relacionan con el programa CGI.
Se puede usar el mismo archivo plantilla como archivo de inicialización utilizando unos valores iniciales por defecto para las variables.

3.3.4. Conjunto de datos

Son los datos a partir de los que generaremos los mapas en las aplicaciones. Estos datos pueden ser tipo ráster o vector. También es posible conectarse a bases de datos para consultar datos alfanuméricos.
Los formatos que podemos utilizar con MapServer son muy variados, ya que podemos utilizar librerías adicionales (también gratuitas) que se instalan automáticamente con el paquete MS4W. Estas librerías son las GDAL y OGR.
Por ejemplo, los datos vectoriales soportados son entre otros: Arc/Info Binary Coverage, , DWG, DXF, ESRI Personal GeoDatabase, ESRI ArcSDE, ESRI Shapefile, GML, , Mapinfo File, MicroStation DGN, Oracle Spatial y PostgreSQL.
Se puede acceder a la lista completa en: http://gdal.maptools.org/formats_list.html y http://ogr.maptools.org/ogr_formats.html.
También es posible acceder a datos a través de las especificaciones Open Geospatial Consortium (OGC): WMS, WFS, WMC, WCS, Filter Encoding, SLD, GML y SOS. Más información en: http://www.opengeospatial.org.

3.4. Ejemplos básicos

En este capítulo crearemos dos aplicaciones: la primera de ellas será para comprobar que tenemos bien instalado el entorno MapServer y para comprender el funcionamiento de las relaciones entre el fichero mapa, la plantilla y las imágenes que crea el propio MapServer. Con el segundo ya accederemos a datos geográficos. Es muy útil partir de ejemplos tan básicos ya que posibles errores de instalación, de localización de ficheros o directorios, o de permisos se detectan enseguida. Pasemos pues al primero.

Ejemplo 01. Configuración y archivo 'map'.

Con este ejemplo no produciremos un mapa, lo que haremos es crear una imagen con elementos gráficos.
Antes de todo crearemos una carpeta donde guardar los ficheros necesarios para la aplicación. Lo haremos en la siguiente dirección: C:\ms4w\apps\tfc.

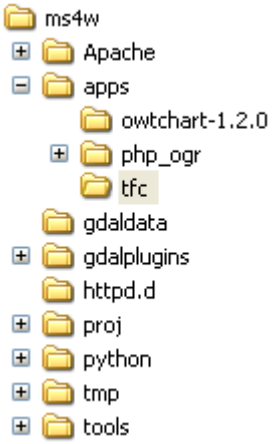


Figura 3.9 Estructura de directorios de la aplicación

Ahora hay que informar al servidor Apache que queremos crear una aplicación ahí y que queremos que esta carpeta sea accesible vía web. Para ello crearemos un fichero de texto que llamaremos 'httpd_tfc.conf', lo guardaremos en la carpeta C:\ms4w\httpd.d\ y contendrá lo siguiente:

```
Alias /tfc/ "/ms4w/apps/tfc/"

<Directory "/ms4w/apps/tfc/">
  AllowOverride None
  Options Indexes FollowSymLinks Multiviews
  Order allow,deny
  Allow from all
</Directory>
```

Para que estos cambios tengan efecto deberemos reiniciar el Apache. Con la instalación del paquete MS4W viene un programa que reinicia apache automáticamente. Está en la carpeta c:\ms4w\ y se llama 'apache-restart.bat'. Ejecutamos este programa y si todo

está correctamente, al escribir en el navegador web lo siguiente: http://gis002/tfc/ nos deberá aparecer la vista del directorio desde el navegador:

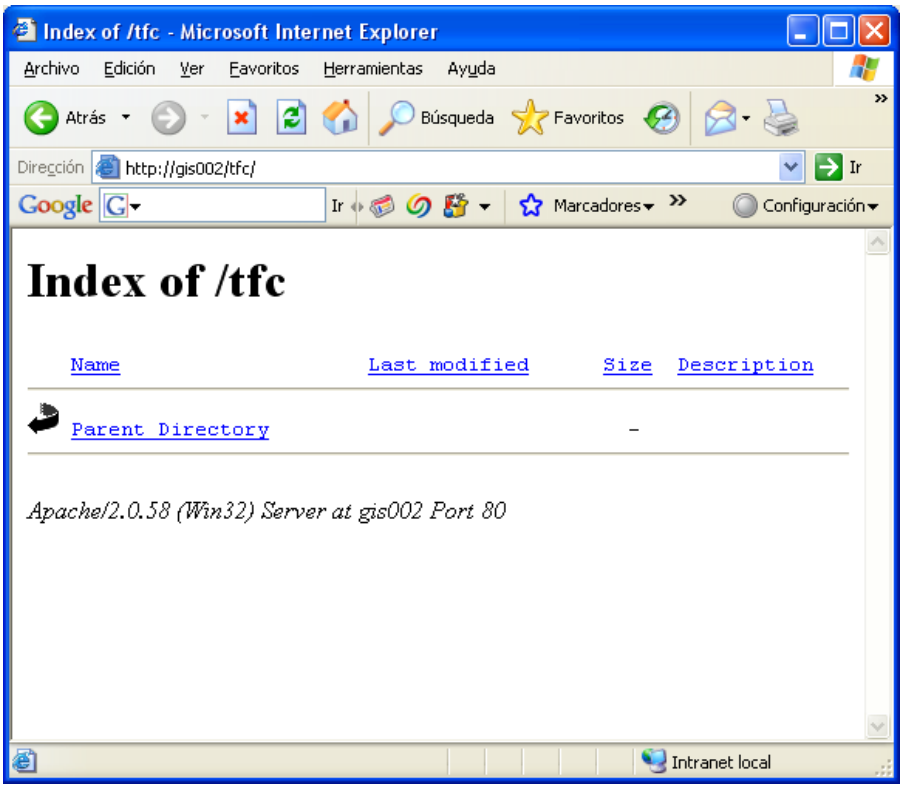


Figura 3.10 Vista de la carpeta a través del navegador

Como lo acabamos de crear está vacío, por supuesto.
Ahora ya podemos crear el fichero 'mapa'. Ya se ha señalado anteriormente que existe software que nos facilita el trabajo a la hora de crear este fichero, pero de momento lo desarrollaremos con el 'notepad' o bloc de notas de Windows. Abrimos un fichero (en el directorio 'tfc' que hemos creado anteriormente) al que llamamos 'hola.map' y escribimos:

```
01  #  Ejemplo-01
02  NAME "hola"
```

Los números al inicio de cada línea no forman parte del fichero, son una ayuda para comentar cada línea por lo que no hay que escribirlos.
La línea 01 es un comentario. Se pueden usar en cualquier lugar del fichero y lo que hace MapServer es ignorar todo lo que hay escrito detrás de este símbolo.
La línea 02 indica el nombre que tendrán todas las imágenes que creará MapServer a través de este archivo.
Las siguientes líneas son:

```
03  SIZE 400 300
04  IMAGECOLOR 249 245 186
05  IMAGETYPE png
06  EXTENT -1.0 -1.0 1.0 1.0
```

La línea 03 especifica el tamaño (en píxeles) de la imagen final. Con IMAGECOLOR definimos el color de fondo de la imagen y con IMAGETYPE el formato de esta imagen final. En este caso png. En la línea 04 definimos la extensión geográfica de la imagen. Es un área rectangular marcada por las coordenadas de dos lados opuestos (el inferior izquierdo y el superior derecho).

```
07  WEB
08      TEMPLATE "/ms4w/apps/tfc/plantilla.html"
09      IMAGEPATH "/ms4w/tmp/ms_tmp/"
10      IMAGEURL "/ms_tmp/"
11  END
```

Entre la línea 07 y la 11 tenemos el primer objeto. Todos los objetos quedan definidos por su nombre y la palabra END. Entre estas dos palabras clave se especifican todos sus parámetros. Este objeto nos permitirá visualizar la imagen creada por MapServer, insertándola en una página web (fichero plantilla) lo que nos facilitará agregar controles para el mapa (como los zooms que veremos más adelante) y otra información que creará MapServer.
El parámetro TEMPLATE especifica el nombre y la localización de esta plantilla. Con IMAGEPATH informamos a MapServer donde debe poner las imágenes que crea y IMAGEURL dirige al navegador a la carpeta donde están estas imágenes. Está claro que se refieren a la misma carpeta, pero la diferencia es que IMAGEPATH lo procesa el CGI mientras que IMAGEURL lo hace el navegador.
Hasta ahora MapServer conoce el tipo de imagen que debe producir, el tamaño, el color de fondo, y como visualizar el mapa en una plantilla.
Ahora especificaremos que es lo que queremos que MapServer dibuje. Esto lo haremos con el objeto LAYER. Este objeto hace referencia a un conjunto de elementos que se 'dibujaran' juntos a una escala determinada y usando una proyección determinada. Como todos los objetos, empezará por la palabra clave LAYER y finalizará con la palabra END. Añadiremos las siguientes líneas a nuestro fichero hola.map:

```
12  LAYER
13      STATUS default
14      TYPE point
```

Con el parámetro STATUS determinamos cuando se debe dibujar la capa en cuestión. Los valores posibles son ON, OFF y DEFAULT. Con la opción DEFAULT la dibujará siempre. Con ON la dibujará, pero nos permitirá cambiar a OFF que no nos la dibujará y viceversa.
Cada LAYER tiene un tipo de datos asociado como por ejemplo POINT, LINE o POLYGON. Existen más tipos de LAYER que se irán describiendo a lo largo del texto. De momento vamos a crear una LAYER de tipo puntual.
En este ejemplo no utilizaremos datos geográficos reales, pero para ver el funcionamiento crearemos los elementos gráficos artificialmente (serán elementos virtuales, que se generarán cada vez que ejecutemos la aplicación). Las siguientes líneas corresponden al objeto FEATURE que nos crearan estos objetos artificiales:

```
15  FEATURE
16          POINTS  0 0 END
```

```
17  END
```

Estas tres líneas nos crean un punto en las coordenadas 0,0. Si el tipo de la LAYER fuera LINE tendríamos que definir la línea con una serie de pares de coordenadas y si quisiéramos un polígono lo mismo, pero teniendo en cuenta que el punto inicial y el final deben ser el mismo.
En cada objeto LAYER se pueden definir diversos objetos CLASS que nos permitirán seleccionar un conjunto de elementos de la LAYER para visualizarlos de diferente forma. Si no se utiliza ningún criterio de selección dibujará todos los elementos de la LAYER. La manera hacer estas selecciones la trataremos en capítulos siguientes.

```
18  CLASS
19          STYLE
20              COLOR 255 0 0
21          END
22  END
```

Como veíamos en el esquema del apartado 4.3.2, el objeto class puede contener a su vez varios objetos: JOIN, STYLE y LABEL. En este ejemplo simple utilizaremos el STYLE que nos define las características del símbolo que usaremos para dibujar los elementos de este objeto CLASS. Definimos únicamente el color con el que queremos visualizar estos elementos. El color se define por los tres componentes RGB separados por espacios. Estos tres componentes varían entre 0 y 255. En la línea 20 definimos el color rojo (R:255, G:0, B:0) para dibujar nuestros elementos.
Para acabar tenemos que ‘cerrar ‘ el objeto LAYER y el objeto MAP con sendos END.

```
23  END
24  END
```

Ejemplo 01. Los archivos de inicio y plantilla.

En el apartado 3.3.1 nos habíamos introducido en el funcionamiento de los archivos de inicio y plantilla. En este ejemplo serán dos archivos diferentes: inicio.html y plantilla.html.
En el mercado existen aplicaciones que ayudan a editar este tipo de ficheros como Macromedia Dreamweaver, pero como con el archivo ‘mapa’ se puede utilizar un editor de texto corriente. Empezaremos abriendo un fichero en la carpeta de trabajo C:\ms4w\apps\tfc al que llamaremos inicio.html y escribimos:

```
01  <html>
02  <head> <title>MapServer</title></head>
03  <center>
04  <h2>P&aacute;gina inicial de MapServer </h2>
05  </center>
06  Ejemplo de p&aacute;gina de inicio
07  <body>
08     <form method=POST action="/cgi-bin/mapserv.exe">
09        <input type="submit" value="Iniciar">
10        <input type="hidden" name="map" value="/ms4w/apps/tfc/hola.map">
```

```
11     </form>
12 </body>
13 </html>
```

Si ejecutamos el fichero desde el navegador haciendo http://gis002/tfc/inicio.html el resultado será:

Figura 3.11 Página de inicio

Con formulario y el botón 'iniciar' que creamos, al hacer clic en él lo que hacemos es utilizar el ejecutable (el CGI) MapServer y pasarle la variable de la línea 10, que le indica el fichero 'mapa' que debe utilizar (el fichero hola.map creado anteriormente). MapServer leerá el contenido de hola.map, lo interpretará y producirá la imagen correspondiente (un punto de color rojo en este caso). Como ya hemos visto, el nombre de esta imagen se crea concatenando el nombre del mapa definido en el fichero 'map', un número identificativo generado por el sistema y la extensión también definida en el fichero 'map'. Esta imagen la guarda en el sitio indicado por el parámetro IMAGEPATH (/ms4w/tmp/ms_tmp/). Si miramos en esta carpeta una vez clicado el botón iniciar veremos un fichero como este:

hola11610029884924.png

Seguidamente MapServer lee el fichero plantilla: plantilla.html como se le indica en el fichero 'mapa':

```
01 <html>
```

```
02  <head> <title>MapServer Plantilla</title></head>
03  <body>
04        <IMG SRC="[img]" width=400 height=300 border=0>
05  </body>
06  </html>
```

y sustituye la variable [img] por el valor de la ruta especificada en el parámetro IMAGEURL del fichero 'map' (/ms_tmp/) seguido del nombre de la imagen que ha creado (hola11610029884924.png). Así, el fichero plantilla se transformará en:

```
01  <html>
02  <head> <title>MapServer Plantilla</title></head>
03  <body>
04        <IMG SRC="/ms_tmp/hola11610029884924.png" width=400 height=300
    border=0>
05  </body>
06  </html>
```

Después de leerlo y de sustituir las variables que encuentra (en este caso solamente [img]), MapServer envía el contenido del fichero plantilla (el modificado) a Apache para que lo reenvíe al navegador.
Al hacer clic en el botón 'inicio' de la página inicio.html obtenemos el siguiente resultado:

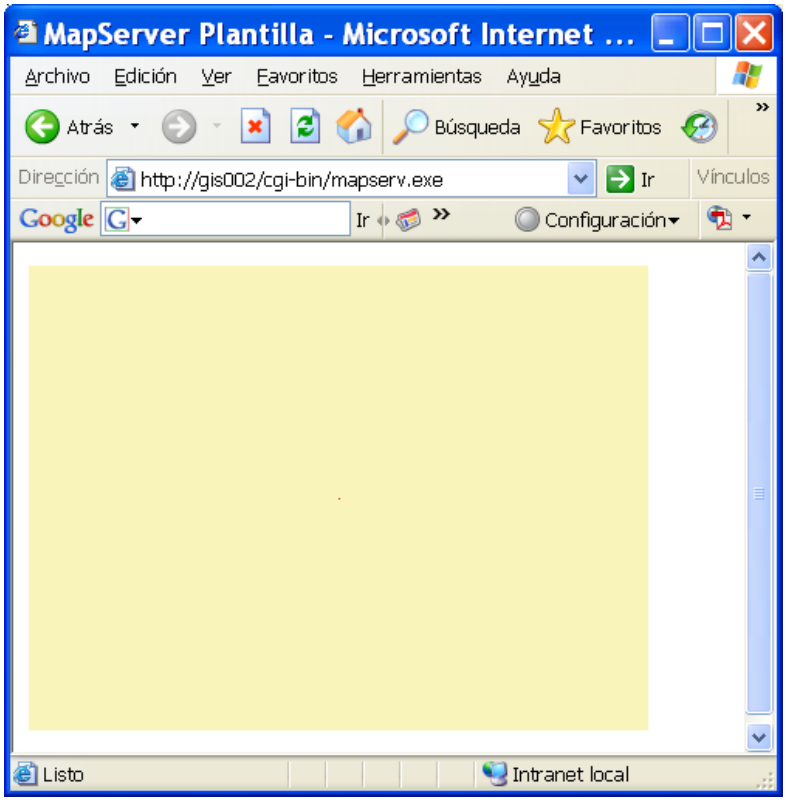


Figura 3.12 Resultado de la aplicación

Si todo está configurado y escrito convenientemente, podremos ver un rectángulo amarillo de 400x300 píxeles con un pequeño punto rojo en el centro.

Si cambiamos las siguientes líneas del archivo ‘mapa’

```
12  LAYER
13      STATUS default
14      TYPE line
15      FEATURE
16          POINTS  0 0 0 1 END
17      END
18      CLASS
19          STYLE
20              COLOR 255 0 0
21          END
22      END
23  END
```

Lo que en vez de un punto nos dibujará una línea definida por los puntos de coordenadas (0,0) y (0,1), y el resultado será:

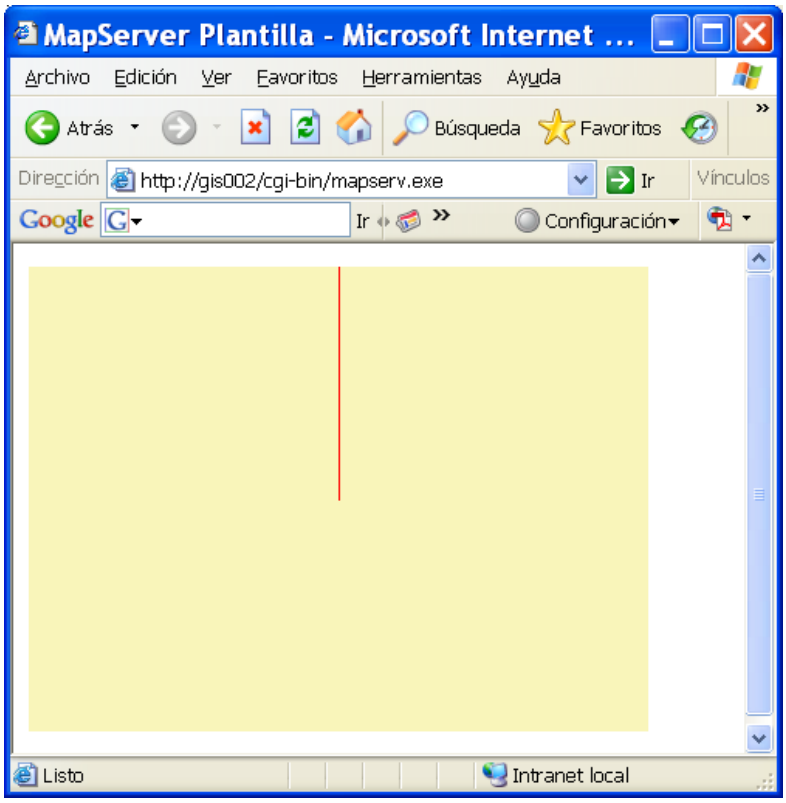


Figura 3.13 Resultado de la aplicación

Todos los cambios que se realizan en el fichero ‘mapa’ pueden probarse con éste abierto en el editor, no hace falta cerrarlo para que MapServer lo lea. Lo único que hay que tener presente es guardar los cambios.

Ejemplo 02. Acceso a datos geográficos

La aplicación que crearemos en este apartado es más compleja que la previa y producirá un mapa real. Se usarán muchos elementos descritos en las anteriores y algunos nuevos necesarios para usar datos espaciales.
El objetivo es crear un mapa de las manzanas catastrales del municipio de Roses. Para ello disponemos de las manzanas en formato SHP en los ficheros:

Masa.shp
Masa.dbf
Masa.shx

Antes de empezar, colocamos estos ficheros en una nueva carpeta dentro de nuestra carpeta de trabajo (tfc) y que llamaremos 'datos' (C:\ms4w\apps\tfc\datos\).
Ahora creamos un nuevo archivo 'map' en la carpeta de trabajo 'tfc' y al que llamaremos 'ejemplo02.map' y escribimos lo siguiente:

```
01  NAME "ejemplo02"
02  SIZE 400 300
03  IMAGECOLOR 196 240 255
04  IMAGETYPE png
05  EXTENT 510720 4676215 519430 4683355
06  SHAPEPATH "datos"
```

El inicio del nombre de las imágenes creadas a partir de este archivo será 'ejemplo02'. Como en el ejemplo anterior, el mapa tendrá un tamaño de 400x300 píxeles. Se ha cambiado el color de fondo del mapa a una tonalidad de azul (R:196, G:240, B:255) y el tipo de imagen de salida se mantiene en 'png'.
En la línea 05 definimos ya en coordenadas reales (UTM en este caso) la extensión geográfica del mapa con dos pares de coordenadas: la X e Y mínimas y la X e Y máximas. Estas son las coordenadas de un rectángulo que contiene el área urbana de Roses, que es precisamente donde se encuentran las manzanas catastrales que queremos dibujar.
En la línea 06 y con el parámetro SHAPEPATH indicamos a MapServer el directorio donde tiene que encontrar los ficheros que contienen los datos geográficos. Es una ruta relativa a la localización del fichero 'mapa'. Si los datos y el fichero 'mapa' estuvieran en el mismo directorio no haría falta poner este parámetro.
El objeto WEB, definido por las siguientes líneas lo dejaremos igual que en ejemplo anterior:

```
07  WEB
08      TEMPLATE "/ms4w/apps/tfc/plantilla.html"
09      IMAGEPATH "/ms4w/tmp/ms_tmp/"
10      IMAGEURL "/ms_tmp/"
11  END
```

Lo que quiere decir que utilizaremos la misma plantilla para mostrar los resultados y los mismos directorios para crear las imágenes.
El siguiente paso es definir que elementos dibujar y la manera de hacerlo con el objeto LAYER:

```
12  LAYER
13    NAME "MASA"
14    STATUS DEFAULT
15    TYPE POLYGON
16    DATA "MASA"
17    LABELITEM "MASA"
```

Con el parámetro NAME especificamos el nombre de la LAYER (en este caso ‘masa’). Esto proporciona la relación entre este objeto LAYER y la página web. Como veremos en ejemplos más avanzados, desde la página web (la plantilla) podremos decidir que objetos LAYER que remos visualizar a través de su nombre.
En la línea 16 y con DATA identificamos el nombre (sin la extensión shp) de los archivos shp que se dibujaran con este objeto LAYER (esto es así porque MapServer interpreta por defecto que los datos están en este formato). Recordemos que en la línea 06 definimos el directorio donde se encuentran estos archivos.
Con STATUS DEFAULT nos dibujará siempre el contenido de los ficheros shp de este objeto LAYER.
Cada capa de información (LAYER) tiene asociado un tipo de datos especificado con TYPE. Este parámetro determina como MapServer tiene que interpretar los datos geográficos asociados a la capa. Los valores posibles son:

POINT: punto
LINE: linea
POLYGON: polígono
ANNOTATION: etiquetas. Se usa para etiquetar elementos (puntos, líneas o polígonos) sin dibujarlos.
RASTER: Dibuja imágenes georreferenciadas.
QUERY: Se usa para asociar un punto que se ha seleccionado del mapa con un conjunto de datos. Una LAYER de este tipo no se dibuja, pero es posible consultar sus atributos.

Hemos visto anteriormente que nuestro conjunto de datos espaciales se componía de las manzanas catastrales definidas por tres ficheros: masa.shp, masa.shx y masa.dbf. En los dos primeros se describe la geometría y el tercero es un fichero de dBASE con atributos. Cada columna de este último tiene un nombre. Especificando en la línea 17 el valor ‘MASA’ para el parámetro LABELITEM, las etiquetas de cada manzana se crearan a partir de los valores de esta columna.
Este conjunto de datos tiene los siguientes campos:

MSLINK
MAPA
DELEGACIO
MUNICIPIO
MASA
HOJA
TIPO
COORX
COORY
AREA
PERIMETRE
NUMSYMBOL
FECHAALTA
FECHABAJA
UTM

El campo MASA contiene las cinco primeras posiciones de la referencia catastral de las manzanas.

Ahora solo nos quedará definir de que modo queremos representar estos datos geográficos.

```
18    CLASS
19      STYLE
20        COLOR 254 226 197
21        OUTLINECOLOR 255 0 0
22      END
23      LABEL
24        SIZE SMALL
25        COLOR 0 0 0
26       END
27     END
```

Dentro del objeto CLASS creamos un objeto STYLE donde especificamos el color de la línea perimetral (OUTLINECOLOR) y el color de relleno (COLOR) de los polígonos. No es necesario que existan ambos. Podemos poner el primero, el segundo o los dos. En este caso nos dibujará los polígonos de un color carne y con una línea perimetral roja.
También dentro del objeto CLASS creamos un objeto LABEL donde definimos el tamaño y color de la fuente para etiquetar las manzanas.
El objeto LABEL, la igual que los demás objetos, tiene muchos más parámetros que iremos viendo paulatinamente. Si no se definen, MapServer toma los valores por defecto.
Ahora nos faltará cerrar los objetos LAYER y MAP con sendos END:

```
28   END
29 END
```

Hemos visto que se ha utilizado el mismo archivo plantilla que en el ejemplo anterior para visualizar el resultado, por lo que no ha hecho falta crear uno nuevo. No será lo mismo para el archivo de inicio, puesto que se ha cambiado el archivo ‘mapa’. Vamos pues a crear uno nuevo:
Creamos en la carpeta de trabajo ‘tfc’ un fichero html de inicio que contendrá:

```
01 <html>
02 <head> <title>MapServer</title></head>
03 <center>
04 <h2>P&aacute;gina inicial de MapServer </h2>
05 </center>
06 Ejemplo 02
07 <body>
08    <form method=POST action="/cgi-bin/mapserv.exe">
09       <input type="submit" value="Iniciar">
10       <input type="hidden" name="map"
   value="/ms4w/apps/tfc/ejemplo02.map">
11    </form>
12 </body>
13 </html>
```

Que como podemos comprobar es idéntico al del ejemplo_01 salvo en la línea 06 que únicamente se cambia el texto, y la línea 10 donde se cambia el nombre del archivo 'mapa'. Veámoslo gráficamente:

Figura 3.14 Página de inicio del ejemplo 02

Y si todo está correcto, al pinchar en el botón 'iniciar' tendremos el siguiente resultado:

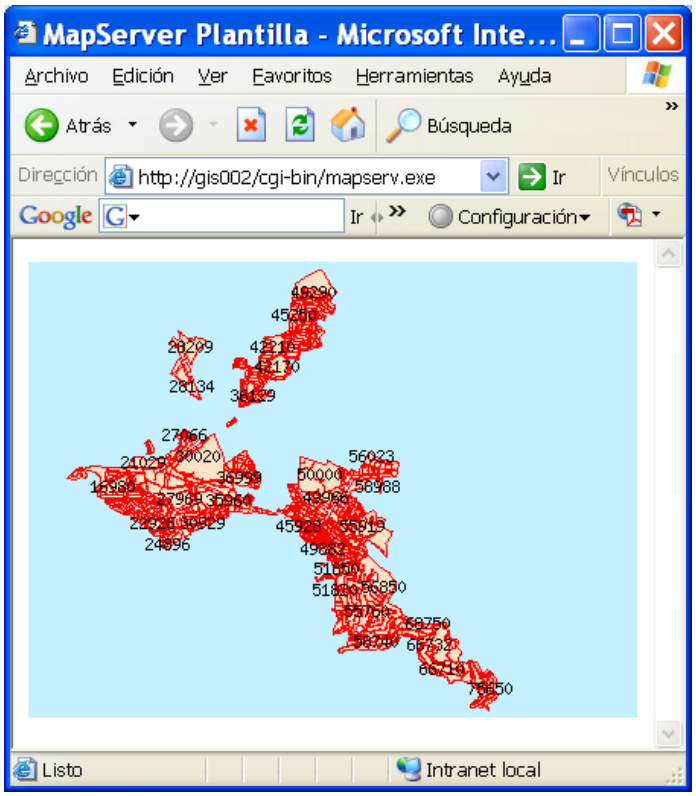


Figura 3.15 Resultado del ejemplo 02

En total hay 610 manzanas catastrales, pero vemos en el resultado que aunque los polígonos nos los ha dibujado todos, no ha ocurrido lo mismo con las etiquetas con la referencia catastral, que solo ha dibujado unas pocas. Esto es debido a uno de estos parámetros que no hemos especificado en el fichero 'map' pero que tiene un valor por defecto. Es el parámetro FORCE del objeto LABEL que por defecto tiene un valor de FALSE, y nos indica si queremos que nos dibuje todas las etiquetas aunque se solapen.

3.5. Ejemplo 03. Creación de una aplicación

En el capítulo anterior creamos una mini aplicación que nos permitía visualizar un mapa de las manzanas catastrales con su referencia como etiqueta. Este mapa no tenía mucha utilidad puesto que la información era muy limitada, no se veía toda, no nos podíamos acercar (hacer zoom) ni movernos por él.
Con el objetivo de perfeccionar un poco la aplicación, haciéndola más útil y visualmente más atractiva tendremos que incrementar la complejidad de los archivos 'mapa' y plantilla.
Al primero de éstos le añadiremos más capas de información. Si hasta ahora únicamente habíamos trabajado con una (manzanas catastrales), ahora añadiremos las capas siguientes:

Nombre	Geometría
Parcelas catastrales	Polígonos
Aceras	líneas
Ortofoto	Imagen Ráster

La gran mejora del archivo plantilla será la posibilidad de navegar por el mapa, haciendo zooms y desplazamientos. También nos permitirá controlar que capas de información queremos ver en cada momento. Empecemos por el primero:

3.5.1. Archivo 'mapa'

```
01  NAME "ejemplo03"
02  SIZE 400 300
03  IMAGECOLOR 196 240 255
04  IMAGETYPE png
05  EXTENT 510720 4676215 519430 4683355
06  SHAPEPATH "datos"
07  WEB
08      TEMPLATE "/ms4w/apps/tfc/ejemplo03.html"
09      IMAGEPATH "/ms4w/tmp/ms_tmp/"
10      IMAGEURL "/ms_tmp/"
11  END
```

Estas once primeras líneas son idénticas a las del ejemplo 02, salvo la primera, en la que definimos el nombre del mapa así que no haremos ninguna consideración más.

Ahora continuaremos con los objetos LAYER. MapServer dibuja estos objetos secuencialmente, empezando por el primero que aparece en el fichero 'mapa', cada capa se dibuja encima de la que le precede hasta llegar a la última del fichero. Esto significa que los objetos dibujados de capas definidas al principio del fichero pueden ser sobrescritos por objetos definidos al final del fichero. Esto debe tenerse en cuenta a la hora de definir estos objetos, ya que si por ejemplo definimos un objeto RASTER al final del fichero 'mapa' nos ocultará todo lo que hayamos definido antes que él. Así pues, empezaremos a definir las capas que irán 'debajo'.

```
12  LAYER
13      NAME "orto25m"
14      STATUS ON
15      TYPE RASTER
16      DATA "orto25m.tif"
17  END
```

Disponemos de una imagen georreferenciada (GeoTiff) del municipio que está colocada en el mismo directorio de datos de los ejemplos anteriores (C:\ms4w\apps\tfc\datos). En la línea 13 definimos el nombre de la capa y en la siguiente indicamos que estará visible, pero no siempre. De esta manera habilitamos la posibilidad de ocultarla a través del fichero plantilla, mediante el cual podemos cambiar de ON a OFF este parámetro. Es en la línea 15 donde se especifica el tipo de datos del objeto, que como hemos dicho será RASTER. Y ya el último parámetro que falta es el nombre del fichero de datos, indicado en la línea 16. Puesto que en la línea 06 hemos definido el directorio donde guardaremos todos los datos, únicamente hay que poner el nombre del fichero.
Encima de esta ortofotografía insertaremos la LAYER 'manzanas catastrales' del anterior ejemplo sin cambiar nada:

```
18  LAYER
19    NAME "MASA"
20    STATUS ON
21    DATA "MASA"
22    TYPE POLYGON
23    LABELITEM "MASA"
24    CLASS
25      STYLE
26        COLOR 254 226 197
27        OUTLINECOLOR 255 0 0
28      END
29      LABEL
30        SIZE SMALL
31        COLOR 0 0 0
32      END
33    END
34  END
```

Y ahora encima de las dos anteriores dibujaremos la capa 'parcelas catastrales'. Ésta es muy similar a la anterior. Los datos están contenidos en los ficheros: parcela.shp, parcela.dbf y parcela.shx, también ubicados en el directorio de datos.

```
35  LAYER
36    NAME "PARCELA"
37    STATUS ON
38    DATA "PARCELA"
39    TYPE POLYGON
40    LABELITEM "PARCELA"
41    CLASS
42      STYLE
43        OUTLINECOLOR 255 0 0
44      END
45      LABEL
46        SIZE SMALL
47        COLOR 0 0 255
48      END
49    END
50  END
```

El nombre de la capa será 'PARCELA', igual que en la capa RASTER, será visualizable pero se puede cambiar, los datos geográficos también se llaman parcela (.shp, .dbf, .shx). Entre las líneas 41 y 49 definimos de que manera vamos a visualizar estas parcelas: una línea exterior de color rojo y una etiqueta de tamaño pequeño y color azul. Los textos de esta etiqueta los adquirirá del campo 'PARCELA' del archivo dbf. Estas etiquetas son los dos últimos dígitos de la primera parte de la referencia catastral, que identifican a la parcela.
La última capa que crearemos serán las aceras. Los datos están en los ficheros elemlin.shp, elemlin.dbf y elemlin.shx en la carpeta de datos. Comprobamos que tiene el mismo estilo que las anteriores, y el único cambio es el tipo de datos (línea 55) que es LINE. El color en que las visualizaremos será el gris claro (línea 58).

```
51  LAYER
52    NAME "aceras"
53    STATUS ON
54    DATA "elemlin"
55    TYPE LINE
56    CLASS
57      STYLE
58        COLOR 192 192 192
59      END
60    END
61  END
```

Por último solo nos queda cerrar el objeto 'MAP' con un:

```
62  END
```

3.5.2. Archivo plantilla

En este ejemplo veremos como se vuelve más complejo este archivo para permitirnos interactuar con el mapa mediante la substitución de variables.

Haremos un diseño rudimentario para colocar en el navegador los controles necesarios. Dividiremos en dos columnas la página. La parte izquierda será la zona del mapa y la parte de la derecha la zona de los controles:

Figura 3.16 Diseño de la interfaz de usuario

Abrimos un fichero html en la carpeta 'tfc' al que llamaremos 'ejemplo03.html' y empezamos escribiendo un inicio similar al de los anteriores ejemplos:

```
01 <html>
02 <head>
03   <title>MapServer - GIS</title>
04 </head>
05 <body>
```

Desde la línea 06 hasta la 25 creamos un formulario que permite al usuario interactuar con MapServer cambiando los valores de las variables CGI contenidas en él. Estas variables están entre corchetes ([]) y la primera que encuentra es [program] en la línea 06. Recordemos el funcionamiento de MapServer: cuando lo invocamos lee el fichero 'mapa' especificado en el archivo de inicio, crea el mapa, lee el fichero plantilla y hace las substituciones pertinentes. Mientras que en los ejemplos anteriores el archivo plantilla solo tenía una variable a sustituir, la [img], en éste hay alguna más ahora describiremos.
Al final del capítulo, cuando creemos el archivo de inicio para este ejemplo, estableceremos el valor "/cgi-bin/mapserv.exe" para la variable 'program' (como en los anteriores ficheros de inicio). Entonces, cuando MapServer lea el archivo plantilla y encuentre la variable [program] la substituirá por "/cgi-bin/mapserv.exe".

```
06 <form method="GET" action="[program]" name="mapserv">
07 <table border="1">
08   <td align="center">
09   <input type="image" name="img" src="[img]" width="[mapwidth]"
   height="[mapheight]" border="0">
10   </td>
```

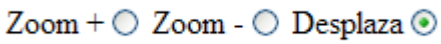
También substituirá la variable [img] de la línea 09 por el nombre de la imagen que se ha creado y las variables [mapwidth] y [mapheight] por los valores de anchura y altura del mapa indicados en la línea 02 del fichero 'mapa'. Adviértase que en esta línea 09 el objeto [img] es de tipo 'input' (entrada de valores). Esto es lo que permite la navegación

por el mapa. Si se hace clic en la imagen, las coordenadas de ese punto se devuelven a MapServer, que coloca al punto en el centro de la imagen.

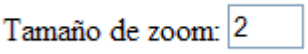
La columna de la derecha, la de los controles será:

```
11  <td valign="center" >
12     <center><input type="submit" value="Actualizar"></center>
13     <p align="center">Zoom +<input type=radio name=zoomdir  value=1
    checked [zoomdir_1_check]>
14        Zoom -<input type=radio name=zoomdir value=-1 [zoomdir_-1_check]>
15        Desplaza<input type=radio name=zoomdir value=0 [zoomdir_0_check]>
16     <p> Tamaño de zoom: <input type=text name=zoomsize size=2
    value=[zoomsize]>
```

Con la línea 11 empezamos esta segunda columna. Creamos un botón para actualizar el mapa. Lo único que hace es volver a procesar la página substituyendo otra vez las variables.
Desde la línea 13 hasta la 16 son las que proveerán las capacidades de zoom de la aplicación. La variable ‘zoomdir’ determinará la dirección del zoom.
El lenguaje html dispone de varios controles visuales para la introducción de valores a las variables y en este documento se verán varios de estos. El elegido en este caso es el tipo ‘radio’, que nos permite asignar a la variable ‘zoomdir’ uno de estos valores: 1, -1 y 0. Este tipo de control es excluyente, lo que quiere decir que solo se puede asignar uno de los tres. Visualmente quedará:

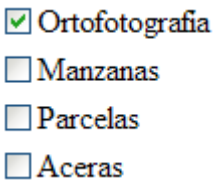


Asignando a ‘zoomdir’ el valor ‘1’ significa que cuando se actualice el mapa (con el botón de actualizar o clicando en la imagen), se hará un zoom + (nos acercaremos) a la vista siguiente una cantidad determinada por la variable [zoomsize] especificada en la línea 16 y definible por el usuario:



Por defecto toma el valor de 2. Esto hace que si asignamos el valor 1 a ’zoomdir’ y el valor 2 a ‘zoomsize’ nos acercaremos (zoom +) al mapa el doble de lo que estamos y centrando la imagen en el punto donde se ha hecho el clic. Análogamente, si el valor de ‘zoomdir’ es –1 nos alejaremos del mapa la cantidad ‘zoomsize’. Si tenemos en cuenta las escalas de los mapas, en el primer caso si el mapa está a escala 1:1000 al hacer clic en el mapa la escala se transformará en 1:500. En el caso que ‘zoomdir’ valga 0 (desplazar) se mantendrá la escala de la imagen siguiente.
La variable [zoomdir_1_check] de la línea 13 representa el estado de la selección del control radio de la variable ‘zoomdir’. Si está activado el botón de valor 1, MapServer reemplaza la variable [zoomdir_1_check] por la cadena ‘checked’. De la misma forma actuaría con las variables [zoomdir_-1_check] y [zoomdir_0_check].
Las líneas 17 hasta la 25 definen otro tipo de controles html, los llamados ‘checkbox’ que son casillas de verificación. Lo que hacen es dar valores a la variable [layer]. El

valor de cada una es el nombre de una de las capas (objetos LAYER) del fichero 'mapa'. Este no es un control excluyente por lo que podremos activar más de uno a la vez. Si queremos visualizar una capa, el checkbox asociado a esta capa tiene que estar marcado.
Las variables del tipo: nombre de capa+'_'+checked (por ejemplo [MASA_check]) funcionan igual que las homónimas del control radio descrito anteriormente.
El aspecto visual de este control es:



Y el código:

```
17      <hr size="1">
18      <table>
19        <tr><td colspan="3"><input type="checkbox" name="layer"
   value="orto25m" [orto25m_check]>Ortofotografía</td></tr>
20        <tr><td colspan="3"><input type="checkbox" name="layer"
   value="MASA" [MASA_check]>Manzanas</td></tr>
21        <tr><td colspan="3"><input type="checkbox" name="layer"
   value="PARCELA" [PARCELA_check]>Parcelas</td></tr>
22        <tr><td colspan="3"><input type="checkbox" name="layer"
   value="aceras" [aceras_check]>Aceras</td></tr>
23      </table>
24    </td>
25  </table>
```

Finalmente, las siguientes líneas, que asignan valores a las variables del formulario html permiten a comunicarse con MapServer para 'pasar' los valores de que fichero 'mapa' leer (variable 'map'), que extensión de mapa mostrar (variable 'imgext' y cuál es el centro de la imagen (variable 'imgxy'), que hacen posible interactuar con el mapa.

```
26  <input type="hidden" name="imgxy" value="[center]">
27  <input type="hidden" name="imgext" value="[mapext]">
28  <input type="hidden" name="map" value="[map]">
29  <input type="hidden" name="program" value="[program]">
30  </form>
31  </body>
32  </html>
```

La variable 'program' de la línea 29 la usa MapServer para almacenar la ruta y nombre del programa que se utilizará cuando se envíen los datos del formulario.

Para acabar la aplicación crearemos un fichero de inicio en la misma carpeta llamado 'ejemplo03_inicio.html' que contendrá lo siguiente:

```
01  <html>
```

```
02  <head> <title>MapServer</title></head>
03  <center>
04  <h2>P&aacute;gina inicial de MapServer </h2>
05  </center>
06  Ejemplo 03
07  <body>
08     <form method=POST action="/cgi-bin/mapserv.exe">
09        <input type="submit" value="Iniciar">
10        <input type="hidden" name="program" value="/cgi-bin/mapserv.exe">
11        <input type="hidden" name="zoomsize" value=2>
12        <input type="hidden" name="map"
    value="/ms4w/apps/tfc/ejemplo03.map">
13     </form>
14  </body>
15  </html>
```

Como vemos es prácticamente igual a los anteriores ficheros de inicio que hemos creado con la salvedad de la línea 06 (el título) y las líneas 10 y 11 en las que asignamos valores a las variables 'program' y 'zoomsize' para que MapServer las procese junto con la variable 'map' en el fichero plantilla. Se utilizaran como valores por defecto en este archivo plantilla.
Si en el navegador escribimos la dirección: http://gis002/tfc/ejemplo03_inicio.html, abrimos esta última página de inicio:

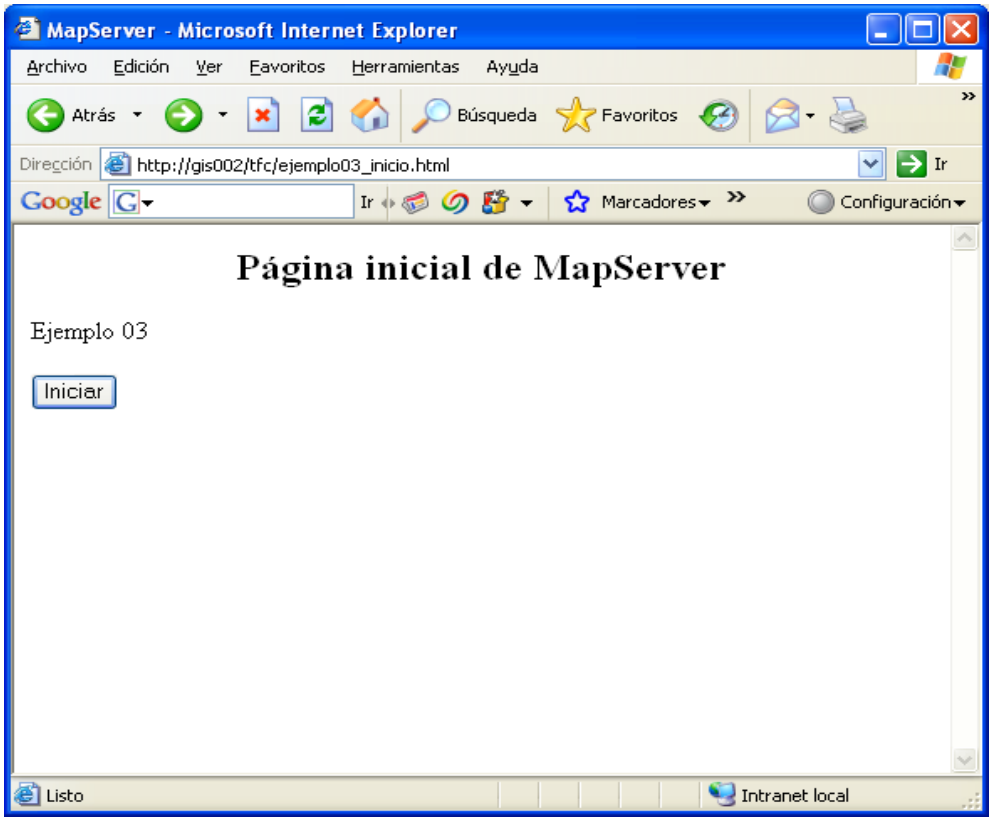


Figura 3.17 Página de inicio del ejemplo 03

Si iniciamos la aplicación, el resultado será el siguiente:

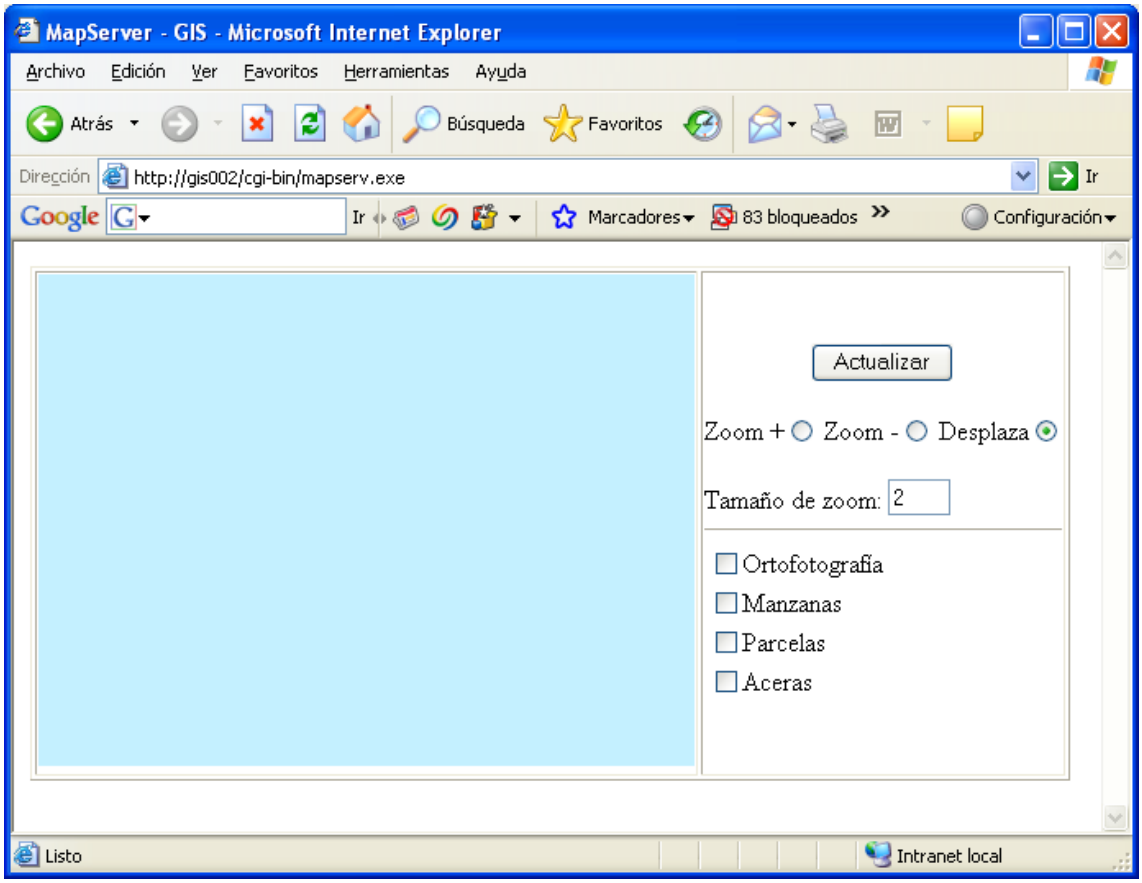


Figura 3.18 Resultado del ejemplo 03.

Aquí ya vemos todos los elementos que hemos creado: el mapa, y los controles (de navegación y de selección de capas). De momento no hay ningún elemento geográfico en el mapa puesto que no se ha activado ninguna capa.
Si activamos la casilla ‘manzanas’ de nuestra aplicación y pinchamos en ‘actualizar’ observaremos que obtenemos el mismo resultado que en el ejemplo 02, con la diferencia que en este caso podemos acercarnos a los elementos representados para verlos con mayor claridad.

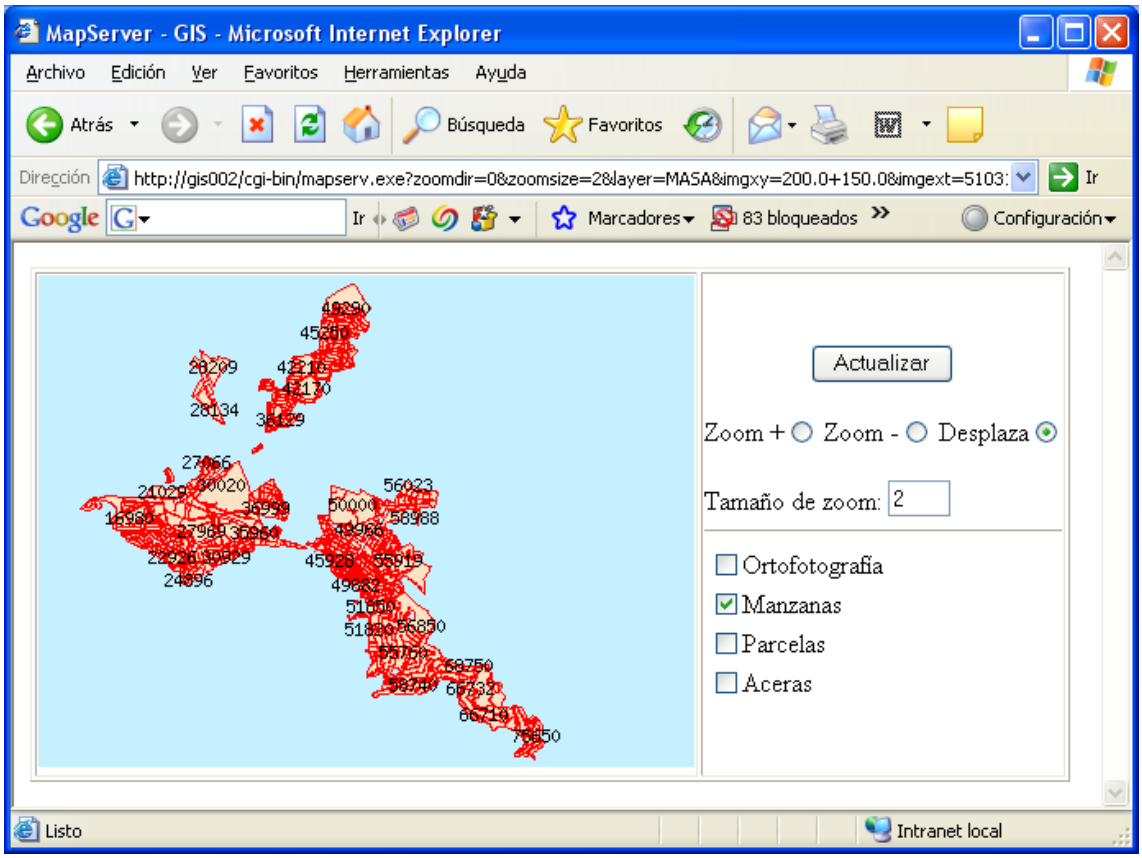


Figura 3.19 Ejemplo 03 activando las manzanas.

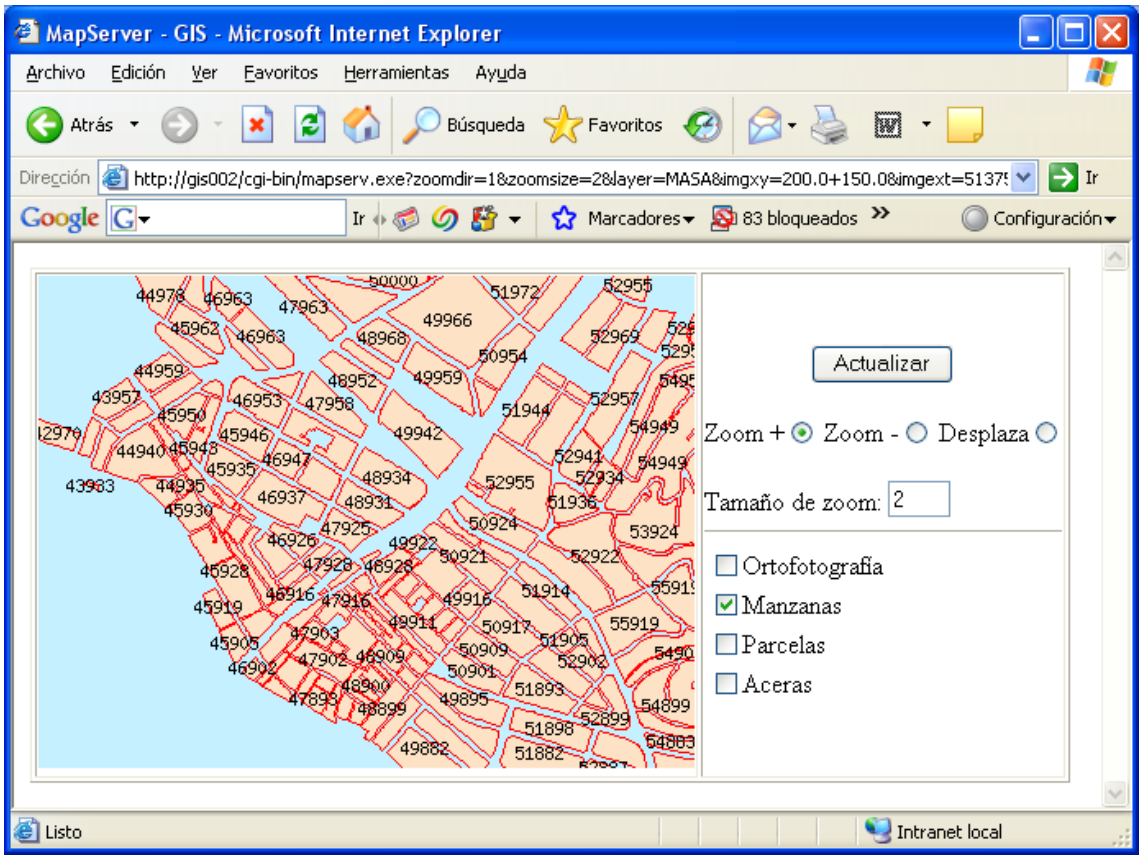


Figura 3.20 Ejemplo 03 haciendo zooms.

Cada vez que hacemos un zoom + nos aparecen más etiquetas con el nombre de las manzanas. Como ya se comentó en el ejemplo 02, esto es así porque MapServer por defecto no solapa etiquetas, pero tiene un parámetro (FORCE) que lo que hace es forzar a dibujarlas todas, y como no se ha incluido en el código toma el valor por defecto que es FALSE.
Ahora si activamos la imagen aérea veremos lo siguiente:

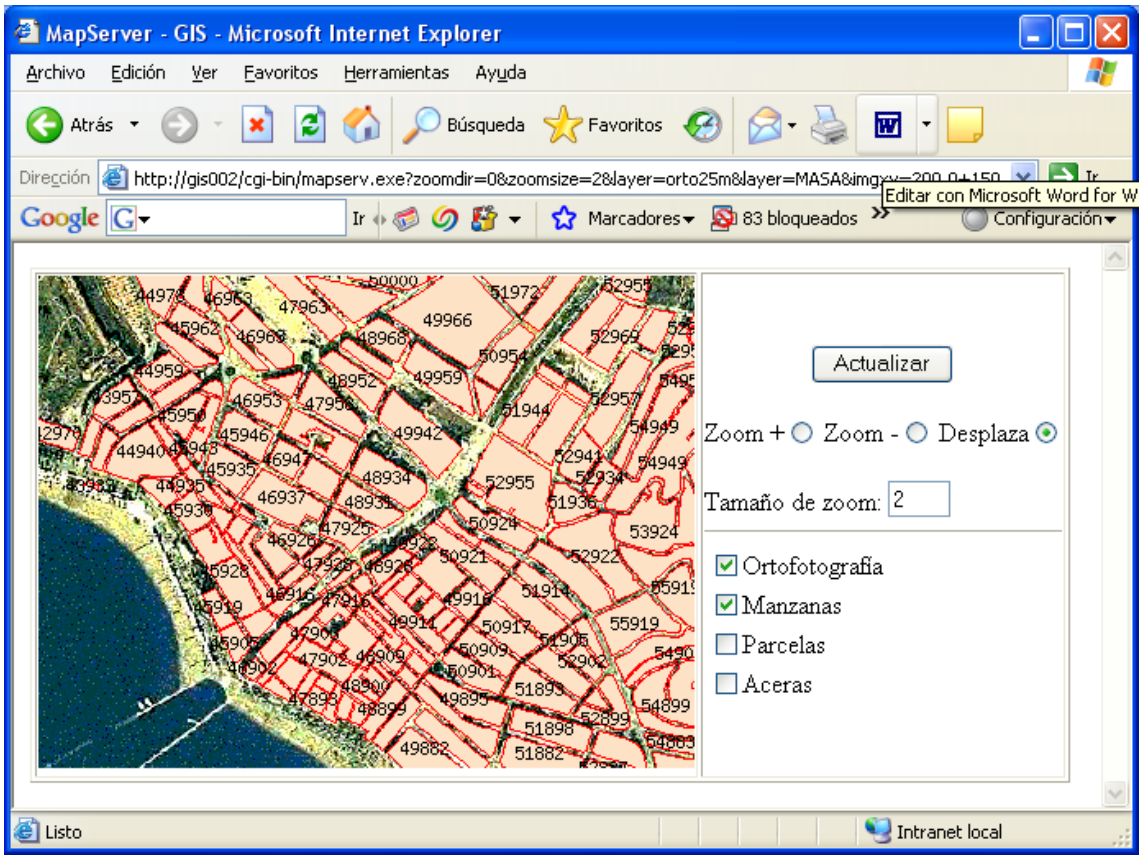


Figura 3.21 Ejemplo 03 activando manzanas y ortofotografías.

Aquí comprobamos la secuencia que sigue MapServer para dibujar las LAYERs. Al fondo la ortofotografía y por encima las manzanas.
Y en la siguiente imagen activamos las manzanas, parcelas y aceras:

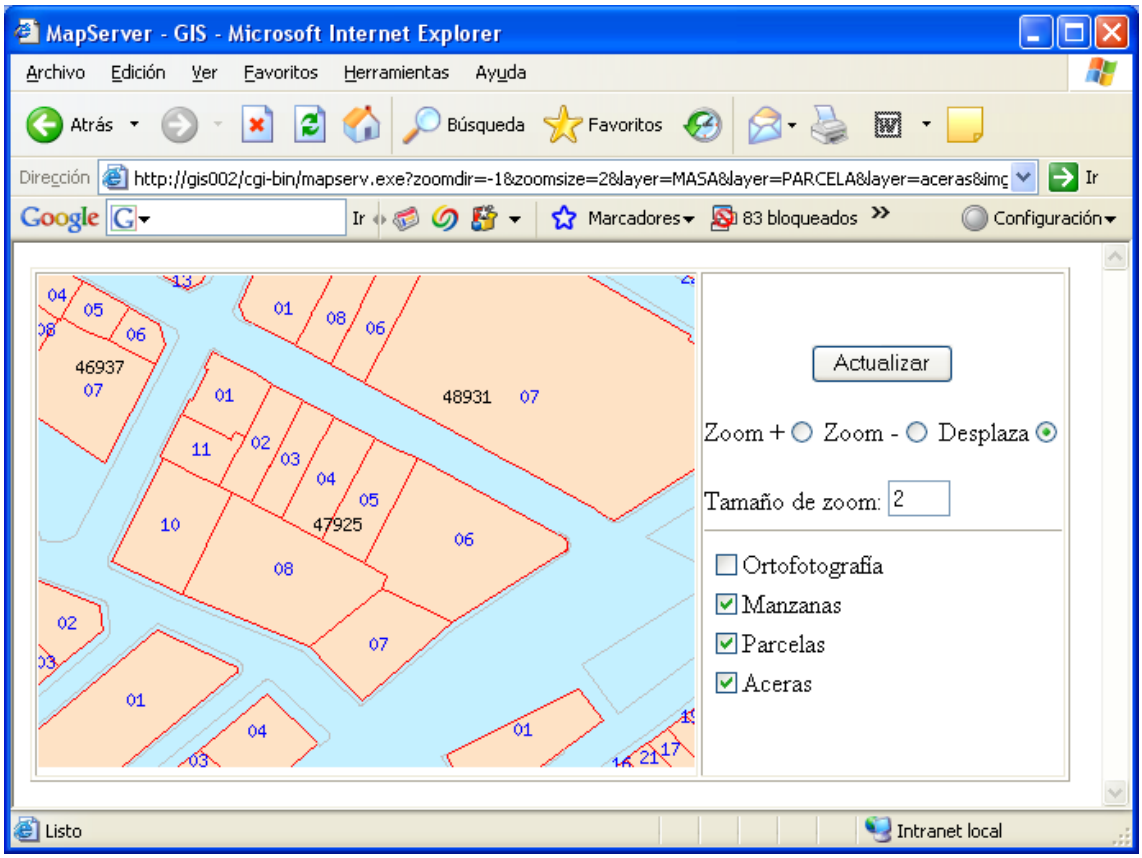


Figura 3.22 Ejemplo 03 activando manzanas, parcelas y aceras.

Si queremos que los zooms vayan más rápido, lo que tendremos que hacer será cambiar el valor de ‘Tamaño de zoom’ por uno mayor.

3.6. Funcionalidades

Hasta este momento hemos logrado un producto bastante interesante: una aplicación para visualizar información geográfica e interactuar con ella. De momento no nos hemos fijado mucho en el aspecto de esta información ni de la interfaz del usuario. Un mapa debe presentar la información eficientemente, pero también debe hacerlo de manera que el resultado final sea atractivo. En este capítulo mejoraremos el aspecto estético tanto de la aplicación como del mapa final. Exploraremos nuevos objetos y conceptos pero también mejoraremos todo lo que se ha hecho hasta ahora manipulando el color, la forma y el tamaño de cada objeto.

3.6.1. Las etiquetas (objeto LABEL)

Un objeto LABEL empieza con la palabra clave LABEL y acaba con END. Está contenido en un objeto CLASS y lo que hace es etiquetar las entidades de esta clase. También puede estar presente en otros objetos como LEGEND o SCALEBAR, pero aunque la sintaxis es muy similar, lo trataremos por separado cuando estudiemos estos otros objetos.
Una etiqueta tiene varias características como el tipo de fuente, color, orientación, posición y tamaño.

Fuentes
MapServer distingue dos tipos de fuentes mediante el parámetro TYPE: las TueType y las bitmap. Estas últimas tienen la ventaja que siempre están disponibles, no necesitan ningún recurso adicional, pero su manipulación es más limitada. Si se usa el tipo truetype, es necesario identificar el nombre de la fuente que se desea utilizar. Indicando el nombre ‘arial’ para el parámetro FONT comunicamos a MapServer que queremos utilizar la fuente asociada a la cadena ‘arial’ en el archivo externo definido en el parámetro FONTSET. Este parámetro se define en el objeto MAP.
Si tenemos el siguiente fichero ‘map’:

```
01  MAP
02    NAME "ejemplo"
03    STATUS ON
04    EXTENT 510720 4676215 519430 4683355
05    SIZE 600 600
06    SHAPEPATH "SHAPEFILES"
07    FONTSET "c:/ms4w/apps/roses/etc/fonts.txt"
08    ...
09    LAYER
10        ...
11        CLASS
12                LABEL
13                        TYPE truetype
14                FONT "arial"
15                ...
16  END
17        END
```

18 ...

En la línea 07 indicamos donde está el archivo de fuentes. Veámoslo por dentro:

fritqat	fritqat.ttf
fritqat-bold	fritqtb0.ttf
fritqat-italic	fritqti0.ttf
fritqat-bold-italic	fritqtb2.ttf
fritqat	fritqat.ttf
arial	arial.ttf
arialbi	arialbi.ttf
arialcur	arialcur.ttf

Vemos que nos relaciona el nombre de la fuente que introduciremos en el parámetro FONT del objeto LAYER con un archivo de fuente de Windows (.ttf), que se deberá encontrar en el mismo directorio que el fichero fonts.txt. En este fichero podemos definir todos los tipos de fuente que queramos, teniendo en cuenta que deberemos disponer de los correspondientes archivos .ttf.
Otra ventaja de este tipo de fuentes es que se puede especificar el tamaño con valores numéricos mediante el parámetro SIZE del objeto LAYER. La fuente bitmap solo puede adquirir uno de estos valores para el parámetro SIZE: tiny, small, medium, large o giant. Unos ejemplos de las fuentes de truetype son:

TYPE TRUETYPE FONT "arial" SIZE 15 55760	TYPE TRUETYPE FONT "fritqat-bold" SIZE 15 **55760**

Figura 3.23 Ejemplos de fuentes.

Colores
El parámetro COLOR determina el color de la etiqueta en formato RGB. Además, se puede utilizar el parámetro OUTLINECOLOR para reseguir el texto con otro color para poder leer mejor la etiqueta.

OUTLINECOLOR 255 0 0
55760

SHADOWCOLOR y SHADOWSIZE son el color y desplazamiento de la sombra del texto. Si no se especifica el color no se dibuja ninguna sombra. El desplazamiento se indica con dos valores uno para el horizontal y el otro para el vertical.

SHADOWCOLOR 255 0 0
55760

SHADOWCOLOR 255 0 0
SHADOWSIZE 2 2

Con BACKGROUNDCOLOR especificamos el color de un rectángulo también usado para resaltar la etiqueta.

BACKGROUNDCOLOR 255 255 0

BACKGROUNDSHADOWCOLOR es el color de la sombra del rectángulo definido anteriormente.

BACKGROUNDCOLOR 255 255 0
BACKGROUNDSHADOWCOLOR 0 0 0

También podemos decidir el tamaño de la sombra anterior con BACKGROUNDSHADOWSIZE utilizando dos valores enteros, uno para el desplazamiento horizontal y el otro para el vertical.

BACKGROUNDCOLOR 255 255 0
BACKGROUNDSHADOWCOLOR 0 0 0
BACKGROUNDSHADOWSIZE 3 5

Orientación

Hemos visto que todas las etiquetas hasta este momento eran completamente horizontales. De momento para las entidades que hemos etiquetado ya era correcto, pero si quisiéramos etiquetar elementos lineales (como calles, ríos, infraestructuras de servicios lineales, etc.) es preferible hacer que estas etiquetas estén a lo largo de estos elementos que describen. Para esto es necesario utilizar fuentes truetype ya que es imposible cambiar el ángulo de las fuentes bitmap. El parámetro ANGLE se puede usar para especificar el ángulo (grados en valor numérico) con el que serán dibujadas las etiquetas. Si es positivo rota las etiquetas el valor introducido en la dirección antihoraria mientras que si es negativo lo hace al revés.

Figura 3.24 Ejemplos de orientaciones del texto

Para los elementos lineales también se puede utilizar el valor AUTO para el parámetro ANGLE, cosa que hará que MapServer calcule el ángulo correcto para que la etiqueta sea dibujada a lo largo de cada línea.

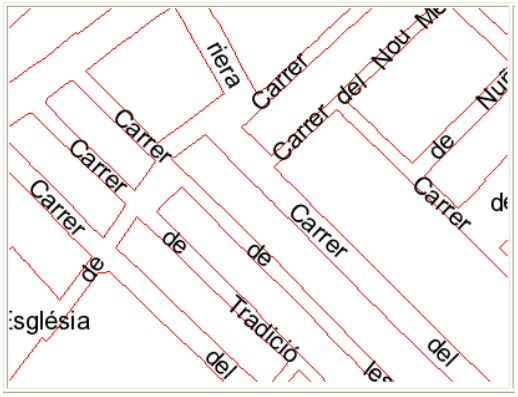


Figura 3.25 Resultado al utilizar el valor AUTO.

Posición

El emplazamiento de las etiquetas respecto a la entidad se define mediante el parámetro POSITION. Existen diez valores posibles:

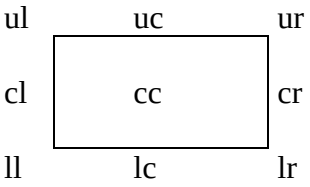


Figura 3.25 Esquema de la situación de los valores del parámetro POSITION

Estos valores indican las iniciales de la posición en inglés:

l: left	u: upper
c: center	c: center
r: right	l: lower

El primero hace referencia a la posición vertical y el segundo a la horizontal.
El otro valor es AUTO. Con éste MapServer selecciona automáticamente la posición para que la etiqueta no interfiera con otras etiquetas. Esto puede provocar algún problema estético ya que dibujará tantas etiquetas como sitio encuentre para ellas. Esto puede provocar que para una sola entidad que sea larga (una calle o carretera) esté etiquetada muchas veces. Si estas etiquetas están suficientemente distanciadas no hay problema, pero si están muy juntas no es recomendable que estén todas, habría que limitarlas. Para solucionar este problema podemos usar el parámetro MINDISTANCE en que especificamos el número de píxeles entre dos etiquetas duplicadas de la misma entidad.
Por otro lado, si tenemos una entidad muy pequeña puede que la etiqueta nos la tape completamente. En este caso no sería necesario etiquetarla. El parámetro MINFEATURESIZE establece el tamaño en píxeles de la entidad más pequeña a etiquetar.

Otros parámetros
LABELCACHE aunque no es un parámetro del objeto LABEL (es del LAYER) afecta a la representación de las etiquetas. Por defecto está activado (on) y hace que MapServer dibuje todas las etiquetas de las capas que tienen este parámetro activado a la vez y después de dibujar el resto de entidades. Esto lo hace para que cuando se utilizan los parámetros que hemos visto anteriormente como POSITION o MINFEATURESIZE con el valor de AUTO, MapServer sea capaz de organizar automáticamente las etiquetas para que no se solapen.
Si se desactiva (off) dibuja las etiquetas al mismo tiempo que las entidades de la capa.

BUFFER especifica en píxeles el espacio libre alrededor de la etiqueta.
FORCE cuando se activa (true) fuerza a dibujar todas las etiquetas aunque se solapen. Por defecto está desactivado (false).

3.6.2. Visualizar según la escala

Con los mapas digitales se tiene la posibilidad de dibujar o no entidades según la escala de visualización. El parámetro MAXSCALE del objeto LAYER establece la escala máxima a la que las entidades de esta capa serán dibujadas. Análogamente, el parámetro LABELMAXSCALE establece la escala máxima a la que serán dibujadas las etiquetas. Los parámetros MINSCALE y LABELMINSCALE establecen la escala mínima de visualización de entidades y etiquetas.
Para definir estos rangos se puede utilizar el denominador de la escala que se guarda en la variable [scale], haciendo que se visualice en el archivo plantilla.

Antes de continuar, vamos mejorar la aplicación del ejemplo 03 creando otra e introduciendo algunos de los parámetros que hemos visto hasta ahora.
Empecemos por el archivo ‘mapa’:

```
01  NAME "ejemplo04"
02  SIZE 400 300
03  IMAGETYPE png
04  EXTENT 510720 4676215 519430 4683355
05  SHAPEPATH "datos"
06  FONTSET "c:/ms4w/apps/roses/etc/fonts.txt"
07  WEB
08      TEMPLATE "/ms4w/apps/tfc/ejemplo04.html"
09      IMAGEPATH "/ms4w/tmp/ms_tmp/"
10      IMAGEURL "/ms_tmp/"
11  END
```

La única novedad es la especificación en la línea 06 del archivo donde están definidas las fuentes truetype que vamos a utilizar más adelante. El resto es igual que en el ejemplo 03.

```
12  LAYER
13          NAME "orto25m"
```

```
14        DATA "orto25m.tif"
15        TYPE RASTER
16        STATUS ON
17        MINSCALE 5000
18  END
```

Esta LAYER también es igual a al del anterior ejemplo, con la salvedad de la línea 17 en que indicamos la escala mínima de visualización de la ortofotografía a escala 1:25000. La imagen a una escala menor no se aprecia correctamente, por ejemplo a una escala 1:4200 se ve así:

Figura 3.26 Visualización de la ortofotografía a una escala menor.

Cosa poco agradable visualmente. Estableciendo este parámetro, cuando hagamos un zoom + y sobrepasemos el límite impuesto (1:4000, 1:3000, etc.) no se visualizará la ortofotografía. Como se verá en ejemplos más sofisticados, se puede crear otra capa con ortofotografías más precisas (1:5000) que se active cuando se desactiva ésta.
Seguimos con la capa de manzanas catastrales:

```
19   LAYER
20     NAME "MASA"
21     STATUS ON
22     DATA "MASA"
23     TYPE POLYGON
24     LABELITEM "MASA"
25     LABELMAXSCALE 4000
26     CLASS
27       STYLE
28         COLOR 254 226 197
29         OUTLINECOLOR 255 0 0
30       END
31       LABEL
```

```
32        TYPE TRUETYPE
33        FONT "arial"
34        SIZE 14
35        COLOR 99 0 255
36        FORCE TRUE
37        POSITION AUTO
38      END
39    END
40  END
```

La primera línea que se ha añadido es la 25. Definimos como escala máxima de visualización de las etiquetas a 1:4000. Es decir, que únicamente serán visibles cuando el denominador de la escala sea menor que 4000. Nótese que no se ha establecido un valor de escala máxima para las entidades (las manzanas), sólo para sus etiquetas.

Figura 3.27 A un denominador de escala mayor, no se visualizan las etiquetas.

En la imagen se ve que la escala de visualización es de aproximadamente 1:4216. Por lo tanto no se dibujan las etiquetas de las manzanas pero sí las manzanas. Vamos a acercarnos un poco:

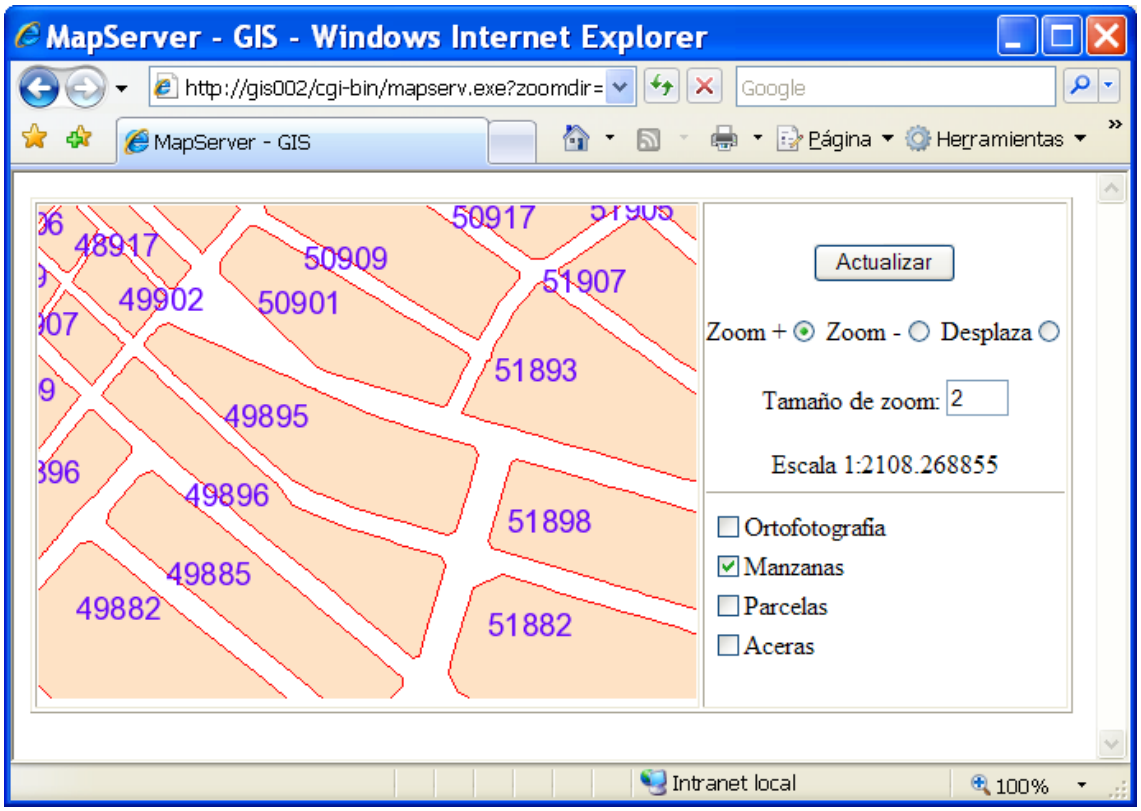


Figura 3.28 Si disminuimos el denominador (haciendo un zoom) ya se ven las etiquetas.

Ahora sí que las vemos. El estilo de la etiqueta viene definido por el objeto LABEL. Nos ha dibujado las etiquetas de todas las manzanas puesto que teníamos activado el parámetro FORCE. Otro aspecto curioso es que algunas etiquetas no están completas. Esto se puede controlar con otro parámetro: PARTIALS. Por defecto está activado (true), y recordemos que si no se define coge este valor por defecto.
Continuemos con las parcelas:

```
41  LAYER
42    NAME "PARCELA"
43    STATUS ON
44    DATA "PARCELA"
45    TYPE POLYGON
46    LABELITEM "PARCELA"
47    LABELMAXSCALE 2000
48    MAXSCALE 5000
49    CLASS
50      STYLE
51        OUTLINECOLOR 255 0 0
52      END
53      LABEL
54        TYPE TRUETYPE
55        FONT "arialBI"
56        SIZE 10
57        POSITION CC
58        COLOR 0 0 0
59      END
```

```
60      END
61    END
```

La línea 47, al igual que la 25 establece la escala máxima en que se visualizarán las etiquetas de las parcelas. En este caso queda definida a 2000. Además, en la siguiente línea también establecemos en 5000 la escala máxima para visualizar las propias parcelas.

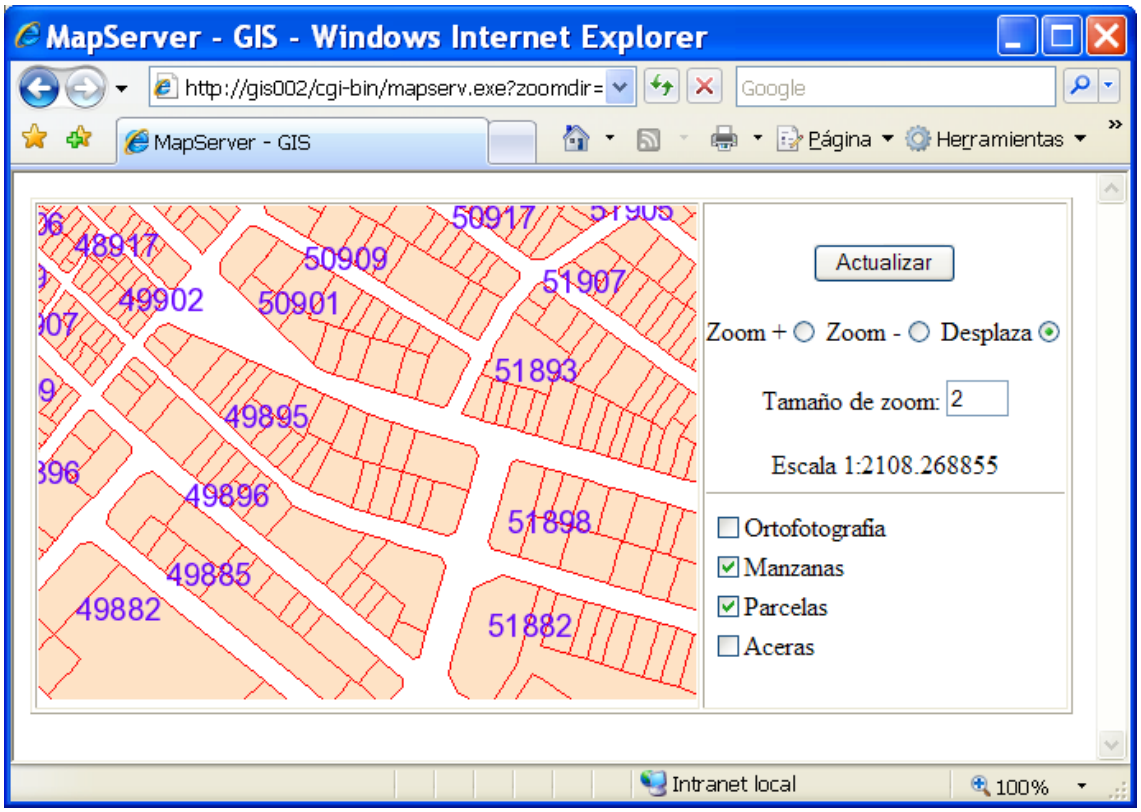


Figura 3.29 A esta escala se ven las parcelas pero no sus etiquetas

Como la escala es aproximadamente 1:2108 se ven las manzanas y sus etiquetas pero solo las parcelas sin etiquetar. Hará falta acercarnos un poco para verlas:

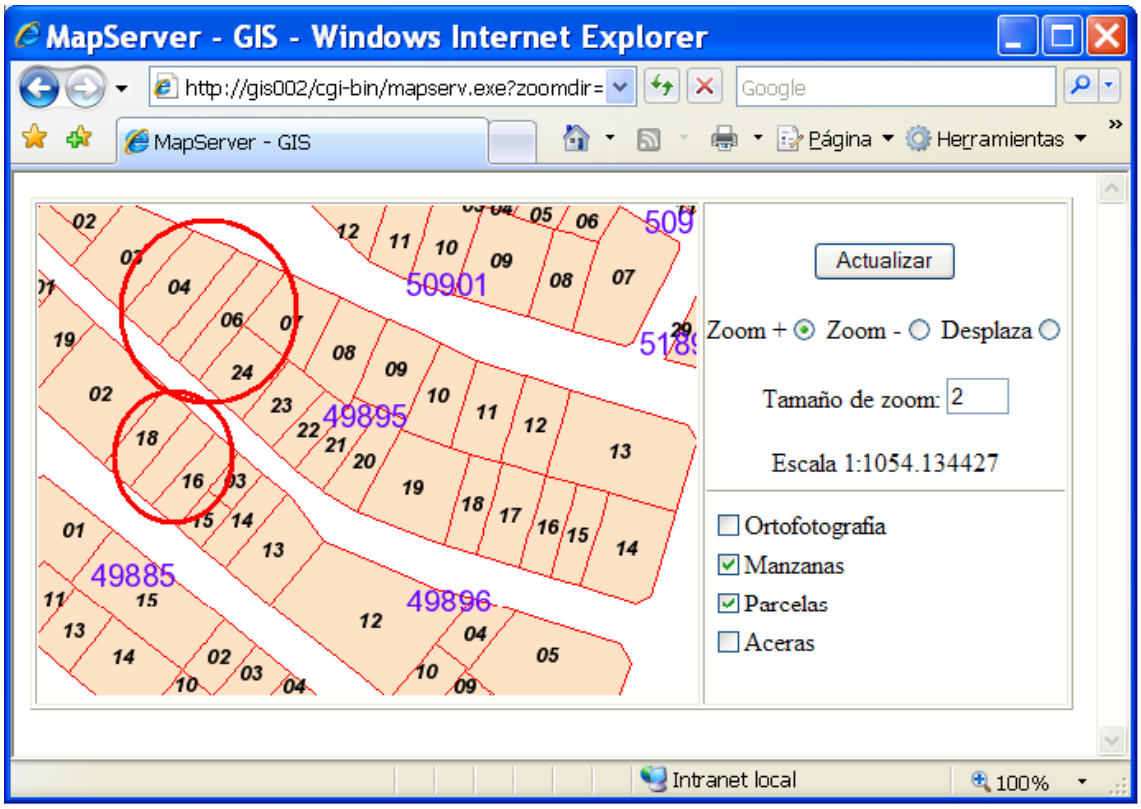


Figura 3.30 A esta escala se ven ya las etiquetas de las parcelas que no se solapan.

¡Ahora sí!. Para estas otras etiquetas se ha elegido un color, tamaño y fuente diferentes para distinguirlas mejor. Como no se ha especificado el parámetro FORCE, coge el valor por defecto que es desactivado (false), así no dibuja las etiquetas que no puede colocar sin que se superpongan. Son las que deberían estar en los círculos marcados en la imagen.
Veamos como insertar los nombres de las calles. Tenemos un archivo shp con los ejes de las calles y cuyo atributo 'ROTULO' contiene el texto con el nombre. Estos ficheros son: carrer.shp, carrer.dbf y carrer.shx que deberemos colocar en nuestra carpeta 'datos'.

```
62  LAYER
63    NAME "carrer"
64    STATUS default
65    DATA "carrer"
66    TYPE line
67    LABELITEM "rotulo"
68    LABELMAXSCALE 5000
69    CLASS
70      LABEL
71        TYPE TRUETYPE
72        FONT "arial"
73        SIZE 8
74        ANGLE auto
75        COLOR 0 0 0
76       END
77     END
```

```
78   END
```

Puesto que nos interesan únicamente los nombres de las calles, sin los ejes, en el objeto CLASS definimos solo un objeto LABEL y a éste le añadimos el parámetro ANGLE auto para que los nombres tengan la orientación de cada eje. El resultado es el siguiente:

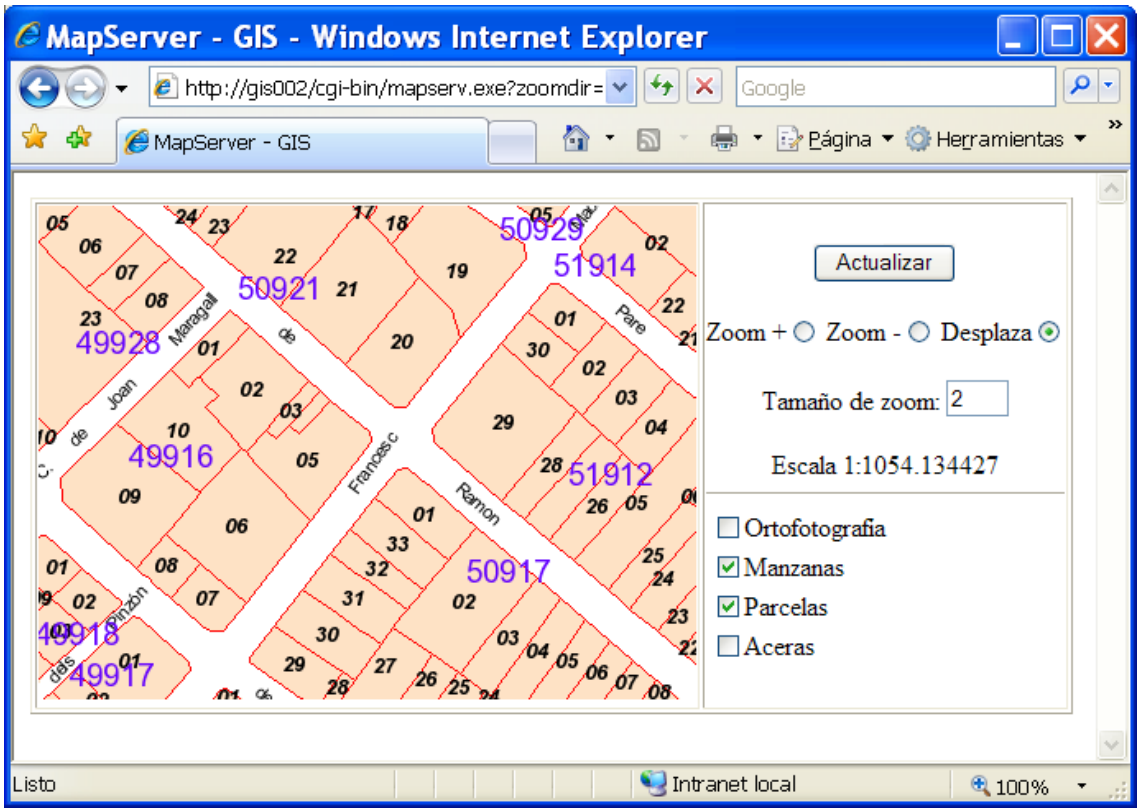


Figura 3.31 Añadimos las etiquetas de los ejes de las calles

Un detalle a tener en cuenta es que todas las etiquetas están dibujadas sobre las entidades, sin importar el orden del fichero 'mapa'. Si MapServer dibuja primero las manzanas y luego las parcelas, estas últimas deberían tapar a las etiquetas de las manzanas. Pero como hemos dicho anteriormente, por defecto MapServer dibuja todas las entidades y luego todas las etiquetas (parámetro LABELCAHE).
Para acabar, nos queda insertar las aceras que no sufrirán más cambios que el control de visualización según la escala respecto al ejemplo 03, y cerrar el objeto MAP:

```
79   LAYER
80     NAME "aceras"
81     STATUS ON
82     DATA "elemlin"
83     TYPE LINE
84     MAXSCALE 15000
85     CLASS
86       STYLE
87         COLOR 192 192 192
88       END
89     END
90   END
```

```
91  END
```

El código del archivo plantilla tampoco sufre grandes cambios:

```
01  <html>
02  <head>
03    <title>MapServer - GIS</title>
04  </head>
05  <body>
06  <form method="GET" action="[program]" name="mapserv">
07  <table border="1">
08        <td align="center">
09        <input type="image" name="img" src="[img]" width="[mapwidth]" height="[mapheight]" border="0">
10        </td>
11        <td valign="center" >
12        <center><input type="submit" value="Actualizar"></center>
13     <p align="center">Zoom +<input type=radio name=zoomdir  value=1 checked [zoomdir_1_check]>
14                              Zoom -<input type=radio name=zoomdir value=-1 [zoomdir_-1_check]>
15                              Desplaza<input type=radio name=zoomdir value=0 [zoomdir_0_check]>
16     <p align="center">Tamaño de zoom: <input type=text name=zoomsize size=2 value=[zoomsize]>
17     <p align="center">Escala 1:[scale]
18     <hr size="1">
19          <table>
20            <tr><td colspan="3"><input type="checkbox" name="layer" value="orto25m" [orto25m_check]>Ortofotografía</td></tr>
21            <tr><td colspan="3"><input type="checkbox" name="layer" value="MASA" [MASA_check]>Manzanas</td></tr>
22            <tr><td colspan="3"><input type="checkbox" name="layer" value="PARCELA" [PARCELA_check]>Parcelas</td></tr>
23            <tr><td colspan="3"><input type="checkbox" name="layer" value="aceras" [aceras_check]>Aceras</td></tr>
24          </table>
25        </td>
26  </table>
27  <input type="hidden" name="imgxy" value="[center]">
28  <input type="hidden" name="imgext" value="[mapext]">
29  <input type="hidden" name="map" value="[map]">
30  <input type="hidden" name="program" value="[program]">
31  </form>
32  </body>
33  </html>
```

Se le añade la línea 17 que inserta el valor de la escala del mapa que hemos usado para definir las escalas de visualización de los elementos.
El archivo de inicio también es prácticamente igual:

```
01 <html>
02 <head> <title>MapServer</title></head>
03 <center>
04 <h2>P&aacute;gina inicial de MapServer </h2>
05 </center>
06 Ejemplo 04
07 <body>
08    <form method=POST action="/cgi-bin/mapserv.exe">
09       <input type="submit" value="Iniciar">
10             <input type="hidden" name="program" value="/cgi-
   bin/mapserv.exe">
11             <input type="hidden" name="zoomsize" value=2>
12             <input type="hidden" name="map"
   value="/ms4w/apps/tfc/ejemplo04.map">
13              <input type="hidden" name="layer" value="orto25m">
14    </form>
15 </body>
16 </html>
```

También se ha añadido una línea, la 13, que utilizamos para enviar una nueva variable: la 'layer' con el valor 'orto25m'. Esto provocará que se inicie la aplicación con esta capa activada. (Recordemos que en anterior ejemplo al iniciar la aplicación nos mostraba una imagen en blanco).
Podemos hacer lo mismo con tantas capas como queramos, añadiendo más líneas como la 13 con valores de nombres de objetos LAYER de nuestro fichero 'map'.

3.6.3. Clasificación de entidades

Para MapServer un objeto LAYER consiste en un grupo de entidades, que forman parte de un mismo grupo de datos geográficos, que se dibujarán al mismo tiempo y a la misma escala. Sin embargo, algunas veces puede no interesarnos dibujar todas las entidades del grupo de datos. Por ejemplo, podría interesarnos visualizar únicamente las manzanas cuya referencia empiece por 5 o visualizar todas y las que empiecen por 2 de otro color.
El objeto CLASS nos permite diferenciar entre entidades utilizando sus atributos. En los anteriores ejemplos ya hemos usado este objeto (todos los objetos LAYER deben tener al menos uno), pero como no hemos indicado lo contrario (no hemos hecho ninguna selección) siempre nos ha dibujado todas las entidades de ese objeto. Siempre nos ha dibujado todas las manzanas, parcelas y aceras.
En el siguiente ejemplo vamos a introducir más datos: las subparcelas catastrales. Los datos están en los ficheros constru.shp, constru.dbf y constru.shx. Las subparcelas son todas las divisiones interiores de las parcelas que tienen una construcción diferente a las que las rodean. El objetivo de este ejemplo será visualizar estas subparcelas con colores diferentes según el tipo de construcción que sea (almacenado en el campo 'constru' del fichero .dbf).
El parámetro CLASSITEM del objeto LAYER identifica el nombre del campo que contiene los atributos a clasificar. Usando el parámetro EXPRESSION del objeto CLASS especificamos la comparación.

Si por ejemplo queremos seleccionar las subparcelas que en el campo ‘construu’ tienen almacenado el valor ‘P’ (patio) el fichero ‘map’ tendrá este aspecto:

```
LAYER
...
CLASSITEM “construu”
CLASS
        NAME “patios”
        EXPRESSION “P”
 ...
```

Existen tres tipos de expresiones:

Comparaciones de valores. Un atributo se compara con una cadena de caracteres (el ejemplo anterior es de este tipo).
Expresiones regulares. Se forman siguiendo el mismo patrón que las comparaciones en sistemas UNIX. Funcionan de forma similar a las comparaciones de valores anteriores, pero permiten operaciones más complejas. Se usan de manera análoga que las anteriores, usando los parámetros CLASSITEM y EXPRESSION pero para diferenciarlas deben estar entre dos caracteres ‘/’.
Se utilizan caracteres especiales como:

- . (punto): es el comodín para un carácter.
- [ ]: Se usa para agrupar. Por ejemplo [A-D] corresponde a los caracteres A, B, C y D.
- ^: inicio de la cadena
- $: final de la cadena

Las líneas:

```
CLASSITEM “año”
EXPRESSION /^20[0-9][0-9]/
```

Seleccionará las entidades cuyo campo ‘año’ esté entre 2000 y 2099 ambos incluidos.

Expresiones lógicas. Permiten hacer clasificaciones aún más complejas teniendo en cuenta uno o más atributos. A diferencia de las otros dos tipos, no necesitaremos el parámetro CLASSITEM. La estructura simple consiste en un nombre de atributo entre corchetes ([ ]), un operador de comparación y un valor, todo entre paréntesis.

Con EXPRESSION ([area]>1000) se seleccionan las entidades cuyo valor del campo ‘area’ sea superior a 1000.
Con EXPRESSION ("[HOJA]" eq "EG1749S") se seleccionan las entidades cuyo valor del campo ‘hoja’ es igual a la cadena ‘EG1749S’.
Se pueden encadenar varias expresiones para acotar más la selección. Por ejemplo, si queremos seleccionar las entidades que cumplen las dos condiciones anteriores escribiremos:

```
EXPRESSION (("[HOJA]" EQ "EG1749S") and ([AREA]>1000))
```

Operadores de comparación:

Operador	Tipo	Operador	Tipo
=	numérico	eq	alfanumérico
!=	numérico	ne	alfanumérico
>	numérico	gt	alfanumérico
<	numérico	lt	alfanumérico
>=	numérico	ge	alfanumérico
<=	numérico	le	alfanumérico
and	lógico		
or	lógico		

Y su significado bastante obvio. De arriba hacia abajo: igual, diferente, mayor, menor, mayor o igual y menor o igual tanto para los numéricos como para los alfanuméricos.

Existen más operadores y tipos de expresiones no tan comunes, pero quizá útiles para algún trabajo más complejo. En la web de MapServer hay un documento mucho más amplio sobre estos temas (http://mapserver.gis.umn.edu/docs/howto/msexpressions).

Para el objetivo de este capítulo usaremos el primer tipo de expresiones. Recordemos que queremos visualizar en diferente color cada tipo de subparcelas.
Empezaremos añadiendo a nuestro fichero 'mapa' un objeto LAYER más y dentro de este crearemos tantos objetos CLASS como subparcelas queramos distinguir.
Si abrimos el fichero constru.dbf y observamos el campo 'constru' veremos que los datos son del estilo: P, -I+II, PI, SOLAR, etc. Estos son los códigos de construcciones que utiliza el catastro. Para mejorar nuestra aplicación usaremos los siguientes colores:

Código	Descripción	Color (RGB)
P	Patio	255 210 166
PI	Piscina	43 149 255
ETQ	Amarre	187 221 255
SOLAR	Solar (sin construcción)	180 164 158
JD	Jardín	155 214 116
El resto	Edificaciones	255 136 136

Puesto que existen una inmensidad de combinaciones de tipos de edificaciones no es una buena opción plantearnos hacer un objeto CLASS para cada uno. Podríamos hacer una expresión lógica del tipo

EXPRESSION (("[CONSTRU]" ne "P") and ("[CONSTRU]" ne "PI") and …

Pero MapServer tarda mucho tiempo en analizar este tipo de expresiones porque cada vez que evalúa una parte vuelve a recorrer todos los elementos para evaluar la segunda parte, y así sucesivamente.
Una buena opción es crear un objeto CLASS que nos dibuje todas los elementos como si fueran edificaciones y encima de éstos dibujamos el resto. Tendremos que tener en cuenta, pues, el orden en que creamos estos objetos CLASS. A diferencia de los objetos LABEL en que el primero en aparecer en el fichero 'map' era el primero en ser dibujado, con los objetos CLASS pasa al revés: empieza por el último.

Deberemos cambiar el orden de las LAYERS que tenemos y alguna característica más, ya que nuestra intención es dibujar las subparcelas coloreadas y taparían a las entidades que se han dibujado antes.
El archivo ‘mapa’ será:

```
01  NAME "ejemplo05"
02  SIZE 400 300
03  IMAGETYPE png
04  EXTENT 510720 4676215 519430 4683355
05  SHAPEPATH "datos"
06  FONTSET "c:/ms4w/apps/tfc/etc/fonts.txt"
07  WEB
08      TEMPLATE
    "/ms4w/apps/tfc/ejemplo05.html"
09      IMAGEPATH "/ms4w/tmp/ms_tmp/"
10      IMAGEURL "/ms_tmp/"
11  END
12  LAYER
13    NAME "orto25m"
14    DATA "orto25m.tif"
15    TYPE RASTER
16    STATUS ON
17    MINSCALE 2000
18  END
19    LAYER
20      NAME "SUBPARCELA"
21      STATUS on
22      DATA "CONSTRU"
23      TYPE POLYGON
24      MAXSCALE 5000
25      CLASSITEM "CONSTRU"
26      CLASS
27       EXPRESSION "P"
28       STYLE
29            COLOR 255 210 166
30       END
31      END
32      CLASS
33       EXPRESSION "PI"
34       STYLE
35            COLOR 43 149 255
36       END
37      END
38      CLASS
39       EXPRESSION "ETQ"
40       STYLE
41            COLOR 187 221 255
42       END
43      END
44      CLASS
45       EXPRESSION "SOLAR"
46       STYLE
47            COLOR 180 164 158
48       END
49      END
50      CLASS
51       EXPRESSION "JD"
52       STYLE
53            COLOR 155 214 116
54       END
55      END
56      CLASS
57       STYLE
58            COLOR 255 136 136
59       END
60      END
61    END
62    LAYER
63      NAME "MASA"
64      STATUS ON
65      DATA "MASA"
66      TYPE POLYGON
67      LABELITEM "MASA"
68      LABELMAXSCALE 4000
69      CLASS
70       STYLE
71        OUTLINECOLOR 255 0 0
72       END
73       LABEL
74        TYPE TRUETYPE
75        FONT "arial"
76        SIZE 14
77        COLOR 99 0 255
78        FORCE TRUE
79        POSITION AUTO
80       END
81      END
82    END
83    LAYER
84      NAME "PARCELA"
85      STATUS ON
86      DATA "PARCELA"
87      TYPE POLYGON
88      LABELITEM "PARCELA"
89      LABELMAXSCALE 2000
90      MAXSCALE 5000
91      CLASSITEM "AREA"
92      CLASS
93       STYLE
94        OUTLINECOLOR 0 0 0
95       END
96       LABEL
97        TYPE TRUETYPE
98        FONT "arialBI"
99        SIZE 10
100       POSITION CC
101       COLOR 0 0 0
102      END
103     END
104   END
105   LAYER
106     NAME "carrer"
107     STATUS default
108     DATA "carrer"
109     TYPE line
110     LABELITEM "rotulo"
111     LABELMAXSCALE 5000
112     CLASS
113      LABEL
114       TYPE TRUETYPE
115       FONT "arial"
116       SIZE 8
117       angle auto
118       COLOR 0 0 0
119      END
120     END
121   END
122   LAYER
123     NAME "aceras"
124     STATUS ON
125     DATA "elemlin"
126     TYPE LINE
127     MAXSCALE 15000
128     CLASS
129      STYLE
130       COLOR 192 192 192
131      END
132     END
133   END
134 END
```

También se ha añadido la opción de activar y desactivar la capa de subparcelas en la plantilla, que quedará como sigue:

```
23    <tr><td colspan="3"><input type="checkbox" name="layer"
      value="SUBPARCELA" [SUBPARCELA_check]>Subparcelas</td></tr>
```

En el archivo de inicio únicamente cambiaremos el nombre del fichero 'mapa' que debe ejecutar.
El aspecto de estos cambios es:

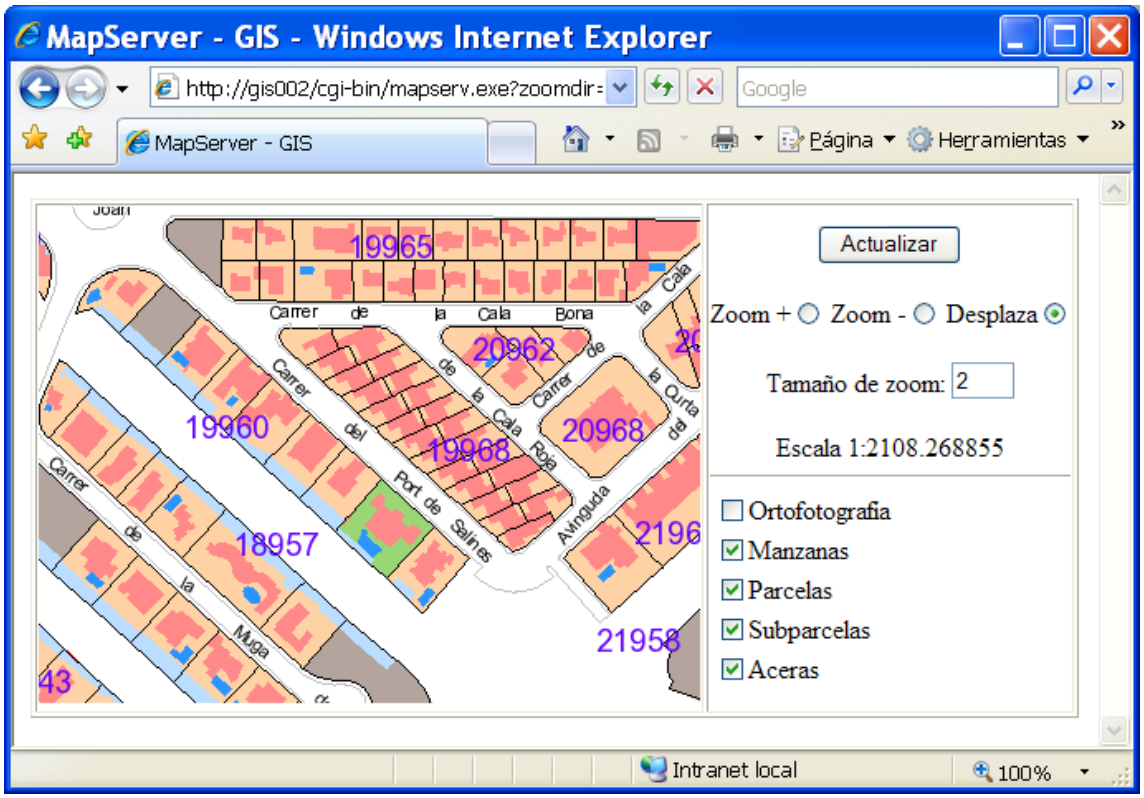


Figura 3.31 Aspecto del mapa coloreando los elementos.

3.6.4. Símbolos

En los capítulos anteriores hemos estado usando únicamente parámetros para determinar los colores de visualización de las entidades. No había ninguno para cambiar el tamaño de las líneas, que siempre tenían 1 píxel de ancho. Usando símbolos podemos establecer el tamaño para cada elemento del objeto CLASS.
El objeto SYMBOL se define dentro del objeto MAP, y por lo tanto estará accesible para todos los objetos LAYER del mapa. También se pueden definir en un fichero externo, con la ventaja que estará disponible para otras aplicaciones y nuestro fichero 'mapa' será menos denso.
Como todos los objetos de MapServer empieza por la palabra SYMBOL y acaba por END.

```
SYMBOL
        NAME “linea”
        TYPE ELLIPSE
        POINTS 1 1 END
END
```

Este es un ejemplo de la definición de un símbolo muy simple. Primero se define un nombre que será el que utilizaremos para hacer referencia en los objetos LAYER. Luego hay que definir que tipo de símbolo es y por último los parámetros para dibujarlo.
Hay varios tipos diferentes de símbolos (TYPE): vector, ellipse, pixmap, cartoline y truetype.
El tipo VECTOR define la forma del símbolo estableciendo una serie de puntos mediante pares de coordenadas usando un sistema de referencia local con origen en la esquina inferior izquierda. Esta serie de puntos se engloban entre las palabras POINTS y END. Por defecto el máximo número de puntos es de 100, aunque se puede cambiar.
El tipo ELLIPSE sigue la misma sintaxis que el VECTOR, pero se interpreta de forma diferente. El parámetro POINTS contiene únicamente un par de coordenadas, que establecen la longitud de los ejes de la elipse.
El tipo PIXMAP son pequeñas imágenes ráster (GIF o PNG). Dentro del objeto SYMBOL en el parámetro IMAGE se especifica el nombre de la imagen.
El tipo CARTOLINE, aunque no es en realidad un símbolo independiente es útil para resolver los problemas de visualización de líneas cuando forman esquina (debido a su escasa aplicación no lo trataremos en este texto).
Finalmente, el tipo TRUETYPE usa caracteres truetype como símbolos.
A continuación vamos a ver unos ejemplos de creación de símbolos más comunes:

Símbolos puntuales
Con las siguientes líneas crearemos un símbolo cuadrado. El parámetro FILLED indica que el cuadrado será rellenado.

```
SYMBOL
  NAME "cuadrado"
  TYPE VECTOR
  POINTS
    0 0
    0 1
    1 1
    1 0
    0 0
  END
  FILLED TRUE
END
```

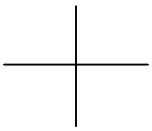
Si queremos dibujar un círculo:

```
SYMBOL
  NAME "circulo"
  TYPE ELLIPSE
  POINTS
    1 1
  END
  FILLED TRUE
END
```

También es posible dibujar líneas sueltas, como cruces o aspas:

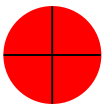
```
SYMBOL
  NAME "cruz"
  TYPE VECTOR
```

```
  POINTS
    0.5 0
    0.5 1
    -99 -99
    0 0.5
    1 0.5
  END
END
```



En el símbolo ‘cruz’ vemos un punto un tanto raro (-99,-99) MapServer lo interpreta como un salto, es decir, que no una el punto anterior con el posterior.
Los símbolos se pueden utilizar conjuntamente para crear otros más complejos; en el objeto CLASS de una entidad se puede poner:

```
LAYER
        NAME pr1
        TYPE POINT
        ...
        CLASS
        STYLE
                SYMBOL “circulo”
                SIZE 40
                COLOR 255 0 0
        END
        STYLE
                SYMBOL “cruz”
                SIZE 40
                COLOR 0 0 0
        END
…
```

Lo que nos producirá que los puntos se visualicen así: 

Recordemos que estas definiciones es aconsejable tenerlas en un fichero externo. En este trabajo las guardaremos en el mismo directorio de otro fichero externo que hemos utilizado: el de fuentes (C:\ms4w\apps\tfc\etc), y lo llamaremos symbols.sym.

Símbolos lineales
Las líneas se dibujan usando símbolos puntuales. Este símbolo se dibuja para cada píxel a lo largo de la línea obteniendo una forma lineal continua.
Se pueden crear patrones de línea usando el parámetro STYLE:

```
SYMBOL
  NAME "linea_discontinua"
  TYPE ELLIPSE
  POINTS
    1 1
```

```
  END
  STYLE
    10 10 10 10
  END
END
```

Este parámetro lo que hace es dibujar una línea discontinua formada por círculos: 10 píxeles sí y 10 no, 10 sí y 10 no.
Como en el ejemplo anterior, se pueden combinar diferentes símbolos para crear otros más complejos:

```
LAYER
        NAME pr2
        TYPE LINE
        ...
        CLASS
        STYLE
                SYMBOL “circulo”
                SIZE 7
                COLOR 255 255 0
        END
        STYLE
                SYMBOL “linea_discontinua”
                SIZE 1
                COLOR 0 0 0
        END
…
```

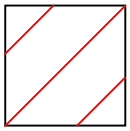
Cosa que producirá la siguiente linea:

Símbolos de área.
Las entidades de tipo POLYGON (de área) pueden rellenarse con símbolos. Por ejemplo, si tenemos un símbolo que es una línea diagonal:

```
SYMBOL
  NAME "diagonal "
  TYPE vector
  POINTS
    0 0
    1 1
  END
END
```

Podemos rellenar una superficie con una trama formada por estas líneas, introduciendo dentro del objeto LAYER el siguiente objeto CLASS:

```
CLASS
  STYLE
```



```
    SYMBOL "diagonal"
    SIZE 8
    COLOR 255 0 0
    OUTLINECOLOR 0 0 0
  END
```

El parámetro SIZE del objeto STYLE no especifica el tamaño de la línea, determina la distancia entre las líneas. El tamaño de las líneas que rellenan áreas no se puede modificar. Es siempre 1.

Símbolos PIXMAP

Estos se forman a partir de una imagen ráster (GIF O PNG). Si tenemos la imagen bola.png:

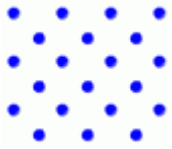
Definimos el símbolo:

```
SYMBOL
  NAME "bola"
  TYPE PIXMAP
  IMAGE "bola.png"
END
```

E insertamos en el objeto CLASS lo siguiente:

```
    CLASS
          STYLE
          SYMBOL "bola"
          COLOR 0 0 0
          END
     END
```

Obtendremos un área:



Para una capa de puntos funcionará exactamente igual. Tenemos una serie de puntos en los ficheros activitats.shp, activitats.dbf y activitats.shx que representan las actividades comerciales de Roses. Vamos a marcarlas en nuestro mapa con formas de estrella para que sean bien visibles. En el fichero de símbolos creamos uno:

```
SYMBOL
  NAME "estrella"
  TYPE PIXMAP
  IMAGE "estrella.png"
```

```
  TRANSPARENT 8
END
```

Por supuesto tendremos que disponer de la imagen en el mismo directorio que este fichero. La imagen es estrella.png: ★

Añadimos una LAYER más:

```
 LAYER
   NAME "activitats"
   STATUS default
   DATA "activitats"
   TYPE point
   MAXSCALE 4000
   CLASS
     STYLE
       SYMBOL "estrella"
     END
   END
 END
```

Y como está activada por defecto, cuando tengamos un zoom a una escala mayor a 4000 veremos lo siguiente:

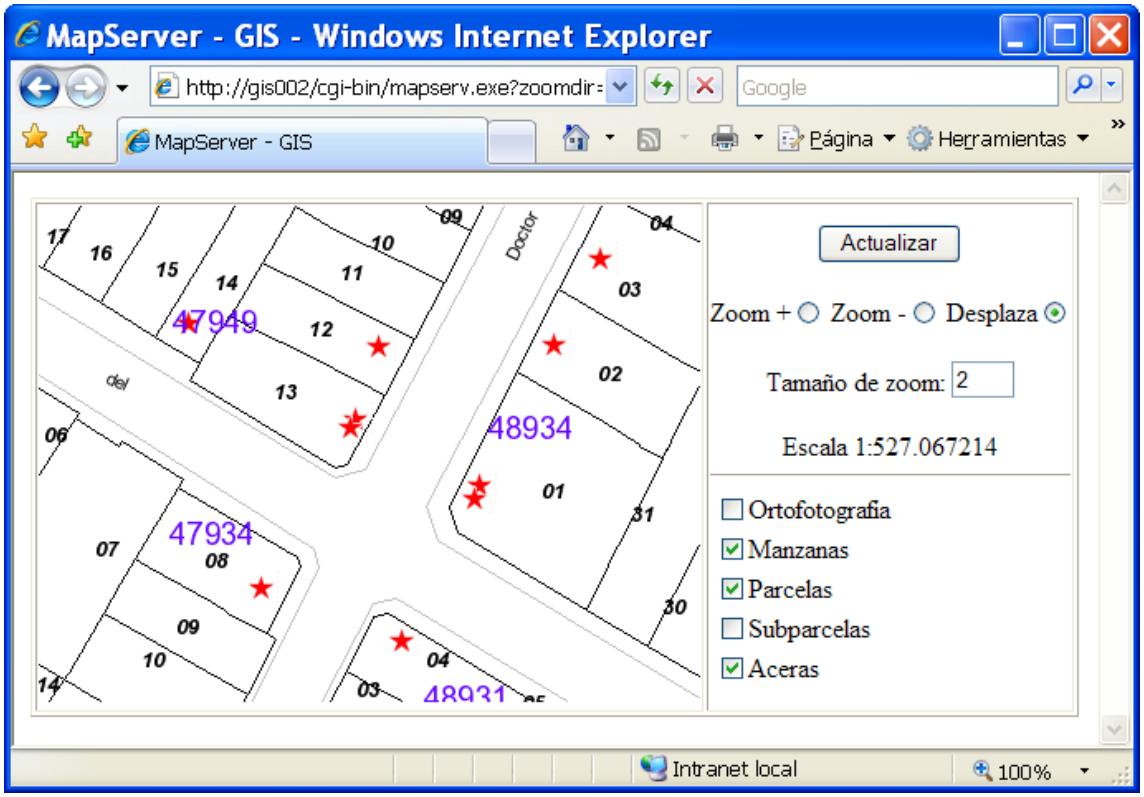


Figura 3.32 Utilización de símbolos PIXMAP

Símbolos Truetype

Son símbolos que se forman con caracteres de fuentes truetype. Los parámetros se definen como en el resto de símbolos, en el fichero de símbolos.
Un ejemplo de los parámetros y sintaxis seria:

```
SYMBOL
  NAME "comercio"
  TYPE TRUETYPE
  FONT "arial"
  CHARACTER "C"
  ANTIALIAS ON
END
```

Como siempre definimos un nombre para el símbolo (T en este caso) y indicamos el tipo de símbolo. Con el parámetro font especificamos el nombre de la fuente. Esta fuente debe estar en el fichero fonts.txt que hemos creado en ejemplos anteriores. Vamos a dibujar los locales comerciales anteriores con la letra "C" y de color rojo, como especificamos en el fichero 'mapa'

```
  LAYER
    NAME "activitats"
    STATUS default
    DATA "activitats"
    TYPE point
    MAXSCALE 4000
    CLASS
      STYLE
        SYMBOL "comercio"
        SIZE 12
        COLOR 255 0 0
      END
    END
  END
```

Y el resultado:

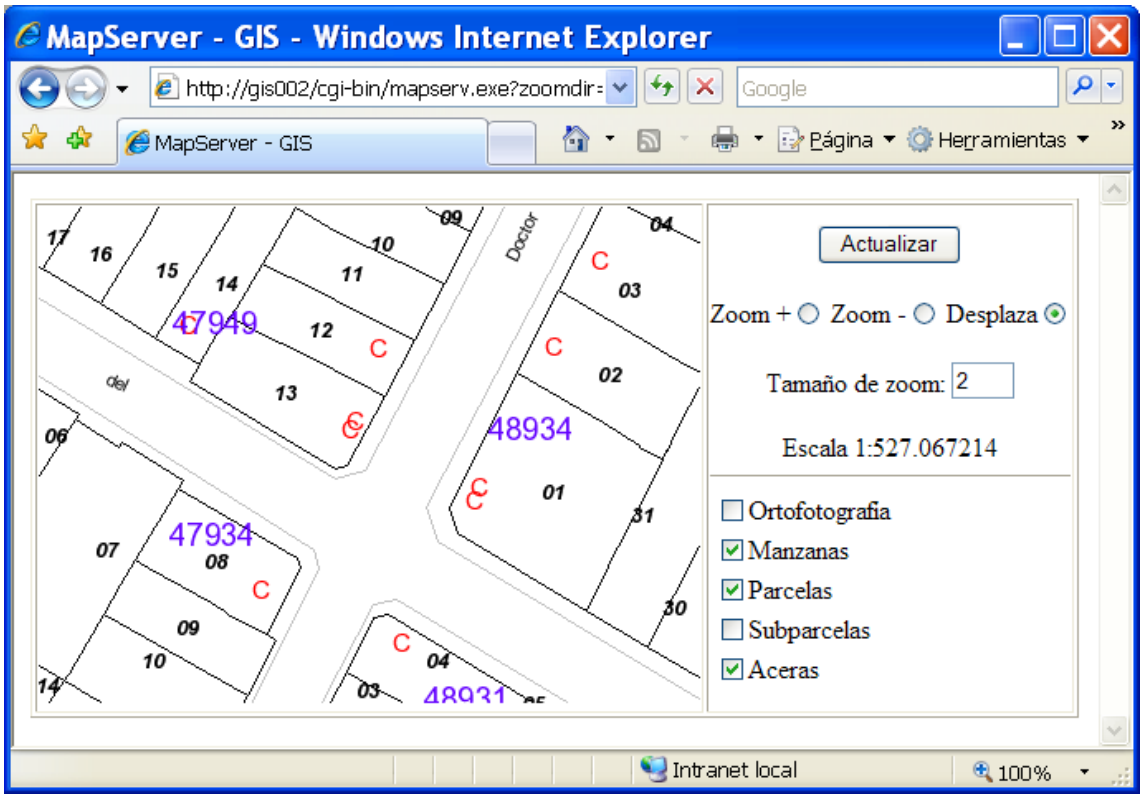


Figura 3.33 Utilización de símbolos TRUETYPE

Hay un detalle en la definición de símbolo: el parámetro ANTIALIAS. Cuando está activado (ON) suaviza el texto.

Figura 3.34 Resultado del parámetro ANTIALIAS

3.6.5. Barras de escala

Se crean mediante el objeto SCALEBAR. Este objeto queda determinado por una serie de parámetros que vamos a describir:

STATUS: similar al parámetro con el mismo nombre de los objetos LAYER. Tiene tres valores posibles on, off y embed. Los dos primeros son obvios. Con embed indicamos a MapServer que inserte la barra de escala en el mapa final. Si no se inserta en el mapa, la deberemos poner en la plantilla. La variable donde se guardará el nombre de la imagen de la barra es [scalebar].

Veamos un ejemplo:

```
SCALEBAR
        STATUS EMBED
        LABEL
                COLOR 0 0 0
                ANTIALIAS ON
                SIZE SMALL
        END
        UNITS METERS
        POSITION lr
        SIZE 150 3
        INTERVALS 3
        STYLE 0
END
```

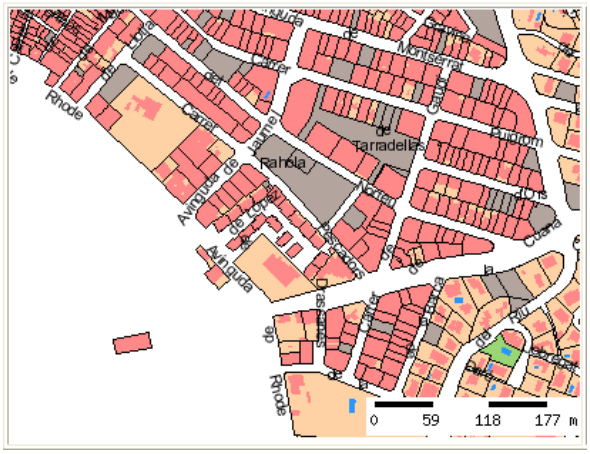
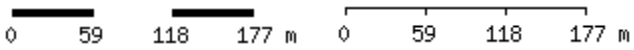


Figura 3.35 Barra de escala.

Se pueden etiquetar con todos los parámetros que hemos visto anteriormente para el objeto LABEL.
El parámetro UNITS nos deja seleccionar las unidades de la barra. Normalmente, para nuestro caso serán meters o kilometers.
Con position indicamos donde situarla, y las opciones son casi las mismas que las posibles posiciones de las etiquetas (ul, uc, ur, ll, lc o lr) ya que la posición cc (justo en el medio del mapa) no tiene mucho sentido. Este parámetro solo es valido cuando la escala va en el interior de la imagen del mapa (STATUS EMBED).
El parámetro SIZE especifica en píxeles el ancho y largo de la escala, y INTERVALS el número de intervalos que la formaran. Por último STYLE tiene dos valores posibles: 0 y 1, y nos permite escoger entre dos estilos diferentes de barra:



STYLE 0 STYLE 1

Figura 3.36 Formatos de la barra de escala

Como todos los objetos que se han descrito, existen muchos más parámetros para configurar. Al no tratarse de un estudio exhaustivo se hace referencia a los más usados y adecuados para el objetivo del trabajo.

3.6.6. Leyendas

Son un recordatorio gráfico del significado de los elementos que aparecen en el mapa. Como todos los objetos, éste también queda definido por el nombre LEGEND y acaba con END.
Podemos escoger que entidades queremos que aparezcan en la leyenda y cuales no (aunque estén en el mapa). Por ejemplo si tenemos una imagen de referencia o un mapa de referencia en un color neutro y encima las entidades importantes que queremos resaltar, no es necesario tener una leyenda muy extensa con todas las entradas que se

vean. Esto se controla nombrando (o no) los objetos CLASS con el parámetro NAME De momento en todos los ejemplos que hemos hecho no hemos nombrado ninguno de estos objetos. Vamos a hacer una leyenda para las subparcelas. Primero nombraremos los objetos CLASS de la LAYER:

```
19  LAYER
20    NAME "SUBPARCELA"
21    STATUS on
22    DATA "CONSTRU"
23    TYPE POLYGON
24    CLASSITEM "CONSTRU"
25    CLASS
26      NAME "Patio"
27     EXPRESSION "P"
28     STYLE
29       COLOR 255 210 166
30     END
31    END
32    CLASS
33      NAME "Piscina"
34     EXPRESSION "PI"
35     STYLE
36       COLOR 43 149 255
37     END
38    END
39    CLASS
40      NAME "Amarre"
41     EXPRESSION "ETQ"
42     STYLE
43       COLOR 187 221 255
44     END
45    END
46    CLASS
47      NAME "Solar"
48     EXPRESSION "SOLAR"
49     STYLE
50       COLOR 180 164 158
51     END
52    END
53    CLASS
54      NAME "Jardín"
55     EXPRESSION "JD"
56     STYLE
57       COLOR 155 214 116
58     END
59    END
60    CLASS
61      NAME "Construcción"
62     STYLE
63       COLOR 255 136 136
64     END
65    END
66  END
```

El objeto LEGEND será:

```
LEGEND
  STATUS embed
  IMAGECOLOR 223 223 223
  POSITION ll
  LABEL
    TYPE TRUETYPE
    FONT "arial"
    SIZE 10
    COLOR 0 0 0
  END
END
```

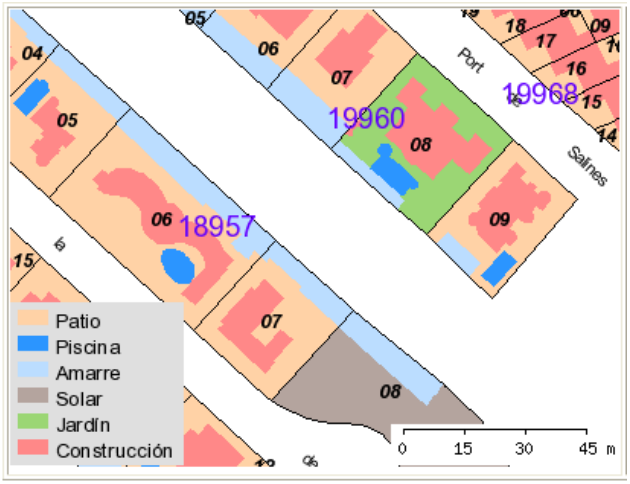
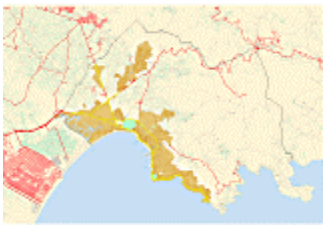


Figura 3.37 Leyenda integrada en el mapa

El esquema es muy similar a la barra de escala que hemos descrito anteriormente. El parámetro STATUS se usa exactamente igual. En este caso hemos insertado la leyenda en la imagen del mapa (como la barra de escala). IMAGECOLOR es el color de fondo del cuadro que ocupa la leyenda (gris en el ejemplo).
El parámetro POSITION y el objeto LABEL ya los hemos descrito en varias ocasiones.

3.6.7. Mapas de referencia

Este tipo de mapas sirve para situar la extensión de la vista que tenemos del mapa en uno general, que queda marcada con un rectángulo o un símbolo cuando este rectángulo es muy pequeño.
Como siempre, el objeto queda definido por la palabra REFERENCE y END. El mapa de referencia necesita una imagen para usarla de referencia. El formato de la imagen debe ser GIF y se aconseja que sean muy esquemáticos y de pequeño tamaño:


ref.gif


ref2.gif

ref3.gif

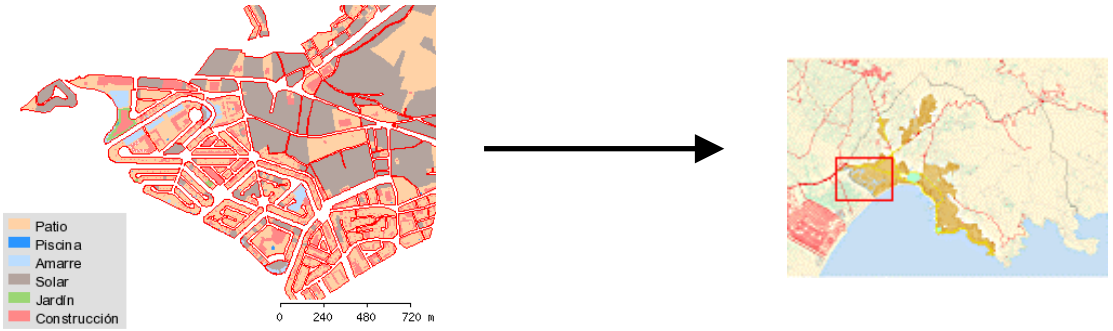
Figura 3.38 Ejemplos de mapas de referencia.

Estas imágenes las guardaremos en una carpeta nueva dentro del directorio de trabajo llamada ‘ref’ (C:\ms4w\apps\tfc\ref).
Un mapa de referencia simple puede tener el siguiente aspecto:

```
REFERENCE
  STATUS ON
  IMAGE "ref/ref.fig"
  SIZE 160 110
  EXTENT 508708 4675468 522449 4684779
  COLOR -1 -1 -1
  OUTLINECOLOR 255 0 0
  MINBOXSIZE 5
  MARKER “estrella”
  MARKERSIZE 8
END
```

El parámetro STATUS aquí solo admite dos valores ON y OFF. Cuando se activa (ON) MapServer genera la imagen correspondiente para que después pueda ser insertada en el archivo plantilla. La imagen de referencia se especifica mediante el parámetro IMAGE. El camino para encontrar la imagen se puede definir relativa (respecto al fichero ‘mapa’) o absolutamente.
Con los parámetros SIZE y EXTENT estamos georreferenciando la imagen para que MapServer sea capaz de calcular en todo momento donde y de que manera tiene que

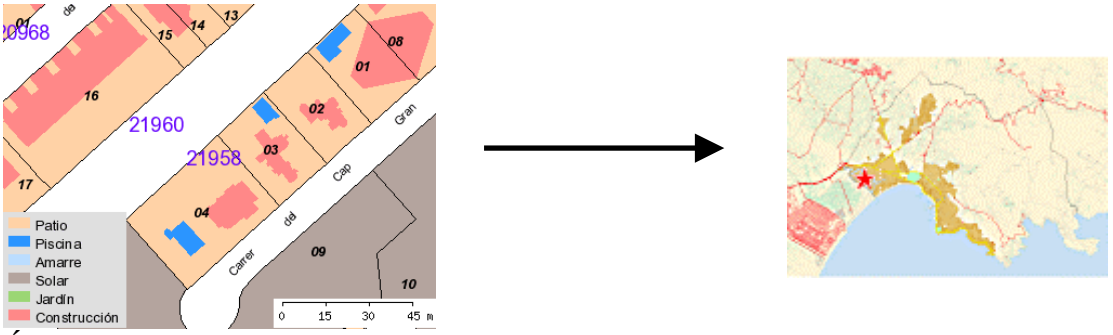
dibujar el marcador de la posición. Nótese que las coordenadas de EXTENT no corresponden a las que venimos poniendo en todos los ejemplos en el parámetro con el mismo nombre del objeto MAPA. Esto es así porqué los mapas de referencia que queremos utilizar abarcan la extensión de todo el término municipal donde se ha hecho el estudio.
El parámetro COLOR indica el color de la caja o símbolo que nos marca la posición. En este caso, al indicar –1 –1 –1 MapServer lo interpreta como transparente. OUTLINECOLOR es el color de los lados de la caja o símbolo.
El resultado es:



Área visualizada en el mapa Mapa de referencia creado

Figura 3.39 Señalización de un área grande en el mapa de referencia.

Cuando estemos visualizando muy detalladamente una zona, puede ocurrir que el rectángulo que se marca en el mapa de referencia sea tan pequeño que no se vea. Las tres siguientes líneas solucionan el problema. Cuando uno de los lados del rectángulo sea menor que lo especificado en MINBOXSIZE (5 píxeles en este caso) en vez de rectángulo usará el símbolo definido en MARKER y del tamaño MARKERSIZE.
Se puede utilizar cualquier símbolo definido en el fichero de símbolos o si no se indica nada el valor por defecto es una cruz.



Área visualizada en el mapa Mapa de referencia creado

Figura 3.40 Señalización de un área pequeña en el mapa de referencia.

El mapa de referencia también es interactivo. Si hacemos un clic en cualquier parte de él, desplazamos el centro de la vista que tenemos a ese punto manteniendo la escala.

3.6.8. Reestructuración de la aplicación

Hasta el momento se han establecido las bases para mejorar el diseño, las capacidades y la visualización a través de la manipulación y creación de objetos gráficos. Ahora vamos a 'ordenar' el archivo plantilla.
Lo primero que haremos es agrandar el área de mapa. En la imagen anterior se ve que entre la barra de escala, la leyenda y su tamaño actual, la visión de los elementos en el mapa es muy reducida.

```
01  NAME "ejemplo07"
02  SIZE 600 600
03  IMAGETYPE png
04  EXTENT 510720 4676215 519430 4683355
```

Con este simple cambio ya notamos una mejoría bastante notable. Veámoslo:

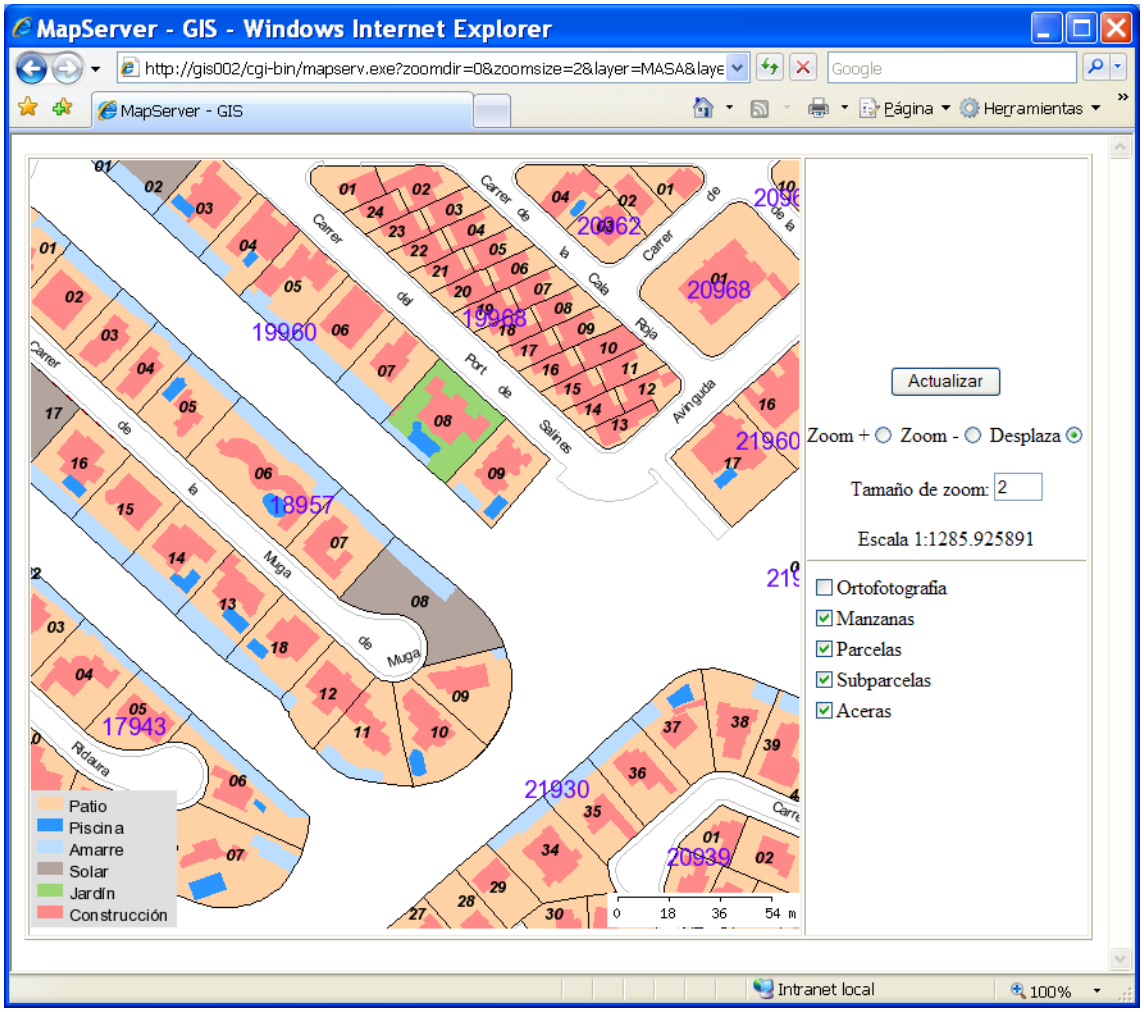


Figura 3.41 Resultado de la ampliación del área de mapa.

El aspecto es muy similar a las aplicaciones cartográficas estándar. Lo siguiente que haremos será quitar la leyenda y la barra de escala del mapa y insertarlas junto con el mapa de referencia en el fichero plantilla.

```
<html>
<head>
  <title>MapServer - GIS</title>
</head>
<body>
<form method="GET" action="[program]" name="mapserv">
<table border="1">
      <tr>
      <td width="600" height="600" align="center">
      <input type="image" name="img" src="[img]" width="[mapwidth]" height="[mapheight]" border="0">
      </td>
      <td valign="top" >
      <center><input type="submit" value="Actualizar"></center>
   <p align="center">Zoom +<input type=radio name=zoomdir  value=1 checked [zoomdir_1_check]>
       Zoom -<input type=radio name=zoomdir value=-1 [zoomdir_-1_check]>
       Desplaza<input type=radio name=zoomdir value=0 [zoomdir_0_check]>
   <p align="center">Tamaño de zoom: <input type=text name=zoomsize size=2 value=[zoomsize]>
    <hr size="1">
        <table>
          <tr><td colspan="3"><input type="checkbox" name="layer" value="orto25m" [orto25m_check]>Ortofotografía</td></tr>
          <tr><td colspan="3"><input type="checkbox" name="layer" value="MASA" [MASA_check]>Manzanas</td></tr>
          <tr><td colspan="3"><input type="checkbox" name="layer" value="PARCELA" [PARCELA_check]>Parcelas</td></tr>
          <tr><td colspan="3"><input type="checkbox" name="layer" value="SUBPARCELA" [SUBPARCELA_check]>Subparcelas</td></tr>
          <tr><td colspan="3"><input type="checkbox" name="layer" value="aceras" [aceras_check]>Aceras</td></tr>
        </table>
     <hr size="1">
     <center><input name="leyenda" type="image" src="[legend]" border="0"></center>
        <hr size="1">
     <center><input name="ref" type="image" src="[ref]" border="0"></center>
     </tr>
        </td>
<tr>
<td valign="middle"><input name="escala" type="image" src="[scalebar]" align="middle" border="0">  1:[scale]</td>
</tr>
</table>
<input type="hidden" name="imgxy" value="[center]">
<input type="hidden" name="imgext" value="[mapext]">
<input type="hidden" name="map" value="[map]">
```

```
39  <input type="hidden" name="program" value="[program]">
40  </form>
41  </body>
42  </html>
```

Con las líneas 27 y 29 insertamos la leyenda y el mapa de referencia en la misma columna que los controles de capas y de navegación. Y en la línea 33 creamos una nueva fila en la que ponemos la barra de escala y la escala numérica que antes teníamos en la columna de los controles.
La aplicación con estos cambios tendrá este aspecto:

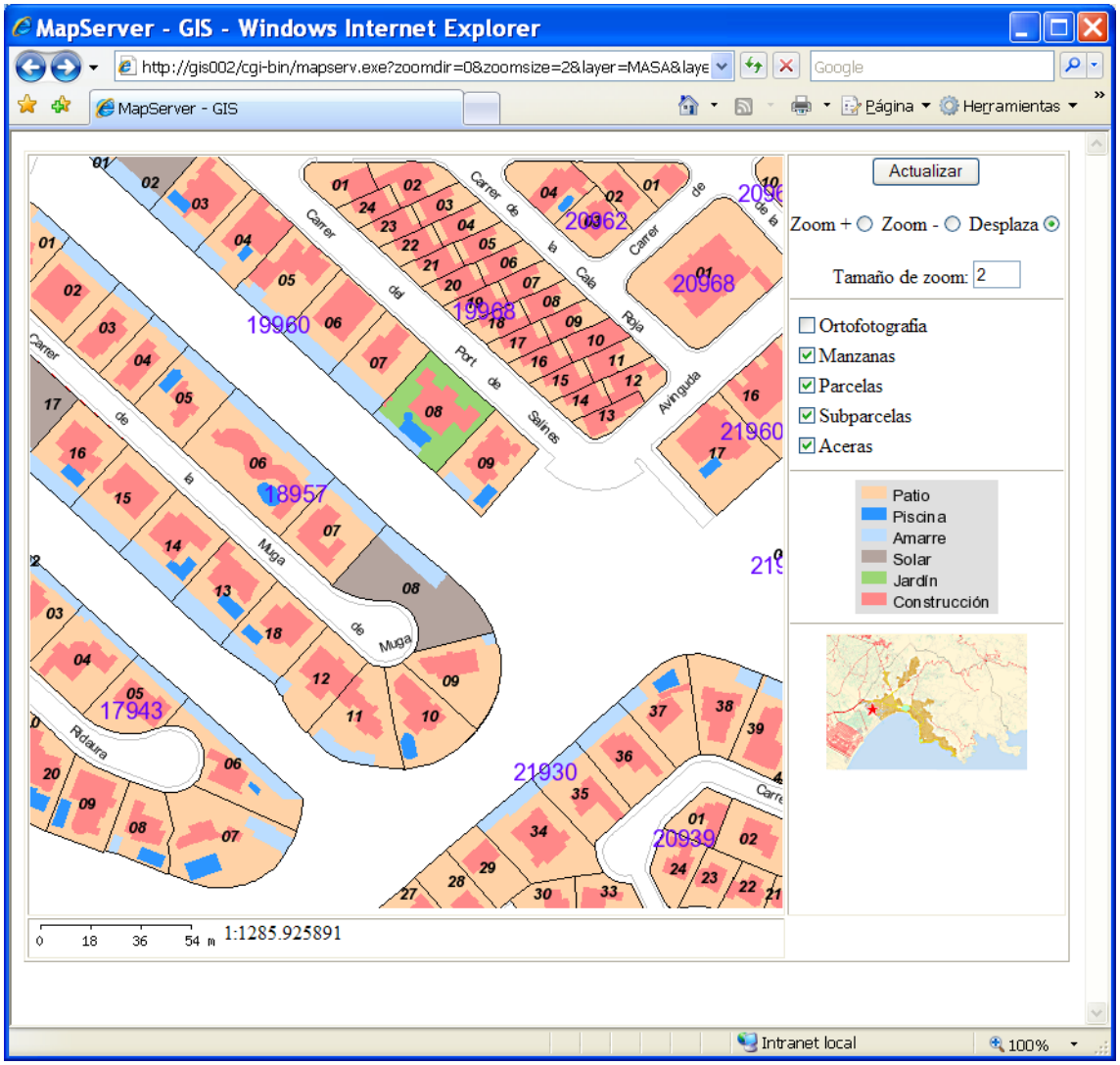


Figura 3.42 Desplazamiento de la leyenda y del mapa de referencia fuera del área de mapa.

Antes de dar por finalizada esta primera reestructuración podemos hacer un último cambio: substituiremos las palabras ‘zoom+’, ‘zoom-‘ y ‘desplaza’ de las herramientas de navegación por unas imágenes que ocuparán menos espacio:
Tenemos las imágenes en formato GIF en una carpeta llamada ‘img’ dentro de la carpeta de trabajo (C:\ms4w\apps\tfc\img):

mas.gif menos.gif desplazar.gif

Figura 3.43 Imágenes de los botones de zoom y desplazamiento.

Las líneas del archivo plantilla modificadas son las siguientes:

14 <p align="center"><img src="http://GIS002/tfc/img/mas.GIF" width="25" height="25" border="0" align="absmiddle"><input type=radio name=zoomdir value=1 checked [zoomdir_1_check]>
15 <img src="http://GIS002/tfc/img/menos.GIF" width="25" height="25" align="absbottom"><input type=radio name=zoomdir value=-1 [zoomdir_-1_check]>
16 <img src="http://GIS002/tfc/img/desplazar.GIF" width="25" height="25" align="absbottom"><input type=radio name=zoomdir value=0 [zoomdir_0_check]>

En el mapa anterior vemos como la leyenda no queda muy bien con el fondo gris. La pondremos del mismo color que el fondo de la aplicación (blanco) cambiando el parámetro IMAGECOLOR del archivo ‘mapa’ a 255 255 255.

El resultado final con todos los cambios descritos es ya un visor de cartografía bastante completo:

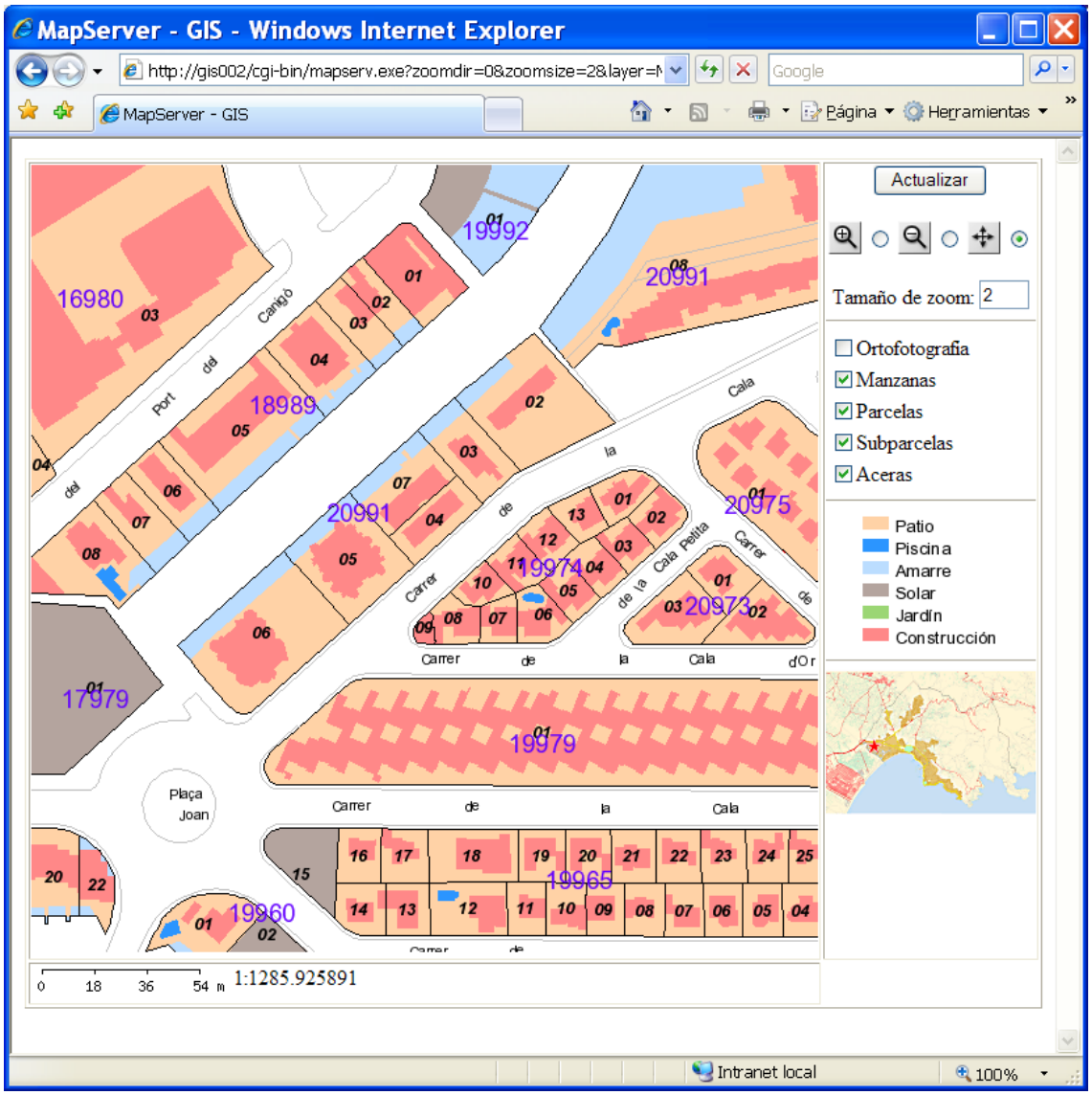


Figura 3.44 Aspecto de la aplicación tras aplicar los cambios.

3.7. Consultas

Además de las capacidades de visualizar información gráfica, MapServer es capaz de realizar una amplia gama de consultas. Éstas pueden ser tanto espaciales (con las que se seleccionan elementos en base a su localización) como de atributos (con las que se seleccionan elementos en base al valor de sus atributos). Para llevar a cabo estos tipos de consultas sin programación, MapServer utiliza ficheros plantilla para hacerlas y para presentar los resultados.
Hasta este momento, las aplicaciones que hemos creado se ejecutaban en el modo 'browse' (navegar) que es el modo por defecto. En este modo, cuando pinchamos en la imagen del mapa, el navegador guarda las coordenadas del clic, del valor de zoom y de la dirección (entre otras). Esta información es devuelta al servidor que la envía a MapServer. Basándose en esta información, se crea una nueva imagen, se examina el fichero plantilla para sustituir las variables necesarias y se devuelve de nuevo al navegador. El usuario en ningún momento tiene acceso a la información gráfica o alfanumérica, es MapServer el que la gestiona. El modo consulta hace que el usuario pueda acceder a esta información, presentando los resultados tanto en forma tabular como gráficamente.

3.7.1. Tipos de consultas

Existen más de dos docenas de modos en MapServer de los que dieciocho son para diferentes tipos de consultas. De éstos, nueve producen tanto mapas como resultados tabulares y los otros nueve presentan solo resultados espaciales de las consultas.
En estos modos se pueden definir los criterios de selección de las siguientes maneras:

- Seleccionando un punto o un área del mapa con el ratón.
- Introduciendo las coordenadas de un punto o un área.
- Introduciendo una expresión para seleccionar según los atributos.
- Introduciendo el identificador único de un archivo shp de una entidad.

En cualquiera de los modos consulta, el navegador envía las coordenadas o la expresión, además de otras variables a MapServer, pero en lugar de dibujar el mapa (como hacía en el modo navegación), MapServer busca en una o más LAYERs y rellena los ficheros plantilla con la información de las entidades o los atributos que cumplen los criterios de selección.
Para que sea posible ejecutar una consulta, el fichero 'mapa' debe tener al menos una capa consultable, esto es que esté activa (con el parámetro STATUS on o default) y que tenga una plantilla de consulta asociada. Si existe una capa consultable, MapServer busca en el conjunto de datos entidad por entidad los elementos que cumplan los criterios espaciales o de atributos.
Algunos modos buscan únicamente en una LAYER y devuelven solo un resultado (por ejemplo los modos QUERY e ITEMQUERY). Otros, como NQUERY, buscan en todas las LAYERs consultables y devuelven todos los resultados. Estos últimos se distinguen por anteponer una N antes de la parte QUERY de la palabra que los designa, como por ejemplo ITEMNQUERY. Es posible especificar que capa sea la consultable en cada momento mediante la variable 'qlayer'.

Las capas se consultan en el orden en que se encuentran definidas en el fichero ‘mapa’.

Los modos de consulta son:

Modo	Tipo	Selecciona
QUERY	Espacial	Elemento más cercano a una distancia dada
NQUERY	Espacial	Todos los elementos a una distancia dada
ITEMQUERY	Atributos	El primer elemento cuyo atributo cumple la condición
ITEMNQUERY	Atributos	Todos los elementos cuyos atributos cumplen la condición
FEATUREQUERY	Espacial	La primera entidad (definida en la variable ‘slayer’) de tipo POLYGON y todos los elementos que estén a una distancia dada de ésta.
FEATURENQUERY	Espacial	Todas las entidades POLYGON (definidas en la variable ‘slayer’) que estén a una distancia dada y todos los elementos que estén a otra distancia dada de éstas.
ITEMFEATUREQUERY	Atributos y espacial	La primera entidad (definida en la variable ‘slayer’) cuyos atributos cumplen la condición y todos los elementos que estén a una distancia dada de ésta
ITEMFEATURENQUERY	Atributos y espacial	Todas las entidades (definidas en la variable ‘slayer’) cuyos atributos cumplen la condición y todos los elementos que estén a una distancia dada de éstas
INDEXQUERY	De índice	La entidad que tiene el índice especificado

Los otros nueve modos son los que únicamente producen mapas, y se denominan ‘map-only’. A todo modo descrito en la tabla anterior le corresponde un modo ‘map-only’que se identifica de la misma manera: el modo ‘map-only’ asociado al modo NQUERY es NQUERYMAP, el asociado a ITEMQUERY es ITEMQUERYMAP, y así sucesivamente. En este trabajo nos centraremos únicamente en los modos que utilizaremos en la aplicación final, aunque se describirán brevemente los más importantes.

3.7.2. Plantillas de consulta

Cuando MapServer evalúa una consulta y devuelve los resultados, los introduce en uno (o más) ficheros plantilla que reenvía al navegador. MapServer no conoce que tipo de resultado obtendrá al hacer una consulta, por lo que habrá que definir las plantillas de modo que abarquen todos los casos posibles. Esto se controla especificando una jerarquía en la plantilla.
Cuando definíamos los tipos de consultas en el apartado anterior, ya avanzamos que en el fichero ‘mapa’ deberíamos tener al menos una plantilla de consulta asociada. Esta

asociación se hace mediante el parámetro TEMPLATE, en el que insertamos la ruta de acceso y el nombre de la plantilla:

```
TEMPLATE "consulta_parcelas.htm".
```

En este caso la plantilla se encuentra en el mismo directorio que el fichero 'mapa'.
Estas plantillas se pueden definir en los objetos WEB, LAYER, CLASS y JOIN. Dependiendo del nivel (objeto) en que se defina una plantilla de consulta afectará en como y cuando la utilizará MapServer.

Objeto WEB
En este objeto se pueden utilizar tres parámetros para especificar plantillas de consulta: HEADER, FOOTER y EMPTY.
Dado que MapServer no conoce el tipo de resultados antes de ejecutar las consultas, lo que hace es dividir la tarea en tres pasos:

1- Produce una cabecera (HEADER) con los tags HTML de inicio (como <html>, <body>, etc.) y substituye las variables de resumen de los resultados que deseemos como [nr] (número de resultados), [nl] (número de capas que han devuelto resultados, etc.
2- Usa las plantillas de consulta de los objetos LAYER y CLASS para presentar los resultados de cada elemento.
3- Produce un pie de página (FOOTER) para cerrar los tags de inicio.

Los parámetros HEADER y FOOTER especifican la ruta de estos ficheros de cabecera y pie.
El parámetro EMPTY contiene la ruta del fichero HTML que se usará si la consulta no devuelve ningún resultado. Si no se usa este parámetro y no se encuentra ningún resultado, MapServer devuelve el siguiente mensaje de error:

"msQueryByPoint(): Search returned no results. No matching record(s) found."

Objeto LAYER
Este objeto también tiene tres parámetros referentes a las plantillas de consulta: HEADER, FOOTER y TEMPLATE. Los dos primeros son muy similares a los parámetros con el mismo nombre del objeto WEB. Si partimos de que con HEADER del objeto WEB abríamos los tag generales de la página, con HEADER del objeto LAYER podemos abrir una tabla y poner los títulos donde insertaremos los resultados y también poner un resumen de los resultados de este objeto LAYER usando la variable [nlr] que nos muestra el número de resultados de la capa actual. El parámetro FOOTER lo utilizaremos para cerrar la tabla.
Con TEMPLATE definimos la ruta de otro fichero HTML que configuraremos para que nos muestre los detalles de los atributos del conjunto seleccionado. Recordemos que este parámetro es imprescindible para indicar a MapServer que esta capa es seleccionable.

Objeto CLASS
En este objeto únicamente encontramos un parámetro relativo a las plantillas de consulta: TEMPLATE. Es exactamente igual al del mismo nombre que el del objeto

anterior, aunque si se define a este nivel se usará únicamente para las entidades seleccionadas de esta clase.
Así podremos personalizar la presentación de resultados para cada entidad que hayamos definido mediante un objeto CLASS. Si recordamos los ejemplos que hemos hecho en los capítulos anteriores, las subparcelas catastrales formaban parte de un objeto LAYER llamado 'subparcela' y las habíamos dividido en seis clases diferentes (patio, piscina, amarre, solar, jardín y construcción) clasificadas según el valor de un atributo. Cada una de estas clases era un objeto CLASS, así si quisiéramos podríamos definir una plantilla diferente para presentar los resultados de cada clase.

3.7.3. El objeto QUERYMAP

Este objeto crea un mapa que resalta los resultados de una consulta. Como hemos señalado anteriormente, existen 9 tipos de consultas que producen solo mapas (map-only) y las otras 9 por defecto sólo producen resultados tabulares. Este objeto permite crear mapas de resultados para este último tipo de consultas.
El objeto QUERYMAP produce una imagen a la que se accede mediante la variable [img], que es igual a la variable para designar la variable que designa una imagen de mapa normal. Esto no tiene porqué representar ningún problema, ya que el resultado de las consultas ya sea tabular o una imagen normalmente se presentarán en una plantilla específica para mostrar resultados.
El objeto empieza por la palabra QUERYMAP y finaliza con END. El parámetro COLOR define el color para resaltar los resultados de la consulta (por defecto es el amarillo). SIZE es el tamaño (anchura y altura en píxeles) de la imagen creada. Por defecto es el mismo que el del fichero 'mapa'. STATUS activará (ON) o desactivará (OFF) la creación de esta imagen.
Con STYLE podemos controlar de que manera resaltar (o no) los resultados:

NORMAL: no resalta los resultados. Dibuja las entidades tal y como determina el fichero 'mapa'.

Figura 3.45 Objeto QUERYMAP con el parámetro STYLE NORMAL.

HILITE: Los elementos seleccionados se dibujan en el color especificado (rojo en este caso)

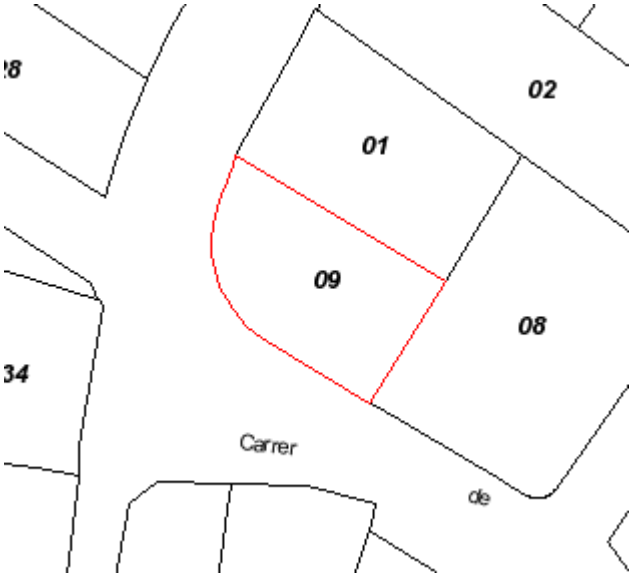


Figura 3.46 Objeto QUERYMAP con el parámetro STYLE HILITE.

SELECTED: solo se dibujarán las entidades de la capa que cumplan los criterios de selección (el resto de capas se dibujarán igualmente).

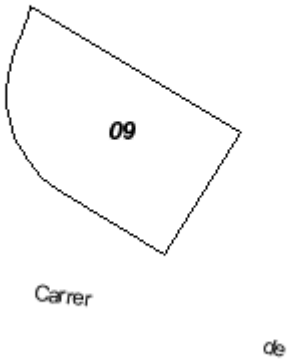


Figura 3.47 Objeto QUERYMAP con el parámetro STYLE SELECTED.

Hasta este momento se han definido los conceptos que intervienen en una consulta cualquiera. Vamos a describir ahora mediante unos ejemplos las capacidades de consulta con las que dotaremos la aplicación final.

3.7.4. El modo QUERY

Este modo realizala consulta espacial más simple: una consulta a través de un punto del mapa. Las coordenadas de la imagen (en píxeles) de un clic en el mapa se devuelven a MapServer mediante los valores de dos variables CGI: img.x e img.y. Estas coordenadas se usan para buscar en los objetos LAYER hasta encontrar la entidad más próxima (con una tolerancia) y devuelve un único resultado: los atributos de esa entidad encontrada.
Esta tolerancia se especifica en el fichero 'mapa' con el parámetro TOLERANCE. Con otro parámetro (TOLERANCEUNITS) definimos que tipo de unidades estamos indicando en TOLERANCE (pixels, feet, inches, kilometers, meters, miles o dd).
Usaremos este tipo de consulta para mostrar los atributos de una parcela cuando pinchemos sobre ella. Este será el ejemplo 08.

En el fichero de inicio (ejemplo08_inicio.html) únicamente cambiaremos la línea donde indicamos el fichero 'mapa' que debe abrir:

```
<input type="hidden" name="map" value="/ms4w/apps/tfc/ejemplo08.map">
```

En el fichero plantilla (ejemplo08.html) vamos a introducir unos botones de opción: uno para navegar (modo browse) como por defecto hacíamos hasta ahora y el otro para consultar (modo query). Ponemos el mismo tipo de botones que usamos en las herramientas de zoom ya que únicamente podemos tener un modo activo en cada momento.
Añadimos las tres siguientes líneas al fichero plantilla del ejemplo 07:

```
30 <hr size="1">
31 <input type="radio" name="mode" value="browse" checked>
   <b>Navegación</b><br>
32 <input type="radio" name="mode" value="query"> <b>Información</b><br>
```

La primera línea (31) es una línea horizontal para separar estos botones del mapa de referencia.
Por defecto está activado el valor de 'browse' (línea 31). Para consultar un elemento tendremos que cambiar a modo consulta y pinchar sobre el elemento.

La aplicación tendrá ahora este aspecto:

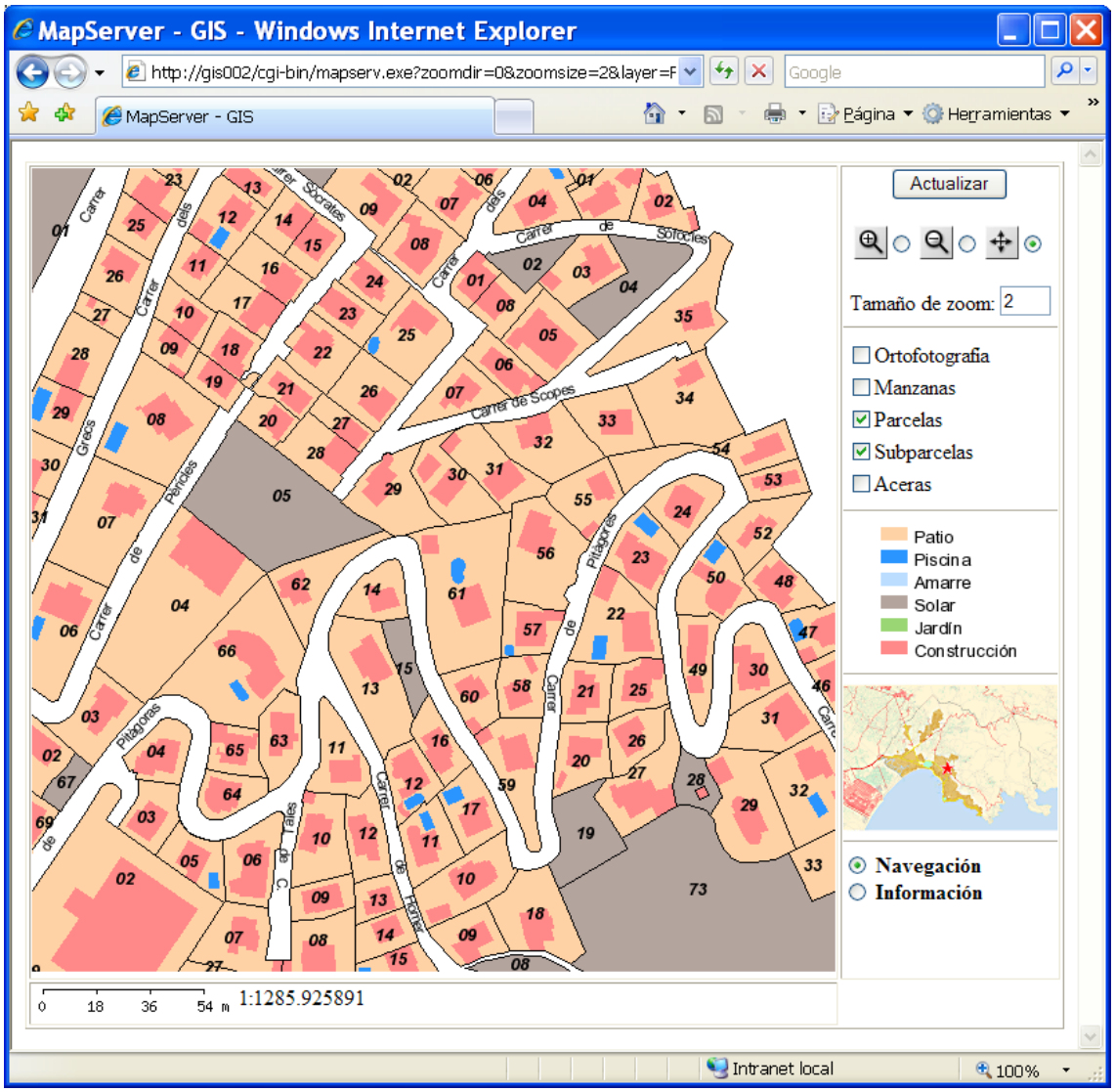


Figura 3.48 Aspecto de la aplicación añadiendo los botones de navegación y consulta.

Como la funcionalidad que queremos añadir es que nos muestre una serie de datos de las parcelas, deberemos crear una plantilla de consulta para ordenar estos datos y la deberemos referenciar en el fichero 'mapa'.
Vamos a modificar primero este último:
Primero crearemos un objeto QUERYMAP para tener gráficamente el resultado de la consulta:

```
QUERYMAP
        STATUS ON
        COLOR 255 0 0
        STYLE HILITE
        SIZE 400 400
END
```

De esta manera nos dibujará la parcela seleccionada de color rojo y nos creará el mapa de resultado de la consulta de 400 por 400 píxeles.

De momento solo queremos que sea consultable la LAYER de las parcelas, por lo que a ésta le añadiremos el parámetro TEMPLATE, para que cuando seleccionemos una nos presente la planilla de consulta que le designemos (consulta_parcelas.html):

```
LAYER
  NAME "PARCELA"
  STATUS ON
  DATA "PARCELA"
  TYPE POLYGON
  LABELITEM "PARCELA"
  LABELMAXSCALE 2000
  MAXSCALE 5000
  TEMPLATE "consulta_parcelas.html"
  CLASS
    STYLE
      OUTLINECOLOR 0 0 0
    END
    LABEL
      TYPE TRUETYPE
      FONT "arialBI"
      SIZE 10
      POSITION CC
      COLOR 0 0 0
    END
  END
END
```

Ahora ya nos queda únicamente crear esta plantilla de consulta. La diseñaremos del siguiente modo: a la izquierda presentaremos un mapa donde se vea resaltada la parcela seleccionada y a la derecha los atributos. Mostraremos dos tipos de atributos:

- De las parcelas. Son los datos almacenados en el fichero dbf. Mostraremos dos de las partes de la referencia catastral (manzana y parcela y la hoja) y el área.
- De la selección. Cuando pinchamos en el mapa para hacer la selección se nos guardan en una serie de variables como las coordenadas del punto pinchado y la capa que consulta.

El fichero de consulta será el siguiente:

```
01 <html>
02 <head>
03   <title>MapServer - GIS</title>
04 </head>
05 <body>
06 <strong>Consulta de los datos de la parcela</strong>
07 <table border="1">
08        <tr>
09        <td align="center">
10        <input type="image" name="img" src="[img]" border="0">
11        </td>
```

```
12      <td align="left" valign="top">
13      <p><strong>Referencia:</strong> [UTM]</p>
14      <p> <strong>Hoja:</strong> [HOJA]</p>
15      <p> <strong>Area:</strong> [AREA] m2</p>
16      <hr size="1">
17      <p> <strong>Capa:</strong> [cl] </p>
18      <p> <strong>X(imagen):</strong> [img.x] </p>
19      <p> <strong>Y(imagen):</strong> [img.y] </p>
20      <p> <strong>X(mapa):</strong> [mapx] </p>
21      <p> <strong>Y(mapa):</strong> [mapy] </p>
22      </td>
23      <tr>
24      <td valign="middle"><input name="escala" type="image"
   src="[scalebar]" align="middle" border="0">  1:[scale]</td>
25      </tr>
26  </table>
27  </body>
28  </html>
```

En las líneas 13 a 21 vemos que las variables de los dos tipos de datos se especifican de la misma manera, mediante corchetes [ ]. Las variables que designan a los atributos de las parcelas son los nombres de las columnas donde están esto datos en el fichero dbf.
Recordemos que para que se ejecute la consulta, a parte de disponer de la plantilla de consulta anterior, la capa debe estar activada (on o default). Como tenemos programado que se activen las parcelas a partir de una escala, para seleccionar una la deberemos tener visible. Si tenemos el siguiente mapa:

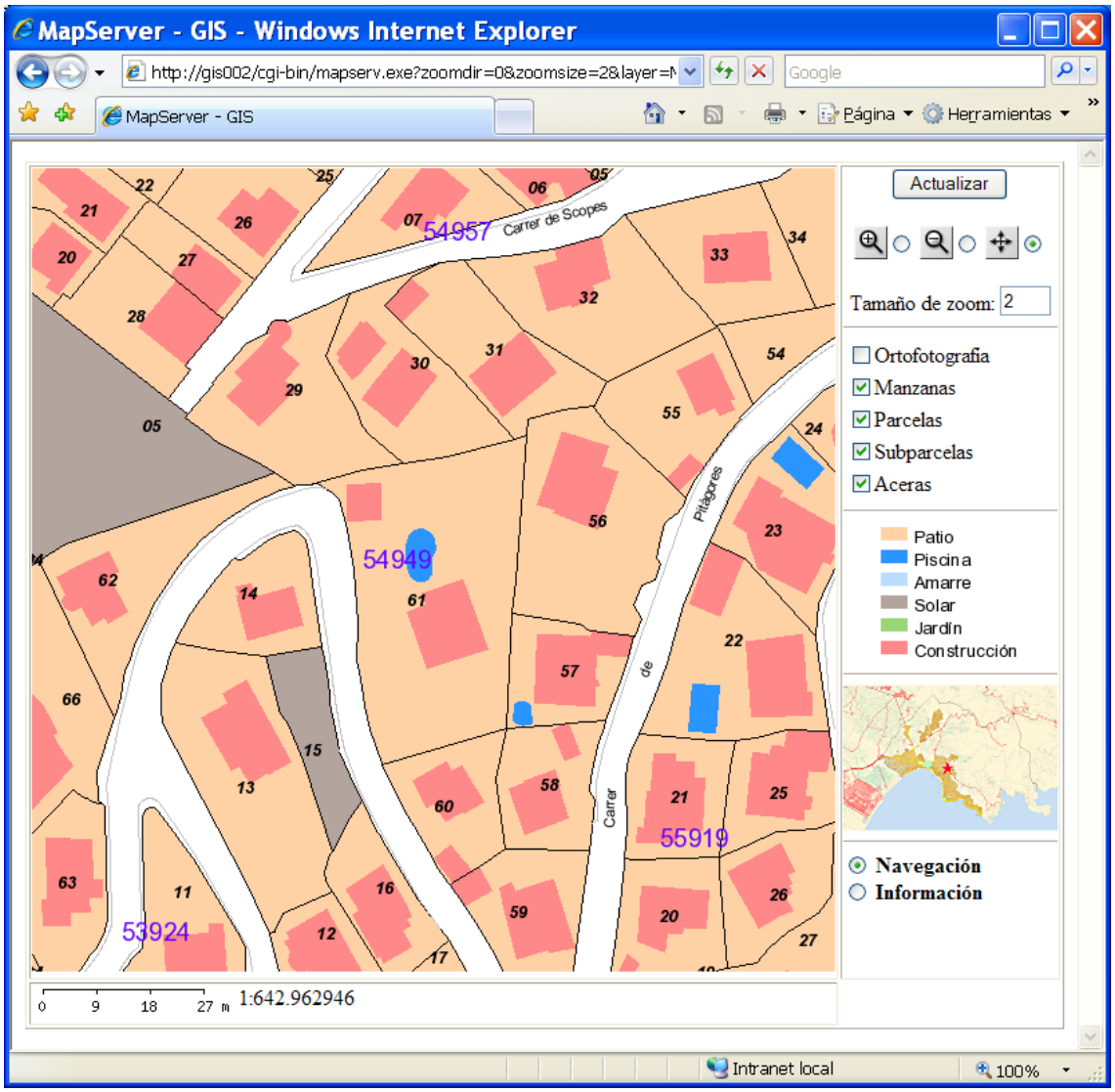


Figura 3.49 Mapa inicial para comprobar el funcionamiento de las consultas.

.

Seleccionamos el modo 'información' y pinchamos sobre la parcela del centro, nos aparecerá lo siguiente:

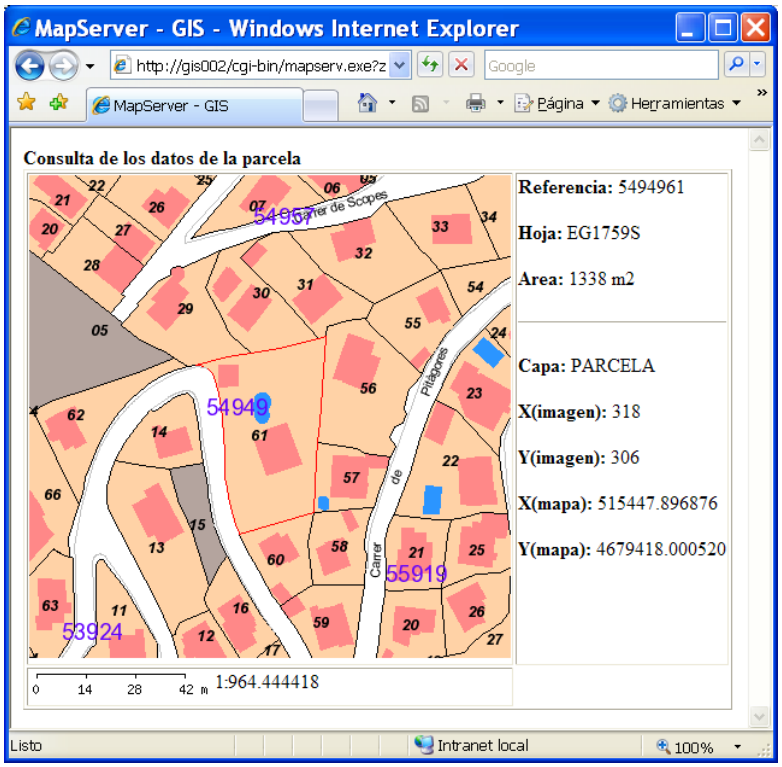


Figura 3.50 Resultado de la consulta.

Como en este caso el objetivo era seleccionar parcelas, no ha hecho falta definir ninguna tolerancia, ya que pinchando en cualquier lugar de la parcela, ésta es seleccionada. Tampoco ha hecho falta definir plantillas de cabecera (HEADER) ni FOOTER, puesto que cada vez sólo seleccionamos un objeto y es más fácil preparar una plantilla personalizada. Este tipo de consultas es el de los más útiles desde el punto de vista de la administración local. El otro tipo más usado es el ITEMQUERY que vamos a tratar a continuación.

3.7.5. El modo ITEMQUERY

El objetivo que perseguimos con esta consulta es localizar una parcela a través de su referencia catastral. Este modo nos selecciona la primera entidad que cumpla la condición que expresemos. Como todas las parcelas tienen una referencia catastral diferente no hará falta usar el modo ITEMNQUERY, puesto que solo obtendremos o un resultado o ninguno.
El fichero de inicio será exactamente igual al del ejemplo anterior, pero cambiando el nombre del fichero 'mapa' a ejemplo09.map.
Este fichero 'mapa' también será idéntico al del ejemplo anterior. No cambia nada, puesto que queremos seleccionar parcelas, y éstas ya son consultables. Tampoco varía nada el fichero de consulta, ya que el resultado será una parcela.
El único que varía es el fichero plantilla, al que añadiremos un cuadro de texto que nos permita introducir la referencia catastral de la parcela que deseamos encontrar. Así, al final del fichero plantilla del ejemplo anterior (ejemplo 08) insertaremos las siguientes líneas (entre el final del formulario y el final del cuerpo html).

```
45 <form name="consultarf" method="GET" action="[program]">
46   <div align="center">Referencia:
47   <input name=qstring type=text value="" size=9 maxlength=9>
48   <input name="submit" type=submit value=Buscar>
49   <input type="hidden" name=mode value=itemquery>
50   <input type=hidden name=map value="C:\ms4w\apps\tfc\ejemplo09.map">
51   <input type="hidden" name="mapext" value="shapes">
52   <input type=hidden name=qlayer value="PARCELA">
53   <input type=hidden name=qitem value="UTM">
54   </div>
55 </form>
56 </body>
57 </html>
```

Con este formulario enviamos los parámetros de la consulta a MapServer a través de unas cuantas variables:

- qstring: es la expresión que utilizaremos para hacer la selección. Esta puede ser una cadena de texto, una expresión regular o una expresión lógica (exactamente igual que cuando tratábamos el capítulo 3.6.3. de clasificación de entidades). En este caso solo introduciremos una cadena de caracteres que será la referencia catastral.
- qlayer: el valor asignado a esta variable es el nombre de la capa (LAYER) que vamos a consultar. Si se omite, buscará en todas las capas.
- qitem: es el nombre del atributo que será consultado. Igual que ocurría con la variable qlayer, si se omite, se buscará por todos los atributos.
- mode: especifica el modo de consulta.

Es aconsejable especificar la capa y atributo cuando los conozcamos (como es el caso), puesto que las consultas tardarán muchísimo tiempo menos.
A parte de estas variables, enviamos otras más para confeccionar el mapa de resultados y la consulta:

- map: el fichero que usaremos para buscar las entidades (que será el mismo que el de la aplicación que estamos ejecutando).
- mapext: la extensión del mapa la ajustamos a los resultados de la selección (a la parcela encontrada).

Tras estas modificaciones la aplicación será ahora:

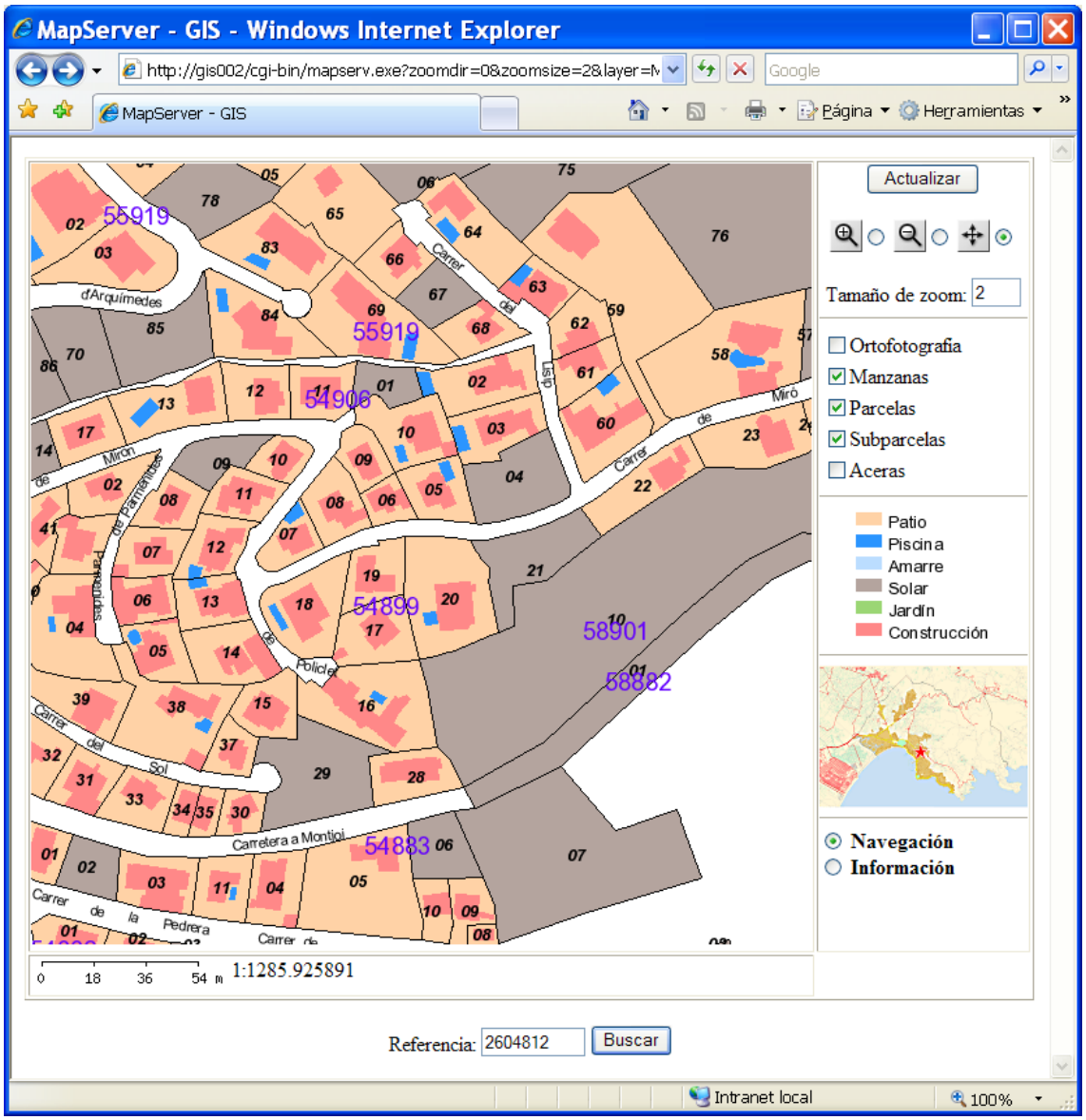


Figura 3.51 Aspecto de la aplicación añadiendo el buscador de parcelas.

Introduciendo el valor de la referencia catastral que queremos buscar, como por ejemplo '2604812', obtenemos el siguiente resultado:

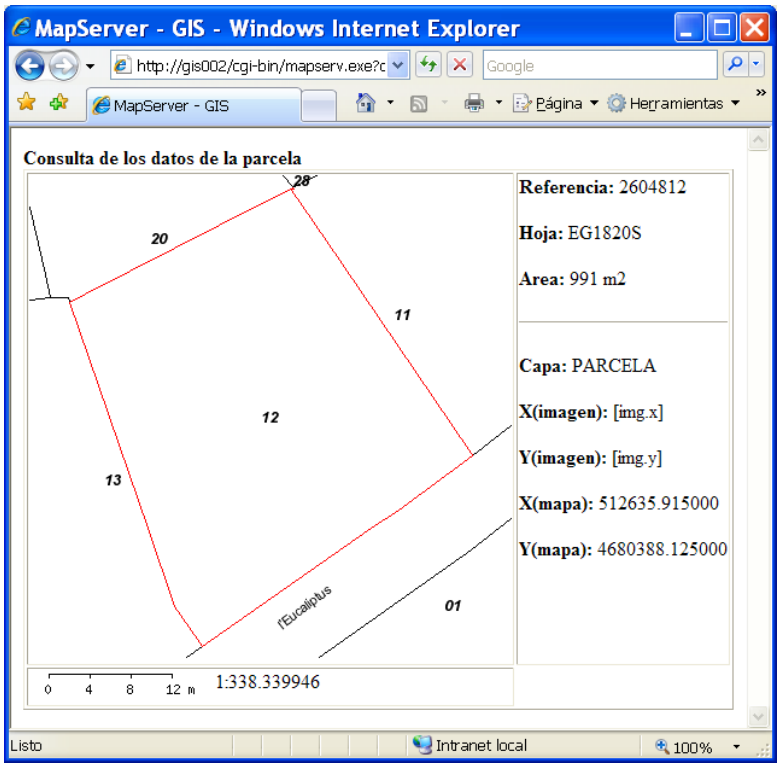


Figura 3.52 Resultado de la búsqueda de parcelas.

Nos calcula la extensión de la parcela y hace un zoom. Por otro lado hace la consulta de los atributos del fichero dbf y nos muestra el resultado, y finalmente nos muestra también los valores de las variables de la consulta (nombre de la LAYER y las coordenadas). Puesto que no hemos pinchado en el mapa en ningún momento para hacer la consulta, las variables x e y de la imagen no tienen valor. La x e y del mapa en este caso se refieren al centro de la extensión.
Si nos fijamos, MapServer únicamente dibuja los elementos de la LAYER (de la manera especificada en el objeto QUERYMAP) y el texto de las calles. Esto es debido a que solo dibuja los elementos que en el fichero 'mapa' tienen el parámetro STATUS igual a DEFAULT.

3.7.6. Selección sin resultados

Probemos introduciendo en el ejemplo anterior un valor que no sea ninguna referencia catastral, por ejemplo '3d3s3s3'. Pinchamos en 'buscar' (o intro):

msQueryByAttributes(): Search returned no results. No matching record(s) found.

Este es el aviso de que MapServer no ha encontrado ningún elemento que cumpla la condición introducida. Existe un parámetro (EMPTY) para el objeto WEB al que podemos asignar el valor de una ruta de un fichero html, en el que podemos introducir un texto avisando que no hay resultados, cosa que hará más vistosa y ordenada nuestra aplicación.
Introducimos esta mejora en el fichero ejemplo09.map:

```
09   WEB
10     TEMPLATE "/ms4w/apps/tfc/ejemplo09.html"
```

```
11     IMAGEPATH "/ms4w/tmp/ms_tmp/"
12     IMAGEURL "/ms_tmp/"
13     EMPTY "sin_elementos.html"
14   END
```

Y ahora creamos el fichero ‘sin_elementos.html’:

```
01 <html>
02 <head>
03   <title>MapServer - GIS</title>
04 </head>
05 <center>
06 <h3>No se han encontrado elementos</h3>
07 <h3>Vuelve a la página anterior</h3>
08 </center>
09 <body>
10 </body>
11 </html>
```

Ahora volvemos a introducir un valor que no sea de tipo referencia catastral y el resultado será:

Figura 3.53 Mensaje de búsqueda sin resultados.

Estos dos tipos de consulta son los más básicos, pero también los más útiles. En las aplicaciones que se usan actualmente en muchas administraciones locales, ya sean de tipo OpenSource como MapServer o de tipo comercial son las herramientas más utilizadas, por lo que para no extender demasiado el estudio no tocaremos el resto. No olvidemos que MapServer no es un GIS completo, y para consultas más especializadas normalmente se usa una herramienta más técnica.

4. La aplicación 'Mapserver Roses'

Como bien se comentó cuando definíamos la justificación y los objetivos del proyecto, la necesidad primordial para la organización era disponer de una herramienta para consultar cartografía y sus atributos en su entorno de trabajo, es decir, exclusivamente para los trabajadores y directivos municipales (intranet). Si bien todo el sistema puede ponerse a disposición del público en general a través de internet, el diseño de la aplicación responde únicamente al punto de vista de la gestión municipal.
Hemos repetido en numerosas ocasiones en este trabajo las limitaciones de MapServer, que aún no siendo un sistema de información geográfica completo, resuelve mucho mejor que uno completo la mayor parte de las necesidades municipales. Estas consultas simples pueden representar más del 99% del total de consultas. Si contamos el número de consultas simples (información de parcelas pinchando sobre ellas, por ejemplo) y el número de complejas (número de habitantes de cada nacionalidad de cada manzana, por ejemplo) veremos reflejada la gran diferencia.
Podemos agrupar en tres grandes bloques la gestión de la información geográfica que se llevan a cabo en un ayuntamiento:

Figura 4.1 Herramientas utilizadas en el Ayuntamiento.

Cada bloque de trabajos requiere unas herramientas muy específicas, y por lo tanto un programa diferente ya que en la actualidad no hay un software único que reúna las características de los tres bloques con suficiente calidad.
Hay diversas aproximaciones como puede ser la solución propuesta por la empresa ABSIS (www.absis.es) que ha desarrollado un conjunto de módulos interrelacionados entre ellos, que usan software comercial como base (como MicroStation Geographics y MapInfo). De la misma manera, muchos grandes fabricantes de software como Autodesk, Esri o Geomedia ofrecen diferentes módulos especializados en uno de los bloques anteriores pero interrelacionados entre ellos.
Una de las grandes desventajas de todos estos productos (a parte del precio elevado) es que hacen que nos 'casemos' con su sistema, perdiendo la posibilidad de aprovechar uno de los módulos de otro sistema que realmente es más efectivo. Por ejemplo si adquirimos el paquete de Autodesk, no tendría sentido hacer la edición con MicroStation.
En el caso que nos ocupa hemos optado por efectuar las labores de edición de cartografía con MicroStation. Todos los delineantes y técnicos municipales lo están usando en estos momentos.
Para crear topologías e insertar los atributos de las entidades que lo requieran y para ejecutar consultas complejas, usamos Geomedia. Geomedia y MicroStation tienen una gran compatibilidad, y no requiere mucho esfuerzo ni tiempo volcar información CAD

para tratarla y convertirla en información SIG. Estos dos tipos de información se pone a disposición de todo el personal municipal a través de MapServer.
En este capítulo vamos a centrarnos en este último bloque de trabajos: visualizar y consultar la información que es el objetivo del proyecto.

4.1. Diseño conceptual

Vamos a diseñar una herramienta que de una solución a la casi totalidad de las necesidades de información geográfica de un ayuntamiento que ya vimos con el tema de justificación del proyecto (Capítulo 1). Puede sonar un tanto pretencioso, pero analizando estas necesidades observamos que son muy simples.
En la siguiente tabla exponemos la información, su tipología y las capacidades:

Información	Tipo	Capacidades
Catastral	Local y WMS	Gráficas y alfanuméricas
Urbanística	Local	Gráficas y alfanuméricas
Topográfica	Local	Gráficas
Ortofotografías	WMS	Gráficas
Actividades	Local	Gráficas y alfanuméricas
Padrón habitantes	Local	Gráficas
Disciplina Urb.	Local	Gráficas y alfanuméricas
Patrimonio	Local	Gráficas y alfanuméricas
Zonas verdes y dotaciones	Local	Gráficas
Servicios urbanísticos	Local	Gráficas y alfanuméricas

El tipo de información nos indica donde está físicamente. El tipo ‘local’ está en el servidor de cartografía del propio ayuntamiento, mientras que el marcado con ‘WMS’ establece conexiones a servidores de mapas externos (Oficina Virtual del Catastro o Institut Cartogràfic de Catalunya).
Las capacidades gráficas de los elementos establecen los tipos de consultas que se puede hacer hacia ellos. La información topográfica y ortofotogramétrica únicamente es consultable gráficamente mientras que para el resto es posible consultar además sus atributos alfanuméricos.
A parte de estas consultas crearemos unas herramientas de búsqueda de información gráfica a través de la alfanumérica (en bases de datos diferentes) aprovechando las capacidades de consulta de MapServer. Del estudio de necesidades se desprende que poder localizar objetos geográficos (parcelas por ejemplo) a partir del nombre de propietario o de su calle y número de policía es una herramienta indispensable.
El siguiente esquema resume las relaciones que se establecerán:

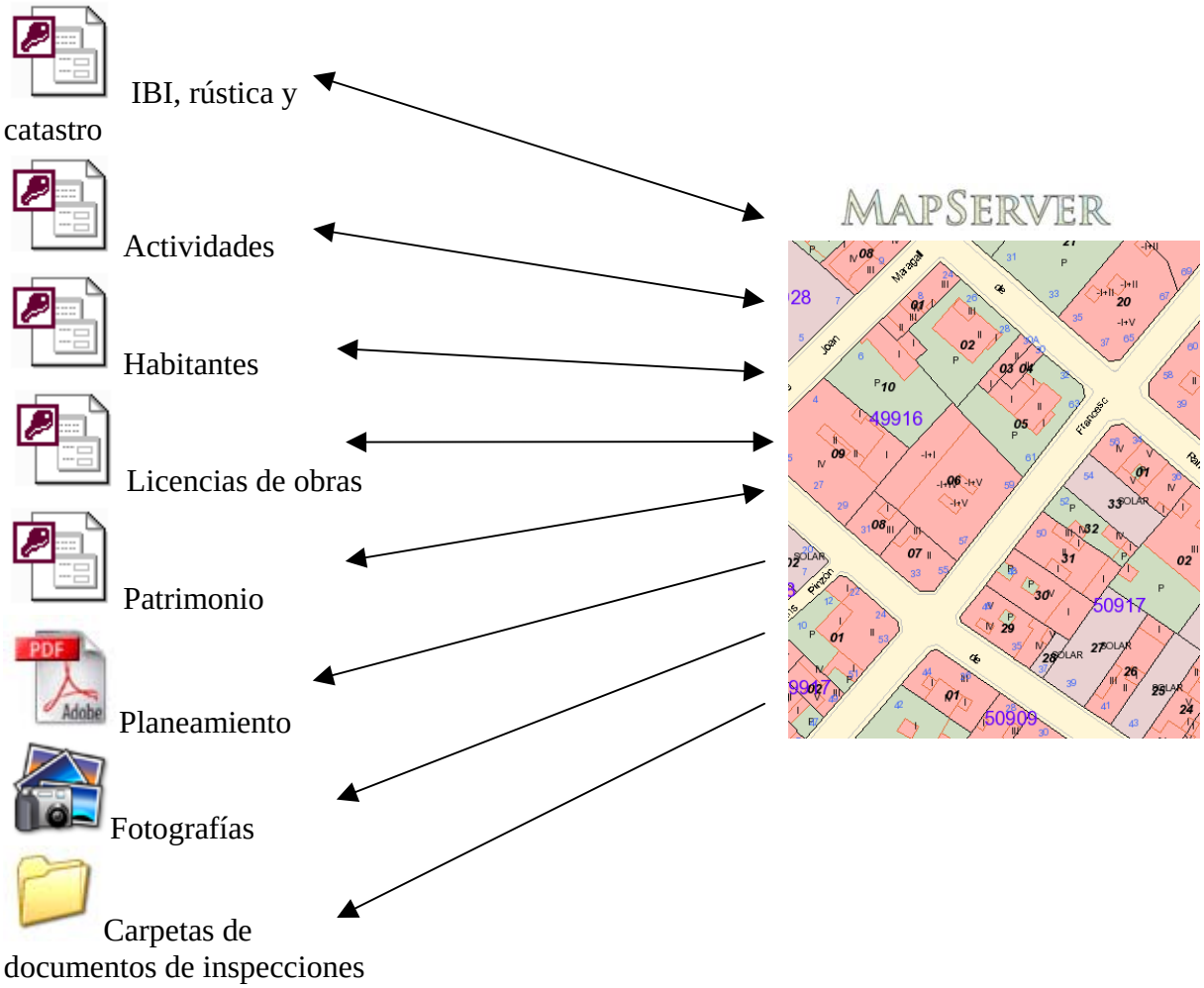


Figura 4.2 Vínculos entre las diferentes bases de datos y la aplicación.

Vemos los dos tipos de relaciones que se establecerán. Por una parte, existen bases de datos como la de IBI, parcelas rústicas y datos referentes a catastro (superficies, usos, valores, etc.) a través de las cuales podemos acceder a su representación gráfica y viceversa. Por otro lado tenemos las relaciones unidireccionales como son las de acceso a las fotografías de las parcelas a través del plano.
Resumiendo, disponemos de información gráfica sin atributos asociados, información gráfica con atributos, información alfanumérica externa relacionada con la gráfica y objetos (documentos, fotografías y carpetas) relacionados también con la gráfica.
Deberemos diseñar una interfaz gráfica de usuario (GUI) para organizar el acceso a toda esta información.

4.2. La interfaz gráfica de usuario (GUI)

Una interfaz de usuario es la parte de una aplicación que el usuario ve y con la cual interactúa. La interfaz incluye las pantallas, ventanas, controles, menús, la ayuda en línea y la documentación.
Nuestra interfaz es una página html y tendrá un aspecto muy similar a la última aplicación creada a modo de ejemplo en el capítulo 4:

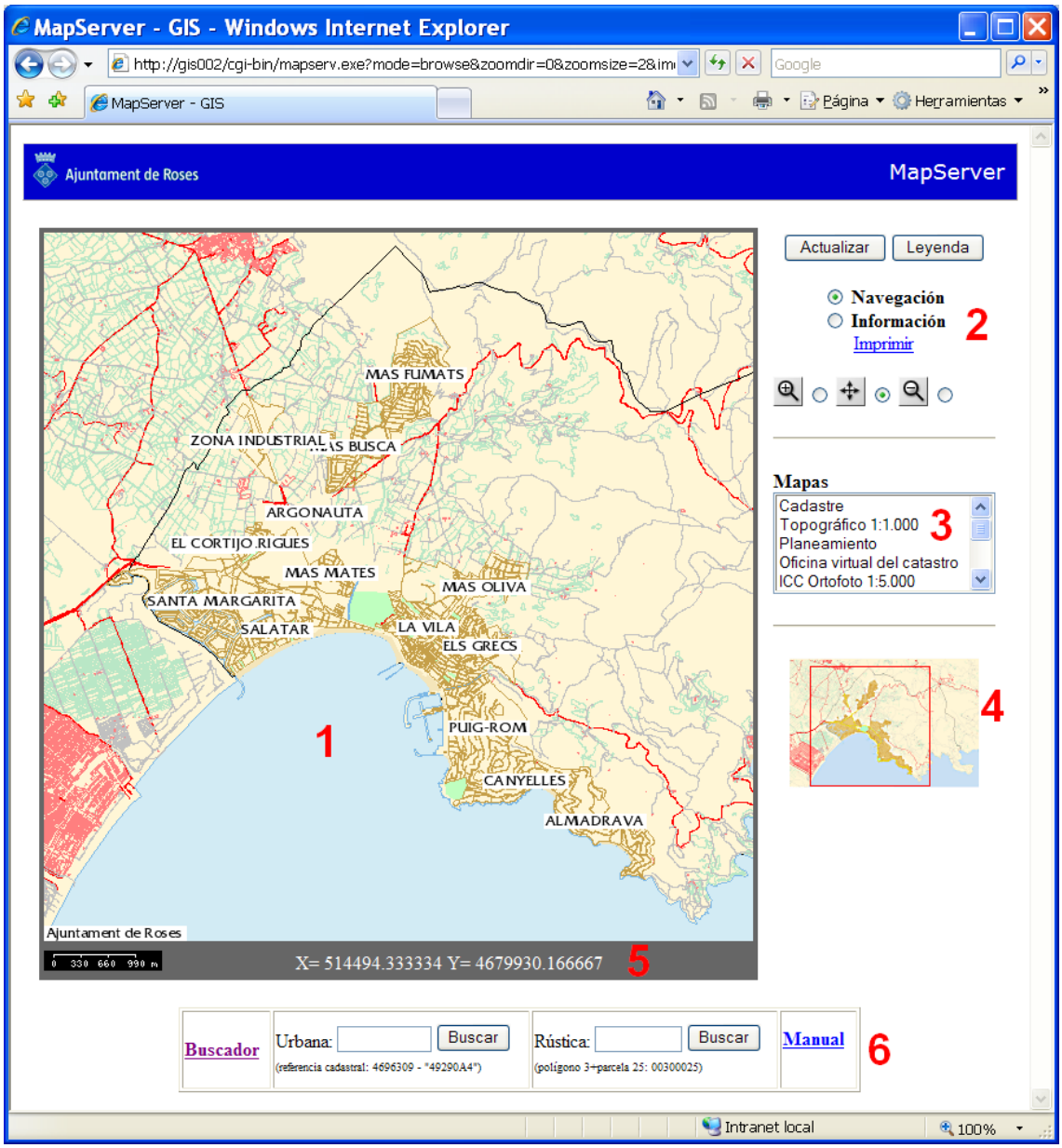


Figura 4.3 La interfaz gráfica de usuario.

Hay seis áreas claramente diferenciadas:

1. Mapa. Cuando se ejecute la aplicación nos muestra una vista general del municipio, señalando las diferentes urbanizaciones.
2. Controles de navegación. A parte de las herramientas básicas de zooms y desplazamiento, de los modos (información y navegación) y del botón de actualizar ya conocidos, se crea otro botón para que nos muestre la leyenda en una ventana aparte y otro que nos inserta el mapa que estamos visualizando en una plantilla para imprimir.
3. Una lista de los mapas disponibles.
4. El mapa de referencia
5. Zona de escala y coordenadas
6. Herramientas de búsqueda y un pequeño manual de la aplicación.

Para acceder a la aplicación, es más elegante hacerlo a través de un acceso directo, no como hasta ahora a través de un fichero de inicio. Todos los datos que enviábamos a MapServer con este fichero de inicio se pueden encadenar en una URL y crear un acceso directo a esta URL:

http://gis002/cgi-bin/mapserv.exe?layer=urb&zoomsize=2&map=%2Fms4w%2Fapps%2Ftfc%2Fguia_tfc.map&program=%2Fcgi-bin%2Fmapserv.exe&root=%2Ftfc%2F&map_web_imagepath=%2Fms4w%2Ftmp%2Fms_tmp%2F&map_web_imageurl=%2Fms_tmp%2F&map_web_template=mapserver.html

Hay que tener en cuenta que es toda una línea de comandos sin espacios y que los símbolos '%2F' representan el carácter '\'.
Este sistema es mucho más cómodo para el usuario y para el responsable del sistema, ya que cuando se quiera añadir alguna funcionalidad más, o cambiar la plantilla inicial, se puede hacer directamente substituyendo la nueva por la vieja en el servidor (manteniendo el nombre). El responsable únicamente debe crear el acceso directo una vez y ponerlo en una carpeta o en una página web a la que todo el personal tenga acceso.

Figura 4.4 Acceso directo a la aplicación.

4.3. Los mapas

Con los ejemplos que hemos descrito hasta ahora, no nos hemos cuestionado cuantas capas (objetos LAYER) es posible tener en un fichero 'mapa', puesto que hemos trabajado con muy pocas capas. En los capítulos anteriores, ya nos empezamos a dar cuenta de que vamos a necesitar muchos objetos LAYER; para el tema 'catastro', por ejemplo, vamos a necesitar las siguientes capas:

Manzanas
Parcelas
Textos parcelas
Subparcelas
Textos subparcelas
Calles
Número de policía
Aceras
Fondo de mapa (tierra)
Fondo de mapa (mar)
Plazas y rotondas
Playas
Límite municipal
Elementos rústica
Polígono de rústica
Parcela de rústica
Construcciones en diseminado
Parcela diseminado
Textos rústica

Que son 19 en total. En principio, la versión precompilada de MapServer está limitada a 100 objetos LAYER, aunque es posible modificar este parámetro cuando se compila.

Si recordamos como trabaja MapServer, nos damos cuenta que cuantas más capas haya, más se tarda en procesar y crear las imágenes, puesto que recorre todo el fichero 'mapa' en busca de información para dibujar.
Para optimizar el rendimiento general y teniendo como condicionante fundamental el uso más efectivo de la aplicación por parte de los usuarios finales, se ha agrupado por temas toda la información. Esta agrupación se ha hecho preguntando a los usuarios que tipo de información y de que manera le es más útil.
Un ayuntamiento es una organización 'limitada' en el sentido que se conocen a todos los posibles usuarios. Tenemos que tener en cuenta que la aplicación que estamos haciendo es para ellos, y se tienen que considerar todos sus comentarios y sugerencias para que la utilicen el mayor número de ellos posible.
No es necesario la mayoría de las veces tener toda la información solapándose. Podemos prever las acciones que siguen los usuarios para consultar la información y nos daremos cuenta que se sigue un patrón bastante exacto, puesto que los posibles usuarios tienen unas necesidades muy similares.
El caso más común es: buscamos una parcela o una característica de esta, como por ejemplo su situación urbanística, su fotografía, si tiene licencia de obras, los propietarios, los habitantes, si hay un local comercial (y cual es), etc.

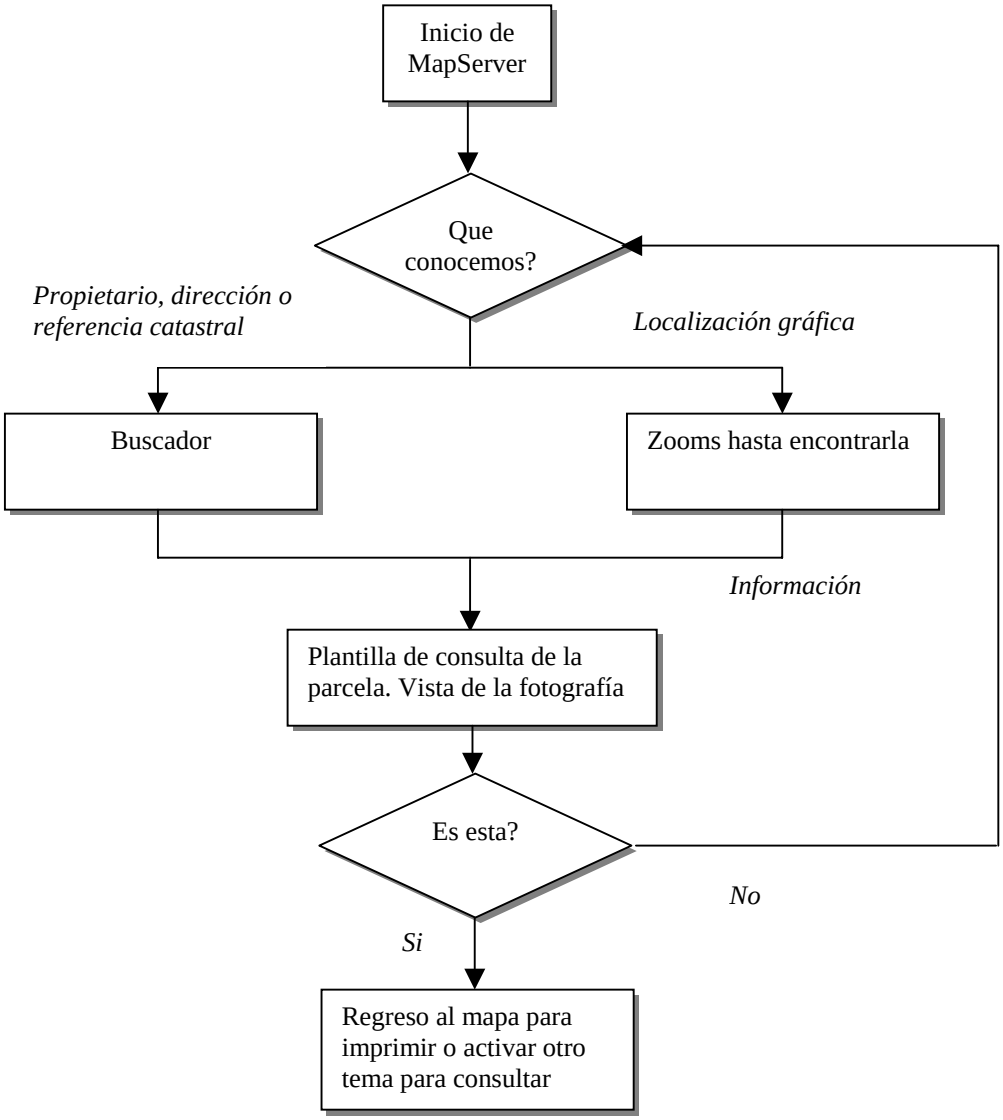


Figura 4.5 Diagrama de flujo de la utilización de la aplicación.

Cuando buscamos información gráfica de cualquier tipo, partimos de alguna otra información que nos ayuda a situar el suceso en el mapa. Por ejemplo si queremos saber las características urbanísticas de una parcela o conocemos la referencia catastral, o la dirección postal o como mínimo el propietario. Usando la herramienta 'Buscador' si conocemos el propietario o dirección postal, o la búsqueda por referencia catastral de la propia aplicación obtendremos la plantilla de consulta de la parcela que cumpla las condiciones que hemos definido. Como en la plantilla, como veremos más adelante, se puede ver la fotografía de la parcela, es muy fácil saber si es la que buscamos. Si lo es consultamos desde la propia plantilla los datos que se nos ofrece (propietarios, superficies, características de los locales o año de construcción) o podemos regresar al mapa (nos mantiene el zoom a la parcela seleccionada) y activar otro tema para seguir consultando.
El otro caso que puede ocurrir es que no sepamos ningún dato, o que simplemente estemos navegando por el mapa sin buscar una parcela en concreto. Cuando queramos, al pinchar en información y en la parcela, iremos a la plantilla de consulta de parcelas para comprobar que es la que queremos, y seguiremos el proceso descrito anteriormente.

Este esquema de trabajo es el que se sigue en la mayoría de las ocasiones. Es lo que en este mismo capítulo hemos denominado consultas simples.
Aclarados estos conceptos, vamos a definir los temas de información (los ficheros 'mapa') que vamos a utilizar en nuestra aplicación.

4.3.1. Catastro

Este tema tendrá todos los elementos catastrales de rústica, urbana y diseminados en un mapa continuo.
Algunos de los elementos gráficos que forman parte de este tema son los que ya hemos utilizado en los ejemplos de funcionamiento de MapServer. Vamos a hacer una relación de todos los ficheros shp (que comprenden los shp, dbf y shx) que utilizamos:

catastro.map

Ficheros shp	Tipo	Descripción
poligons	Polígonos	Urbanizaciones
carrer3	Líneas	Textos de las calles
npol	Puntos	Textos de los números de policía
buit	Polígonos	Polígono de fondo para los espacios vacíos
places	Polígonos	Plazas y rotondas
elemlin	Líneas	Aceras
canals	Polígonos	Canales y ríos
platja	Polígonos	Zonas de playa
limit_terme	Línea	Límite del término municipal
rmasa	Polígonos	Polígonos de rústica
constex	Línea	Texto de las construcciones
constru	Polígonos	Construcciones
masa	Polígonos	Manzanas
parcela	Polígonos	Parcelas
parctex	Líneas	Texto del número de parcela
relemlin	Líneas	Líneas auxiliares de rústica
rparcela_	Polígonos	Parcelas de rústica
rdconstru	Polígonos	Construcciones en diseminado
relemtex	Línea	Textos de rústica
rdparcela	Polígonos	Parcelas en diseminado

En el CD que se entrega con el trabajo, en un anexo, se puede ver el fichero catastro.map completo. En este capítulo únicamente comentaremos lo más relevante.
A todos los temas se les ha añadido una LAYER nueva que hemos denominado 'tema', y lo que hace es crear una etiqueta en la esquina inferior izquierda con el nombre del tema. Esto es muy útil, ya que en todo momento tenemos conocimiento de la información que estamos viendo, y cuando se imprime también.

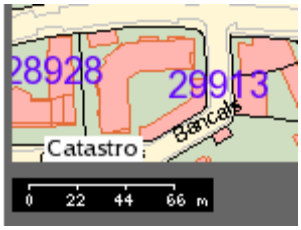


Figura 4.6 Detalle de la esquina del mapa que muestra el nombre del tema.

Esta capa tiene la siguiente sintaxis:

```
63  LAYER
64  NAME tema
65  STATUS default
66  TYPE annotation
67  TRANSFORM false
68  FEATURE
69  POINTS
70  40 593 #Posición del texto en
    coordenadas imagen (píxeles)
71  END
72  TEXT "Catastro"
73  END
74  CLASS
75  LABEL
76  FONT "fritqat"
77  TYPE TRUETYPE
78  SIZE 8
79  BUFFER 1
80  COLOR 0 0 0
81  BACKGROUNDCOLOR 255
    255 255
82  FORCE TRUE
83  END
84  END
85  END
```

Otra característica de este fichero ‘mapa’ es la que vemos en las líneas 243 a 248 del listado completo:

```
243   CLASSITEM "MASA"
244   UNITS METERS
245   LABELITEM "MASA"
246   CLASS
247     NAME "POLIGON"
248     EXPRESSION /^0[0-1]/
```

Es una parte de la LAYER de polígonos de rústica. La línea 243 establece el atributo que se usará para la clasificación y la con la expresión de la línea 248 seleccionará únicamente los polígonos cuyo primer carácter del campo MASA comience por 0 y le siga otro 0 o un 1. Esto se ha resuelto así, ya que existían dentro de este fichero shp otros polígonos (de las zonas urbanas) cuyo atributo MASA comenzaba por ‘09’ y no nos interesa que se dibujen.
De la misma manera, solo se dibujarán las parcelas de rústica que comiencen por ‘00’:

```
427   CLASSITEM "PARCELA"
428   UNITS METERS
429   LABELMAXSCALE 10000
430   LABELITEM "PARCELA"
431   CLASS
432     NAME "RPARCELA"
433     EXPRESSION /^00/
```

Los elementos de este tema que podemos consultar serán las parcelas urbanas, rústicas y en diseminado. El resto de elementos es necesario que se dibujen en el mapa pero generalmente no tiene mucho sentido ni interés acceder a sus datos.
Si pinchamos en nuestra aplicación sobre ‘información’ y luego sobre una parcela, veremos algo similar a:

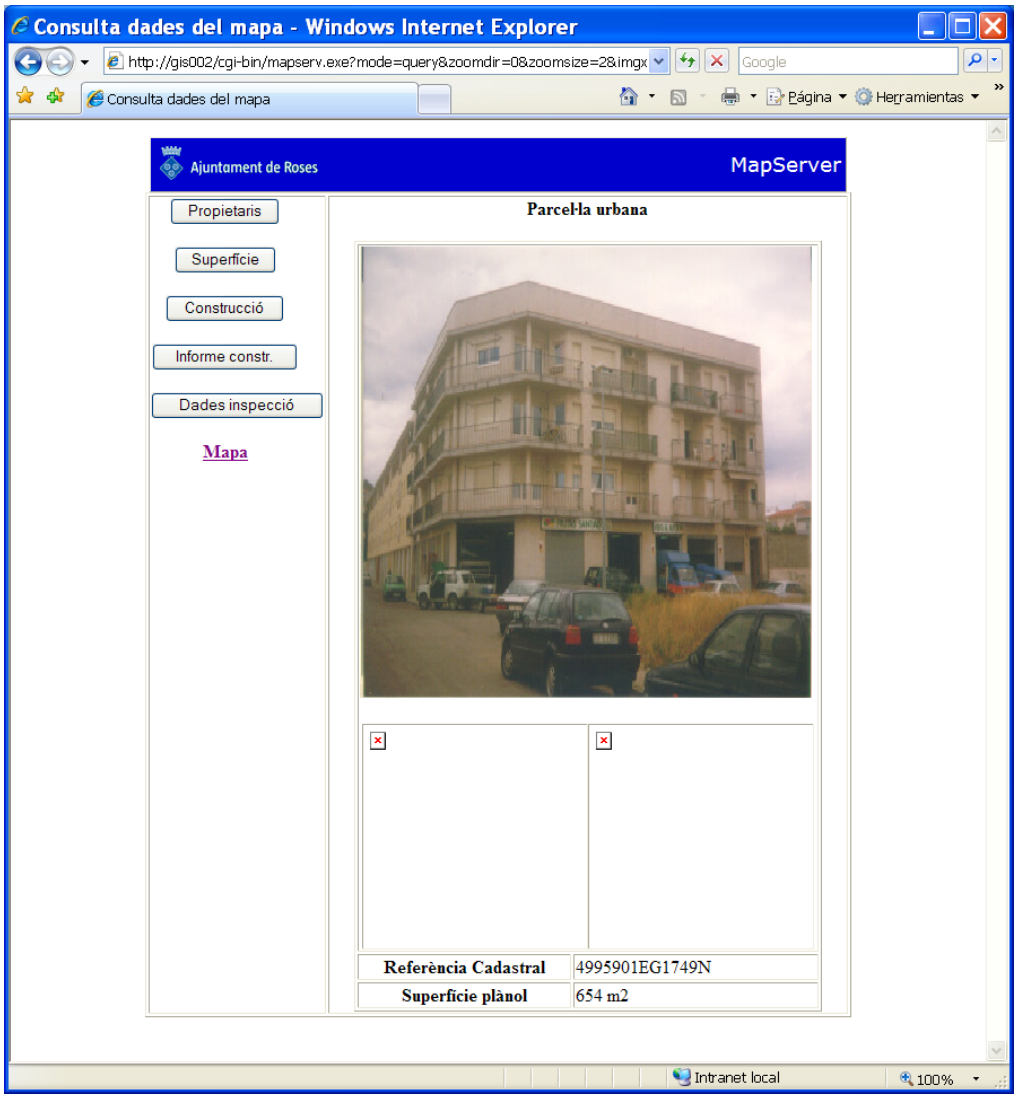


Figura 4.7 Página de información de la parcela urbana.

Los botones de la parte izquierda los comentaremos en el próximo capítulo, pero podemos avanzar que serán consultas a bases de datos, carpetas y documentos.
Bajo estos botones tenemos un link llamado 'Mapa'. Este nos permite volver al mapa en el que estábamos con la misma extensión que teníamos.
En el centro de la imagen vemos una fotografía de la parcela que hemos pinchado y bajo esta dos espacios para imágenes sin insertar.
Disponemos de las fotografías de toda las parcelas en un directorio. El nombre de estas fotografías es su referencia catastral (por ejemplo 4995901.jpg). Alguna de las parcelas tiene más de una fotografía, por ser muy extensa o dar a diferentes calles, así se han nombrado con la referencia seguida de un guión bajo y un número (por ejemplo 4995901_1.jpg).
En la plantilla de consulta se insertan hasta tres fotografías automáticamente de la siguiente manera:

```
58          <th colspan="2"><div align="center">
59            <p><img src="http://GIS002/roses/Fotos_Roses/[UTM].jpg"
   width="400" height="400"> </p>
```

```
60          <p><img src="http://GIS002/roses/Fotos_Roses/[UTM]_1.jpg"
   width="200" height="200">
61                  <img
   src="http://GIS002/roses/Fotos_Roses/[UTM]_2.jpg" width="200"
   height="200"></p>
62          </div>
```

Si ahora vamos a una zona rústica y pedimos información de una parcela, tendremos un resultado diferente al anterior, ya que los datos y las consultas también son diferentes:

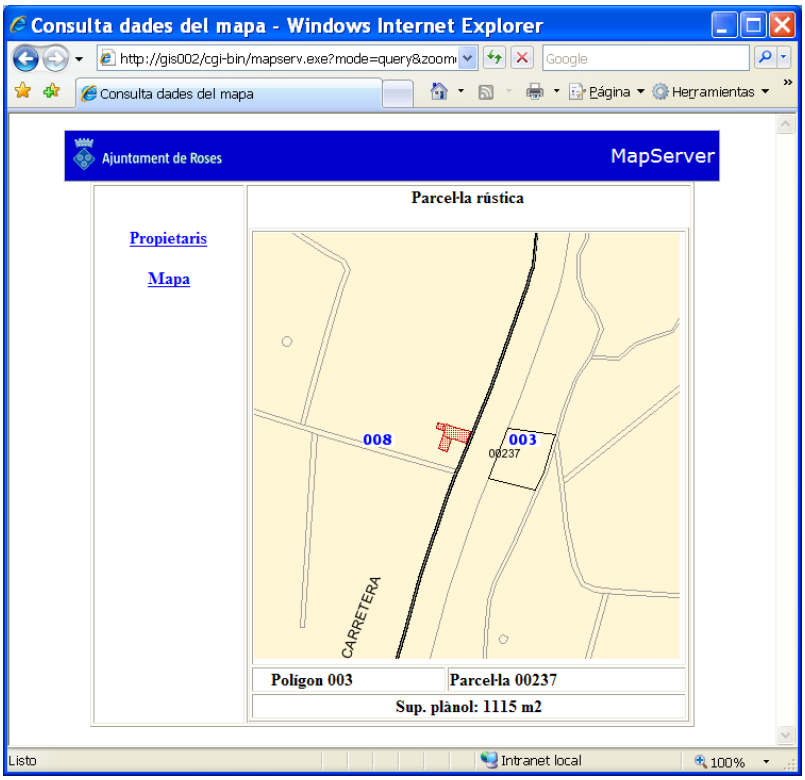


Figura 4.8 Página de información de la parcela rústica.

En la parte izquierda tenemos dos links, una hacia los propietarios de la parcela y el otro hacia el mapa anterior. En la parte del medio tenemos un mapa con la parcela seleccionada y sus datos.
Si ahora pedimos información de una construcción en diseminado obtenemos el resultado en una plantilla muy similar a la de parcelas urbanas, pero en lugar de fotografía insertamos el plano de situación:

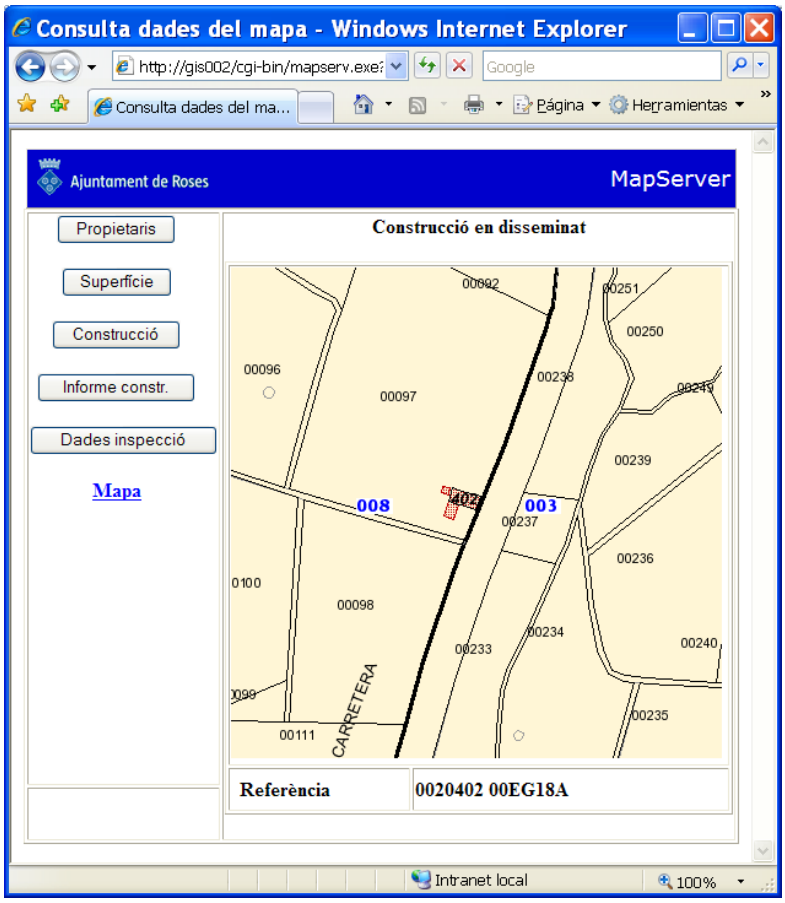


Figura 4.9 Página de información de la construcción en diseminado.

4.3.2. Urbanismo

La información más útil y solicitada de este campo es la zonificación y sus características. Puesto que necesitaremos una referencia para orientarnos pondremos los planos topográficos de fondo en un color neutro (gris). Los ficheros shp que utilizaremos son:

urbanismo.map

Ficheros shp	Tipo	Descripción
Topoline5000	Líneas	Topográfico 1:5000 de la zona rústica
lavilaline	Líneas	Topográfico 1:1000 de la zona 'la vila'
Buscafum line	Líneas	Topográfico 1:1000 de la zona 'busca-fumats'
Stamarga line	Líneas	Topográfico 1:1000 de la zona 'sta margarita'
Puigrom line	Líneas	Topográfico 1:1000 de la zona 'puigrom'
servituds	Líneas	Servidumbres
pe_annex	Líneas	Planeamiento derivado
unitats_actuació	Polígonos	Unidades de actuación
text_ua	Puntos	Textos de las unidades de actuación
text_pe_ed	Puntos	Textos del planeamiento derivado
text_servituds	Puntos	Textos de las servidumbres
pn_line	Líneas	Límite parque natural
limits	Líneas	Límites del suelo urbano, urbanizable y municipal
areas2	Polígonos	Zonas del Plan General

El resultado será:

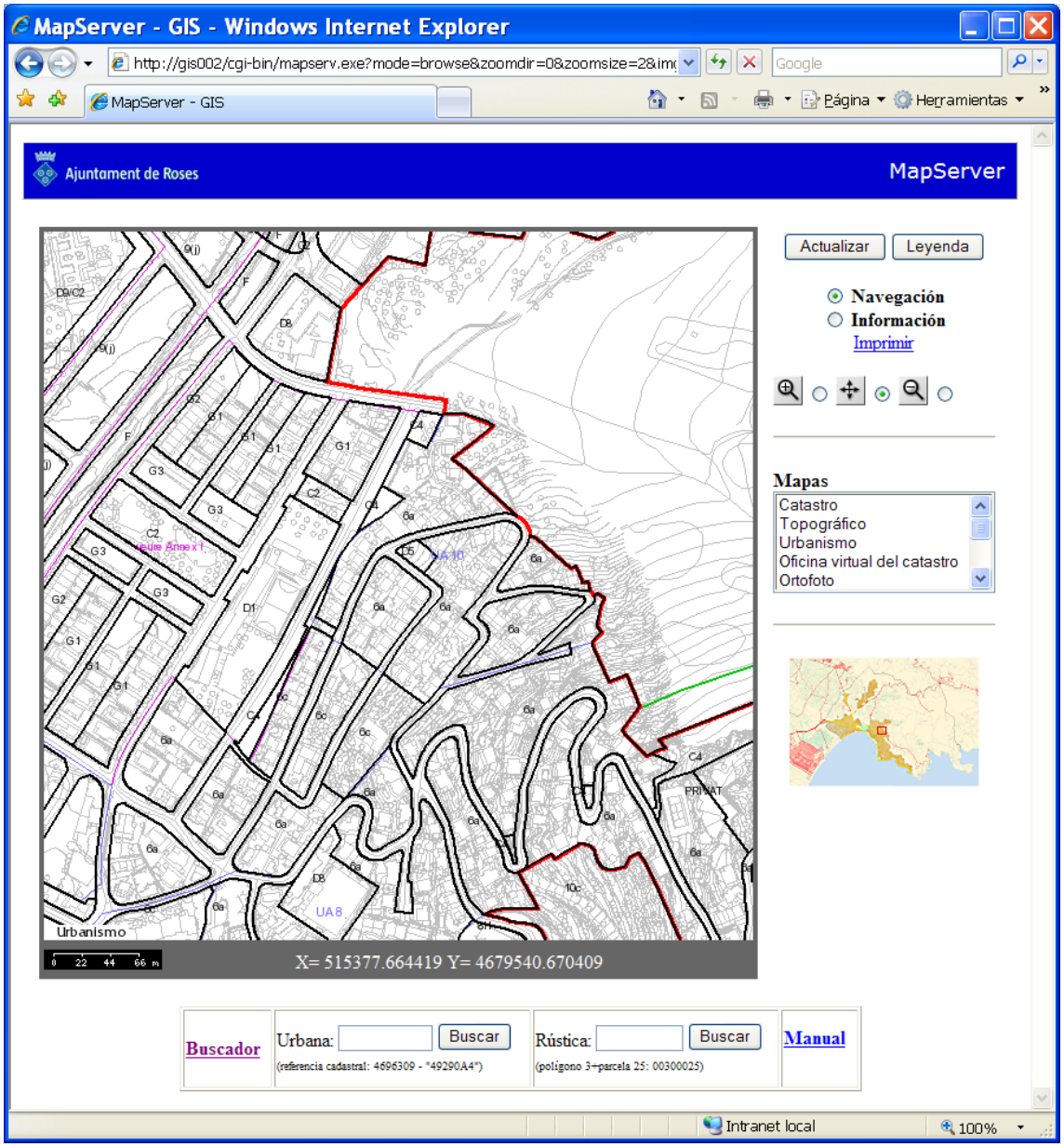


Figura 4.10 Vista del tema urbanismo.

Se ha optado en poner algunos de los textos (servidumbres, unidades de actuación y planeamiento derivado) mediante puntos. Así se tiene un mayor control de la situación de las etiquetas.
Si pedimos información de una zona:

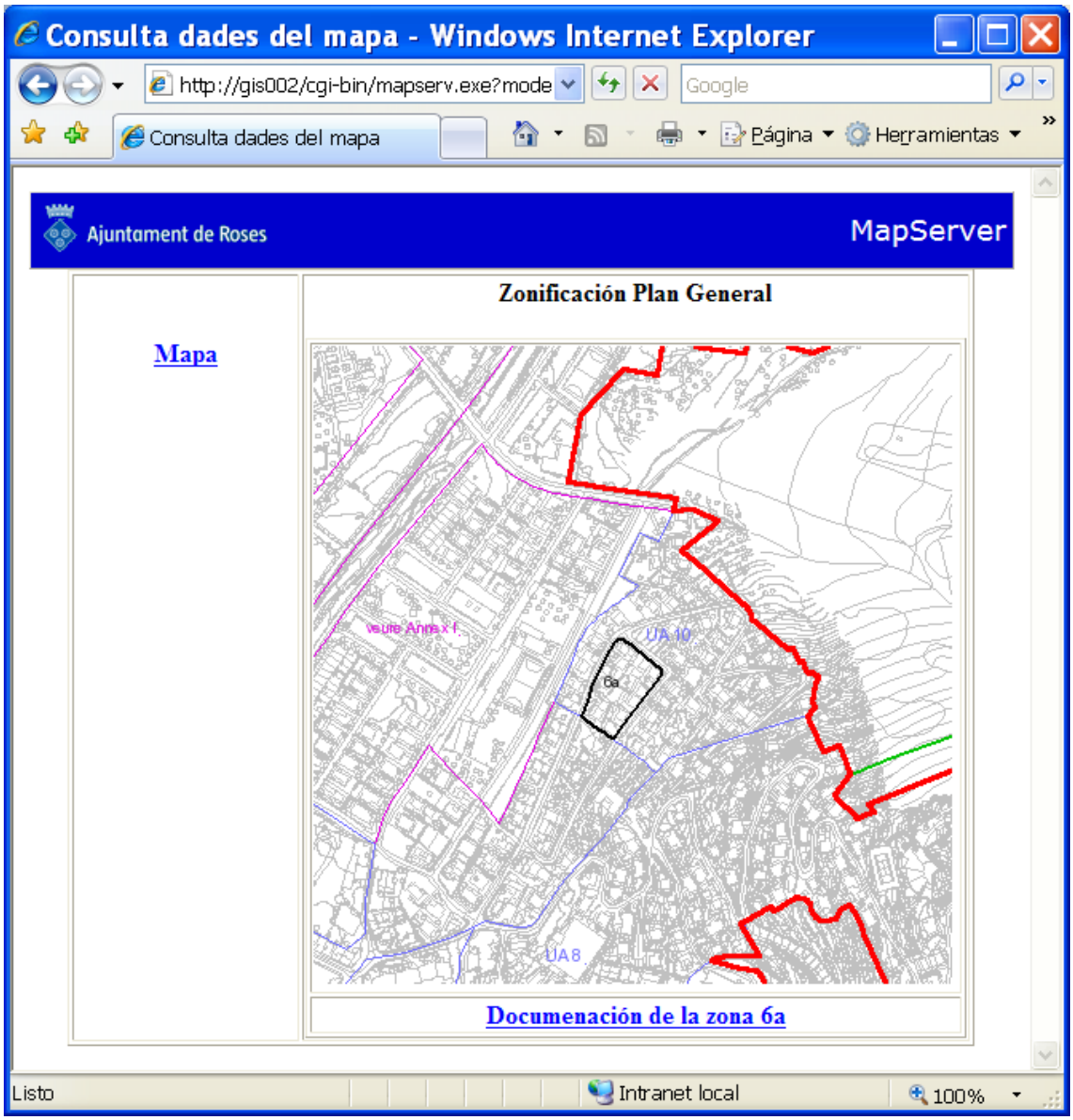


Figura 4.11 Página de información urbanística.

Nos muestra un plano de situación de la zona en cuestión y un link a la documentación asociada. En este caso es un documento pdf.
Si observamos la parte del código de la plantilla de consulta del Plan General que relaciona la zona con el pdf veremos:

Documenación de la zona [Attribute1]

en la carpeta tfc/datos/pgou/ tenemos documentos pdf asociados a cada zona. A estos documentos accedemos creando la ruta por una parte textual y por la otra mediante substitución de atributos.
Si ahora pinchamos en el link:

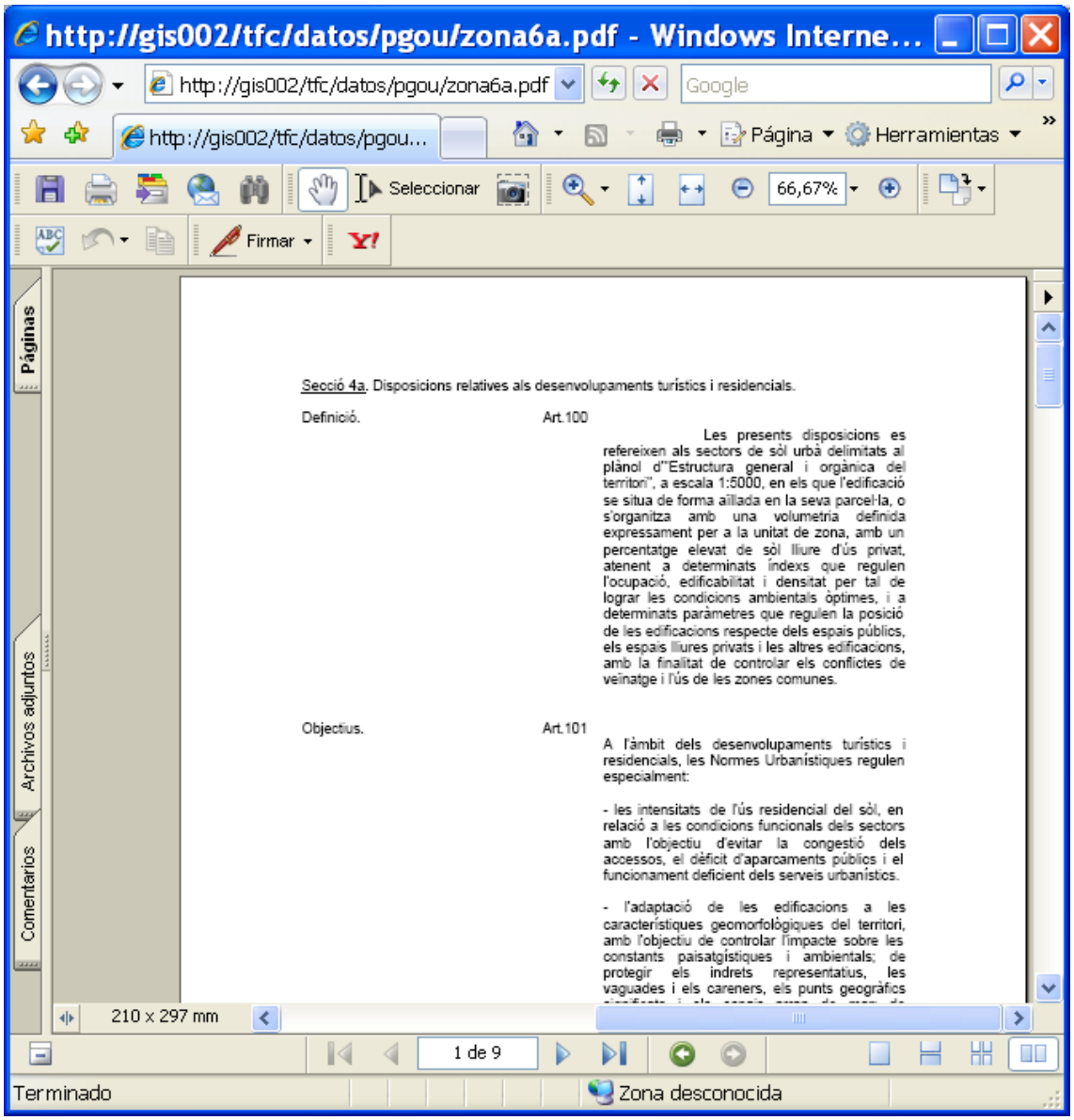


Figura 4.12 Documento con información urbanística de la zona seleccionada.

4.3.3. Topográficos

En este caso la información principal será la topográfica. Aprovecharemos la capacidad de MapServer de leer ficheros dgn directamente y asignar a cada elemento su color original en el dgn. Disponemos de los siguientes:

topografico.map

Ficheros dgn	Tipo	Descripción
Buscafumats_v7	Líneas	Topográfico 1:1000 de la zona 'busca-fumats'
Lavila_v7	Líneas	Topográfico 1:1000 de la zona 'la vila'
Puigrom_v7	Líneas	Topográfico 1:1000 de la zona 'puigrom'
Stamarga_v7	Líneas	Topográfico 1:1000 de la zona 'sta margarita'
5000x1000_v7	Líneas	Topográfico 1:5000 de la zona no urbana
5000_v7	Líneas	Topográfico 1:5000 de todo el municipio

Un ejemplo de la sintaxis de las capas en el fichero ‘mapa’ es el siguiente:

```
87  LAYER
88  NAME dgn
89  TYPE LINE
90  STATUS default
91  CONNECTIONTYPE OGR
92  CONNECTION "buscafumats_V7.dgn,0"
93  STYLEITEM "AUTO"
94  maxscale 3000
95  CLASS
96  END
97  END
```

Es en la línea 93 donde indicamos a MapServer que mantenga el color original de los objetos.
Otra característica de este tema es la continuidad de la información. Con un denominador de escala superior a 3000 nos mostrará solo el fichero ‘5000_v7.dgn’. Es decir, toda la información será la propia de la escala 5000:

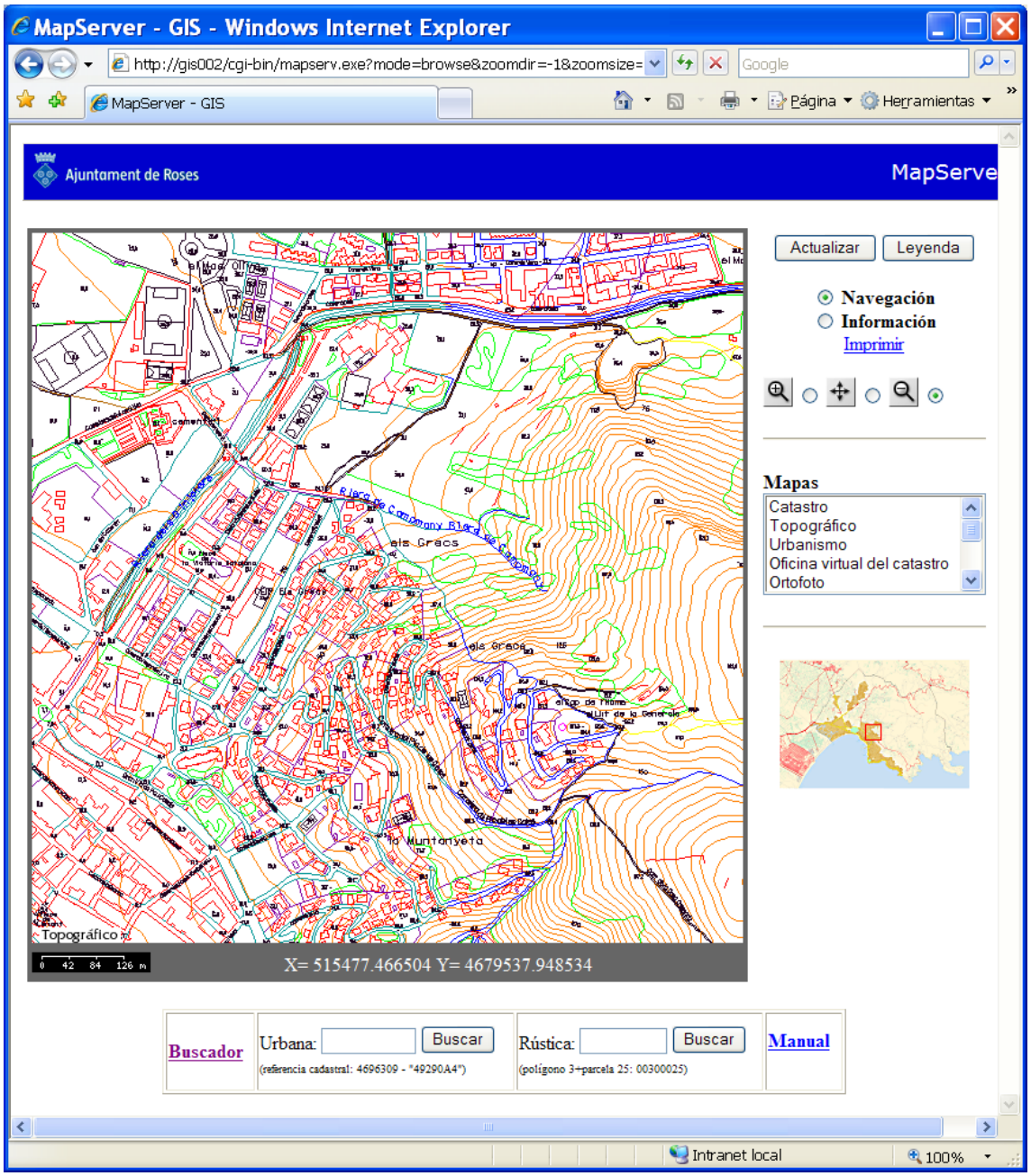


Figura 4.13 Vista del tema topografía.

Pero si bajamos el denominador a valores menores de 3000, en las áreas urbanas mostrará información de los topográficos 1:1000 y en las áreas no urbanas continuará mostrando la información propia de 1:5000. Aunque sea cartografía a escalas diferentes, aprovechamos la sensación de continuidad.

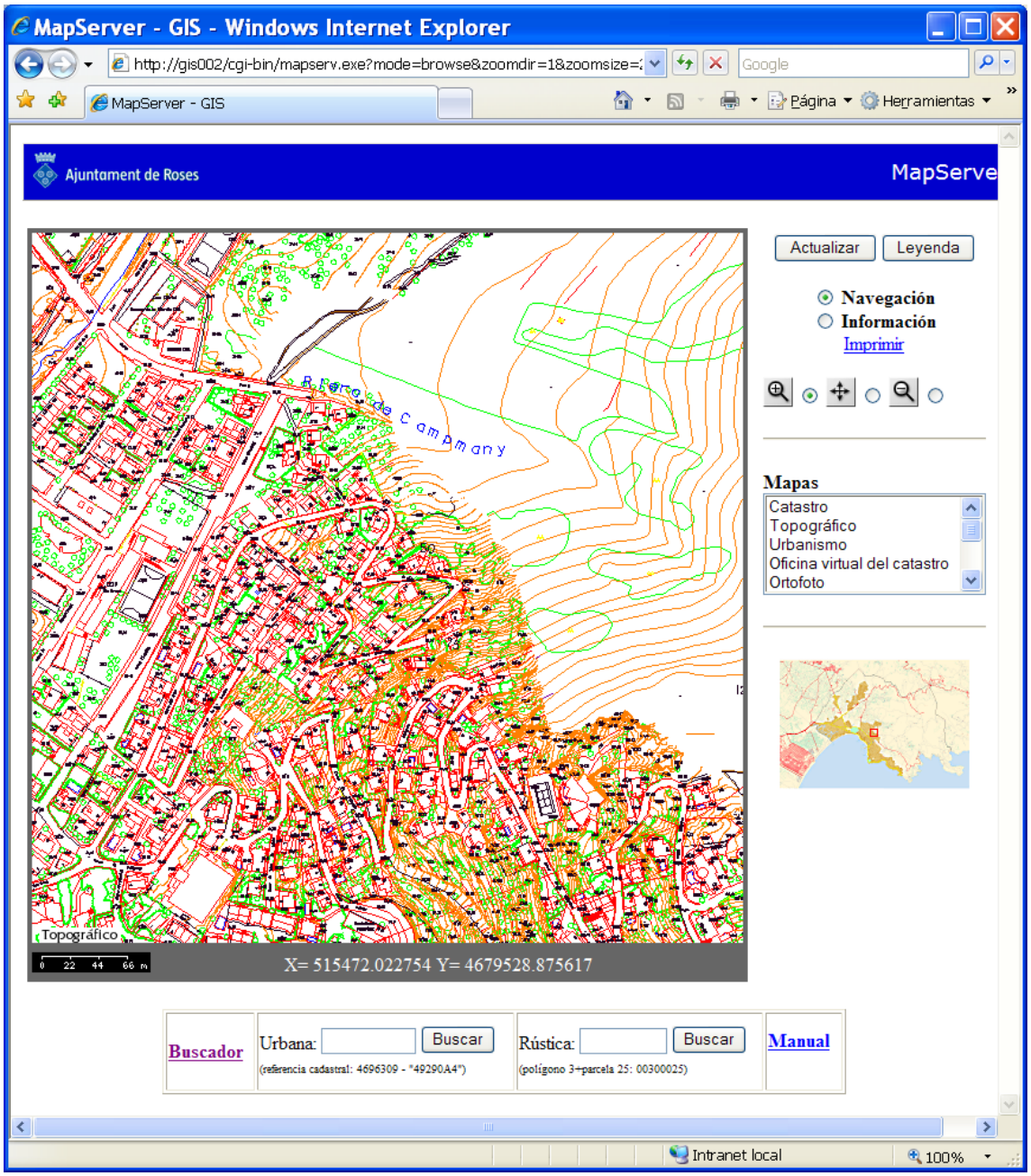


Figura 4.14 Vista del tema topografía uniendo datos gráficos a escalas 1:1000 y 1:5000 .

Hay que señalar que de momento MapServer puede leer únicamente ficheros dgn de la versión 7 (o J).

4.3.4. Actividades

Este tema comprende las actividades comerciales del municipio. Es un inventario exhaustivo de todo local comercial o actividad económica que se utiliza para controlar las licencias, la adecuación a las normativas vigentes según el tipo de actividad y para el cobro de una tasa especial de recogida de basuras entre otras. Se toma como base el tema catastro, y se insertan los símbolos de las actividades.

actividades.map

Ficheros shp	Tipo	Descripción
Los del tema catastro actividades	puntos	Actividades comerciales

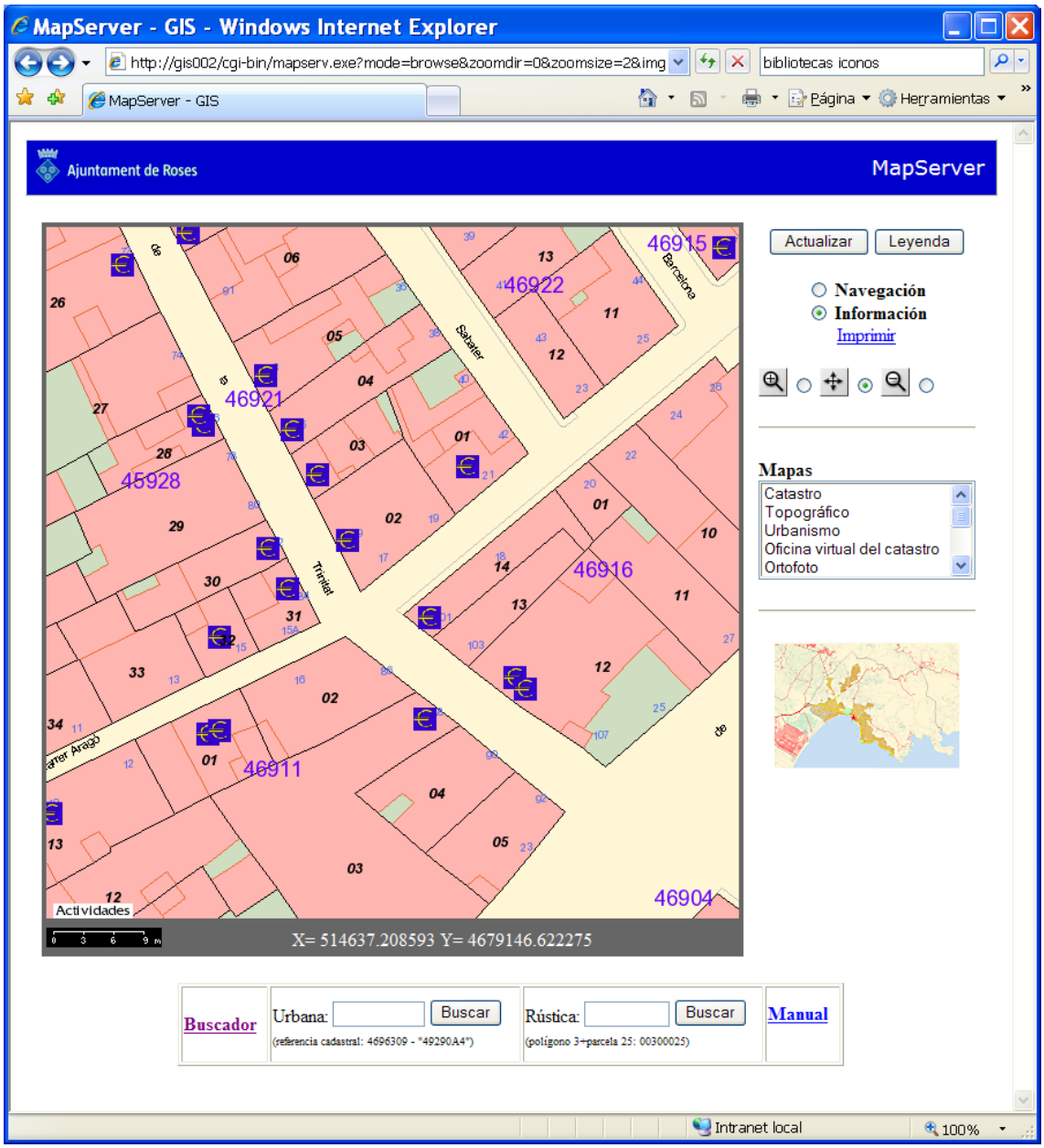


Figura 4.15 Vista del tema actividades económicas .

El objeto LAYER es el siguiente:

```
98   LAYER
99     NAME "activitats"
100   STATUS default
101   DATA "activitats"
102   TYPE point
103   TOLERANCE 10
104   MAXSCALE 4000
```

```
105   CLASS
106     TEMPLATE "consulta_activitats.htm"
107     STYLE
108       SYMBOL "euro"
109       SIZE 20
110       COLOR 255 0 0
111     END
112   END
113 END
```

Un detalle a tener en cuenta es la relación entre la tolerancia (línea 502) y el tamaño del símbolo (línea 508). La tolerancia es la distancia alrededor del punto (de un píxel de tamaño) dentro de la cual se puede seleccionar el objeto. Esto nos limita a 10 píxeles alrededor del punto. Como el símbolo tiene un tamaño de 20 píxeles, prácticamente todo el símbolo será consultable. Se ha ajustado tanto ya que pueden existir dos símbolos muy juntos, y de esta manera ganamos precisión a la hora de seleccionar.
Del mismo modo que en los anteriores ejemplos, si pinchamos en información y luego en una actividad obtenemos:

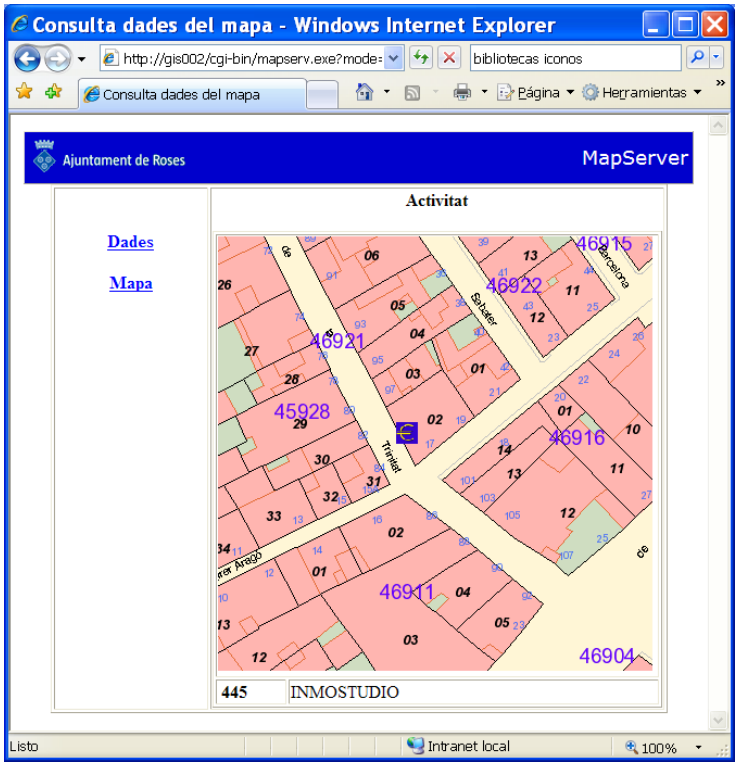


Figura 4.16 Página de información de actividades económicas .

Como resultado obtenemos un plano de situación, y los datos de código de la actividad y el nombre del negocio. Veremos en el próximo capítulo la consulta a la base de datos a través del link ‘Dades’.

4.3.5. Padrón de habitantes

En estos momentos se dispone de la división del municipio en distritos, secciones y manzanas poblacionales (que no se corresponden exactamente con las catastrales, pero se está trabajando para que concuerden físicamente con las catastrales).
Así mismo se está realizando una revisión de las calles y números de policía para obtener un callejero completo y actualizado. El objetivo último es tener una codificación de los edificios, y de cada elemento (o local) de los edificios, para así tener un núcleo que forme la base territorial del que se sirvan todas las aplicaciones y bases de datos.
Del mismo modo que el tema de actividades, se utiliza como mapa de referencia el catastral, pero eliminando las etiquetas catastrales y los colores. Los ficheros que forman este tema son:

seccionado.map

Ficheros shp	Tipo	Descripción
Los del tema catastro		
seccio	Polígonos	Las diferentes secciones del municipio
districte	Polígonos	Los distritos censales
Illa	Polígonos	Las manzanas censales

Las consultas de la información del padrón de habitantes las consideraremos complejas hasta que resolvamos el tema de la codificación para relacionar cada edificio con el mapa. Por lo tanto, los estudios se están haciendo (a nivel de manzana, no de edificio) utilizando Geomedia.
La siguiente imagen muestra una vista de la información en el mapa y los datos de la consulta de las manzanas censales son solo el distrito, la sección y la manzana.

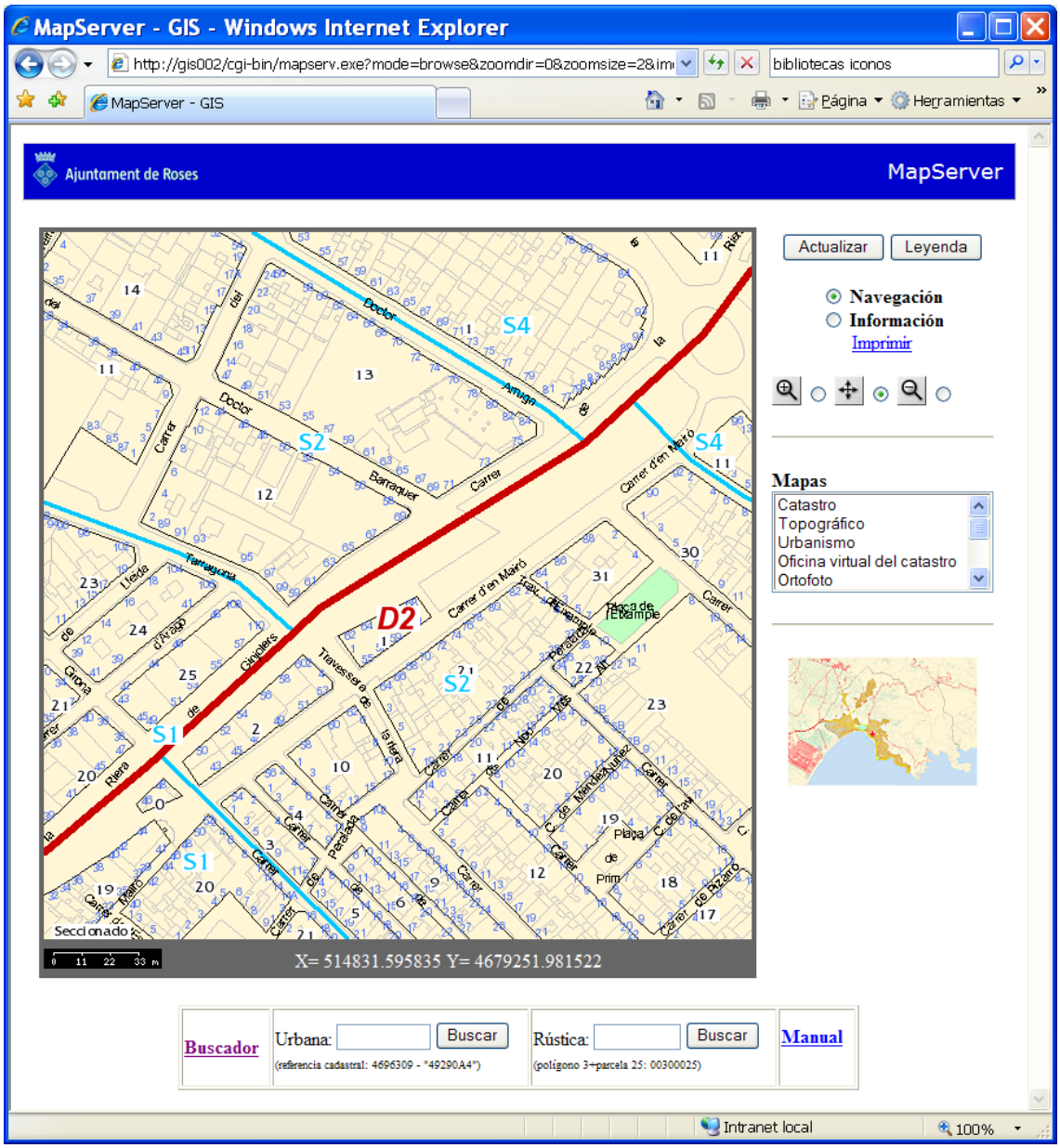


Figura 4.17 Vista del tema padrón de habitantes.

4.3.6. Disciplina urbanística

Agrupa la información referente a las licencias de obras y su gestión. Para solicitar la licencia de obras es necesario entregar un plano de situación, que se genera con MapServer utilizando el tema catastro. Estos mapas de situación los elabora cualquier persona del ayuntamiento, pero generalmente el personal del servicio de atención ciudadana (SAC). En este tema se representan mediante puntos los lugares donde se ha solicitado una licencia, y de diferente color según el estado de la licencia.
Se han creado un símbolo diferente para cada estado de la licencia, en el fichero 'mapa' se les ha asignado uno a cada obra según el valor del atributo 'Codi'.

Figura 4.18 Símbolos para el tema de control de obras.

Y la aplicación tiene este aspecto:

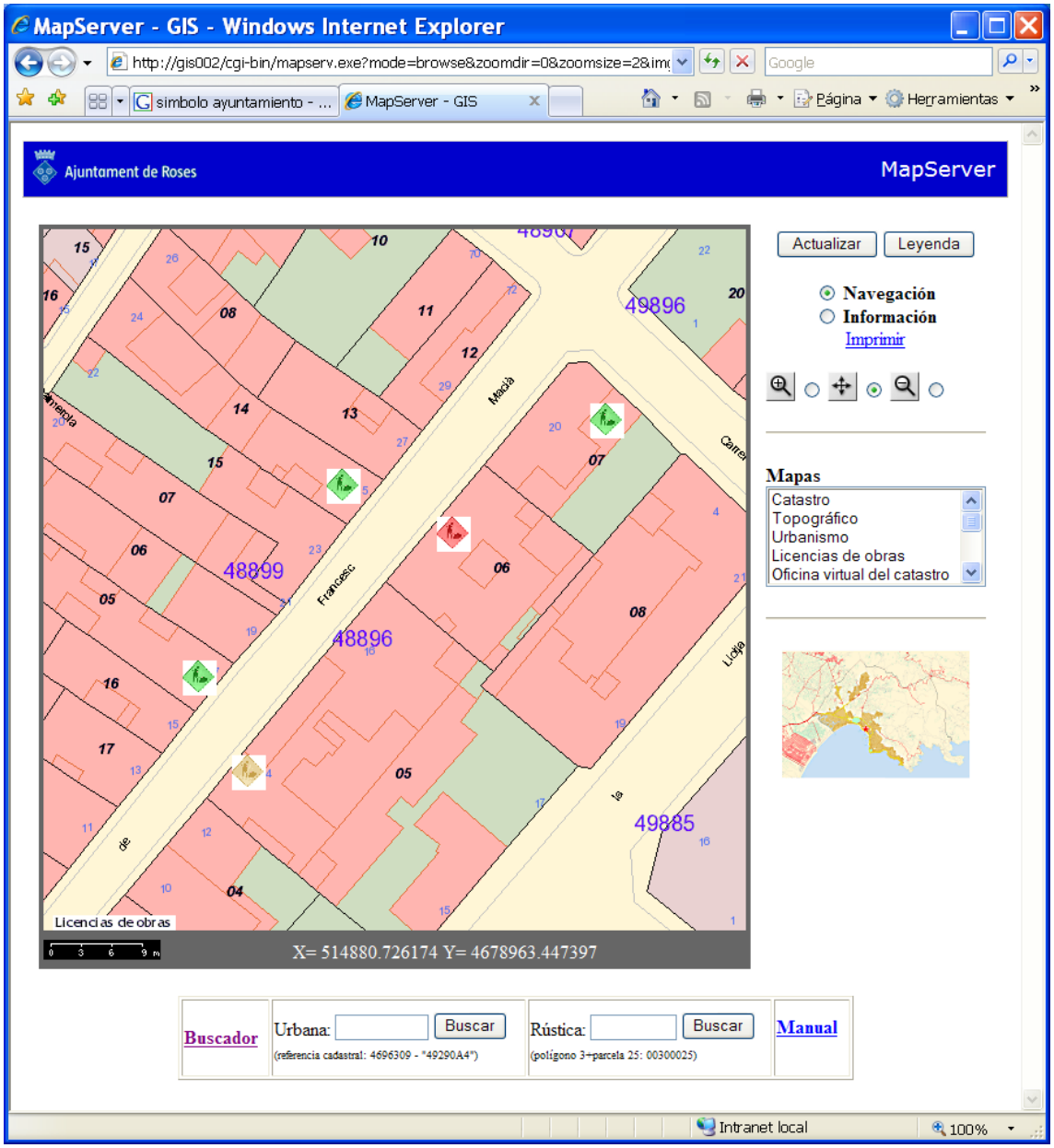


Figura 4.19 Vista del tema disciplina urbanística.

El fichero ‘mapa’ contiene los siguientes elementos:

obras.map

Ficheros shp	Tipo	Descripción
Los del tema catastro		
obras	puntos	Licencias de obras

Si pedimos información de una obra cualquiera:

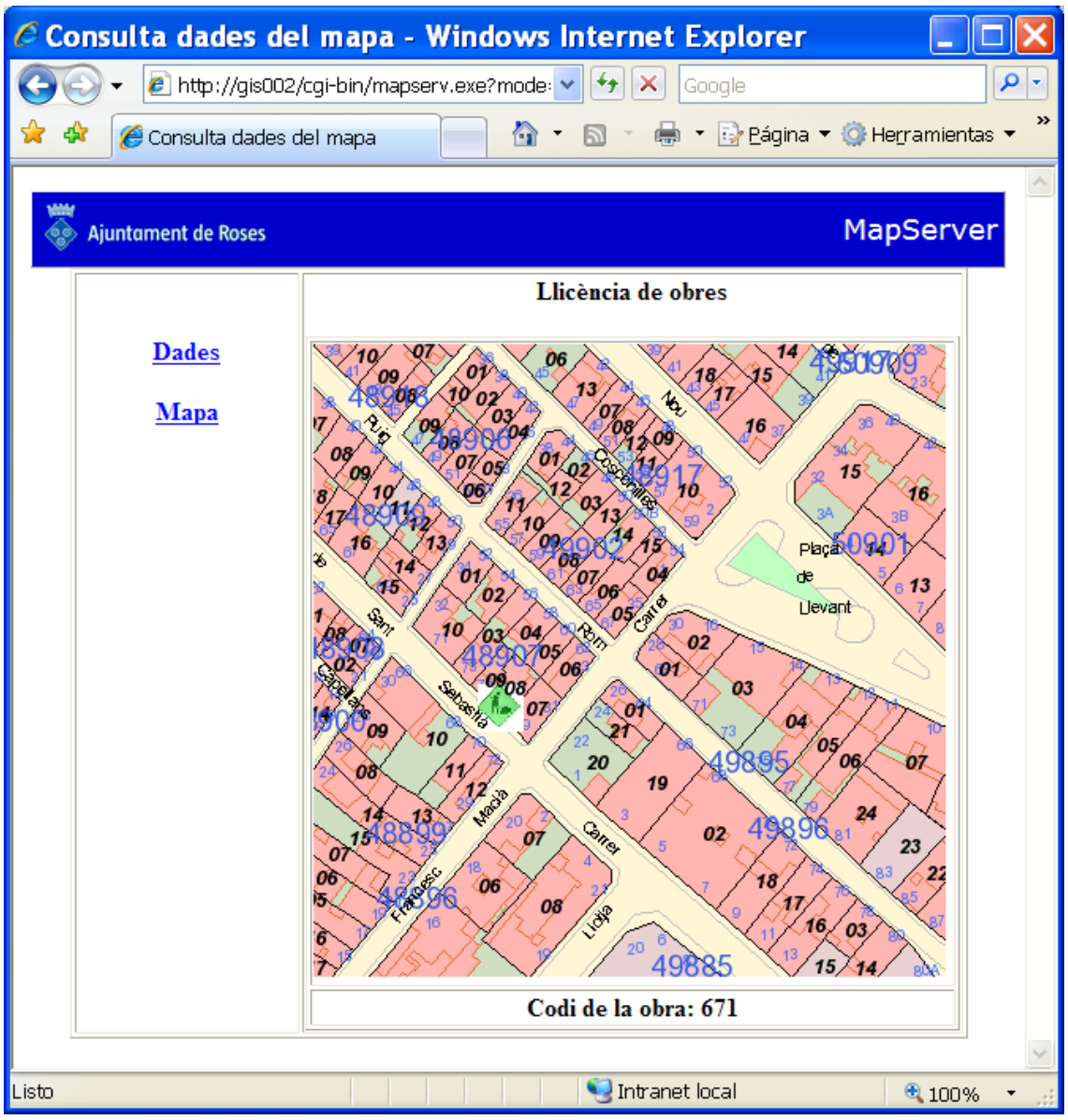


Figura 4.20 Página de información de una licencia de obra.

Solo obtenemos el código de la obra, y como veremos en el siguiente capítulo, con el link ‘dades’ accederemos a la base de datos de licencias de obras.

4.3.7. Patrimonio

En este tema recogemos todos los elementos que forman parte del patrimonio del ayuntamiento. Forman parte de este conjunto los elementos como edificios, plazas, calles, etc. Están representados por elementos puntuales:

bens.map

Ficheros shp	Tipo	Descripción
Los del tema catastro		
Bens_inmobles_total	puntos	Inventario del patrimonio

También se ha creado un símbolo para estos elementos:

Figura 4.21 Imagen utilizada para el tema patrimonio.

Y el resultado es:

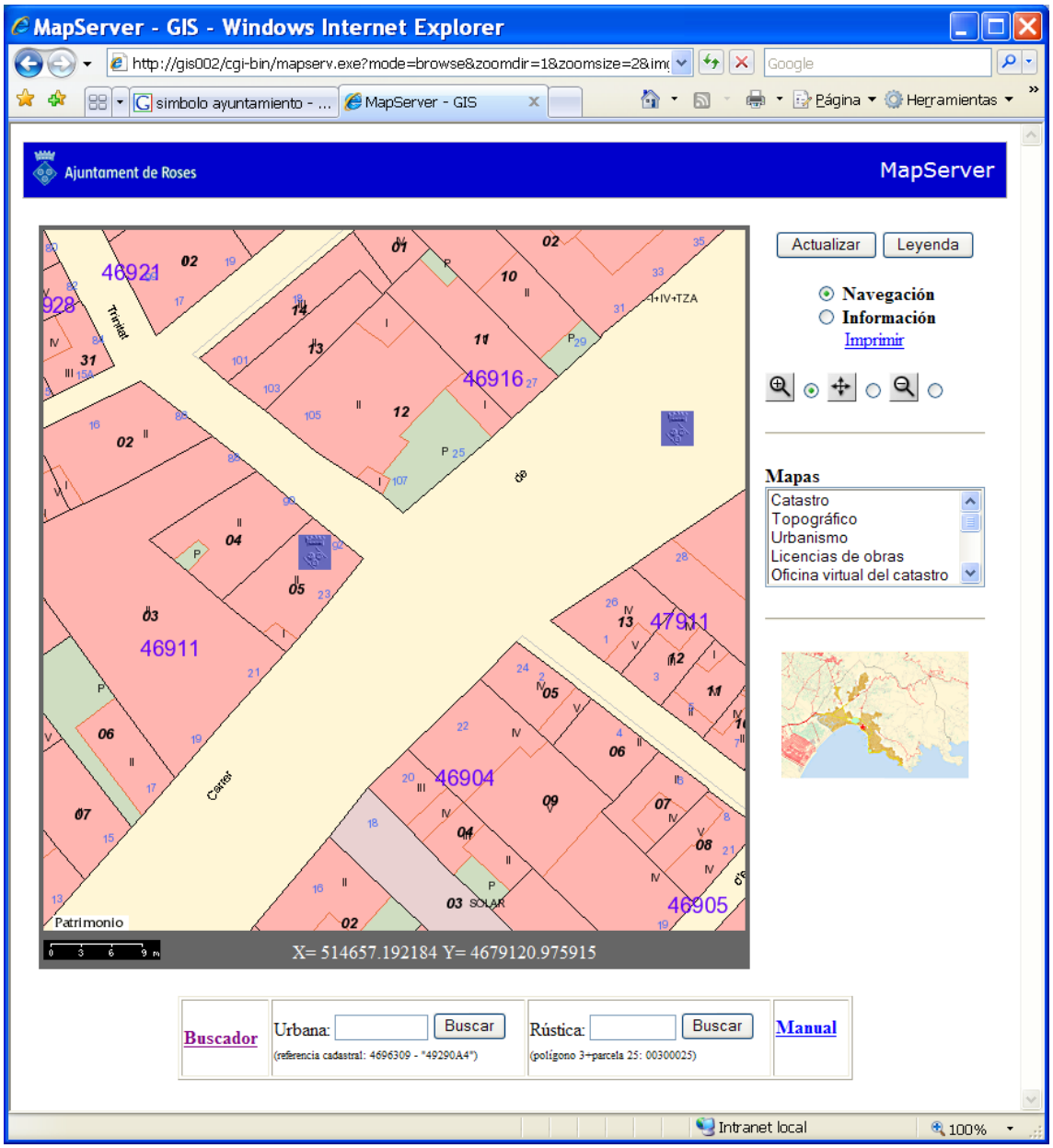


Figura 4.22 Vista del tema patrimonio.

Dado que este inventario se ha hecho utilizando ArcView, tendremos accesibles todos los datos en el mismo la misma plantilla de consulta. Si pedimos información tendremos:

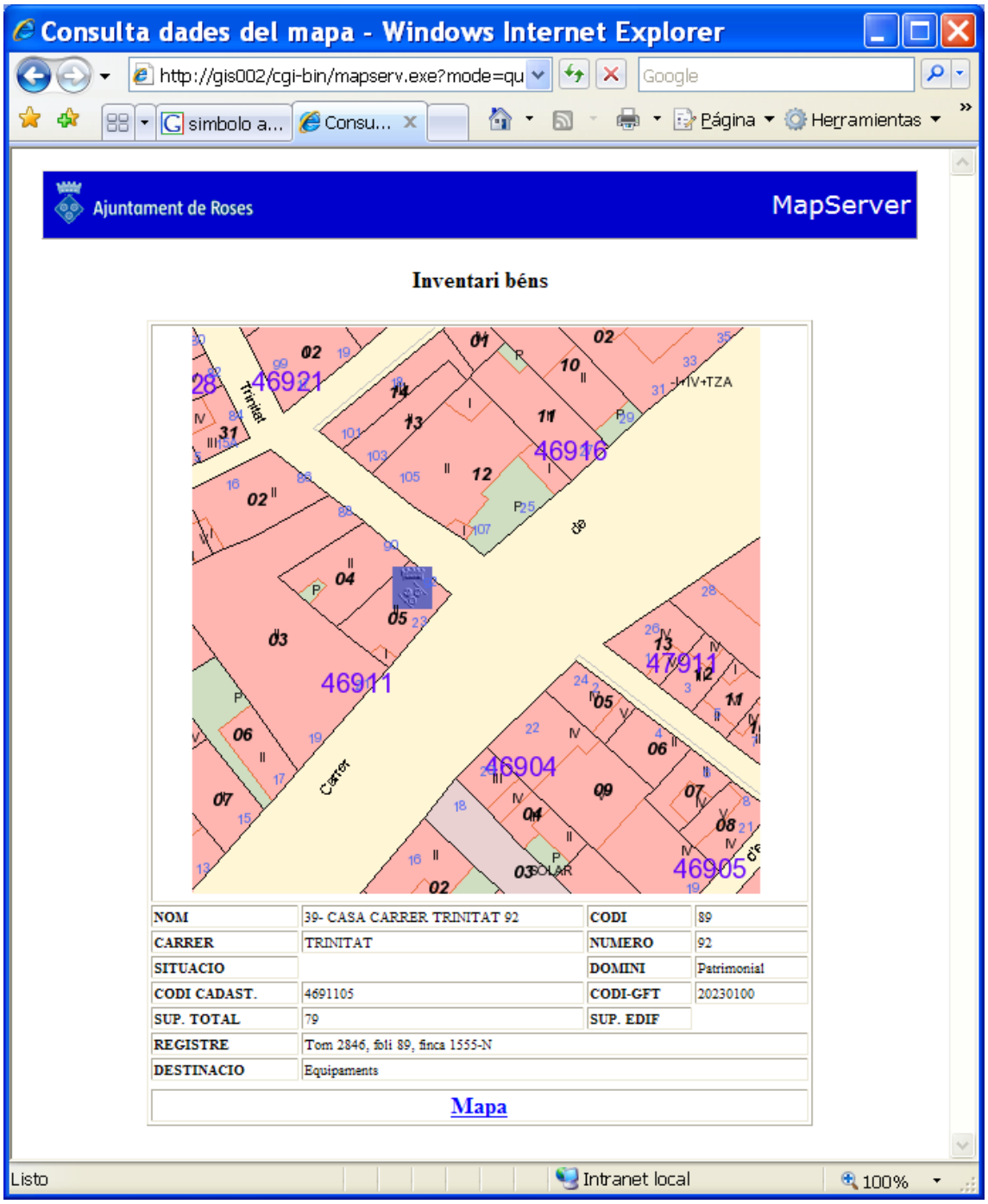


Figura 4.23 Página de información de un elemento del tema patrimonio.

4.3.8. Zonas verdes y dotaciones

Este tema forma parte del plan general, pero por su importancia se ha creado uno específico. Veremos que los ficheros shp son muy similares a los de urbanismo, y se ha añadido uno con la información de zonas verdes y dotaciones. Los elementos no tienen atributos asociados. Es simplemente un plano informativo.

zonasverdes.map

Ficheros shp	Tipo	Descripción
espais_lliures_dotacions	Polígonos	Zonas verdes y dotaciones
Topoline5000	Líneas	Topográfico 1:5000 de la zona rústica
lavilaline	Líneas	Topográfico 1:1000 de la zona 'la vila'
Buscafum line	Líneas	Topográfico 1:1000 de la zona 'busca-fumats'
Stamarga line	Líneas	Topográfico 1:1000 de la zona 'sta margarita'
Puigrom line	Líneas	Topográfico 1:1000 de la zona 'puigrom'
limits	Líneas	Límites del suelo urbano, urbanizable y municipal
Pla general	Líneas	Zonificación del plan general
Pgou_text	Puntos	Textos de las zonificaciones

El tema tiene este aspecto en la aplicación:

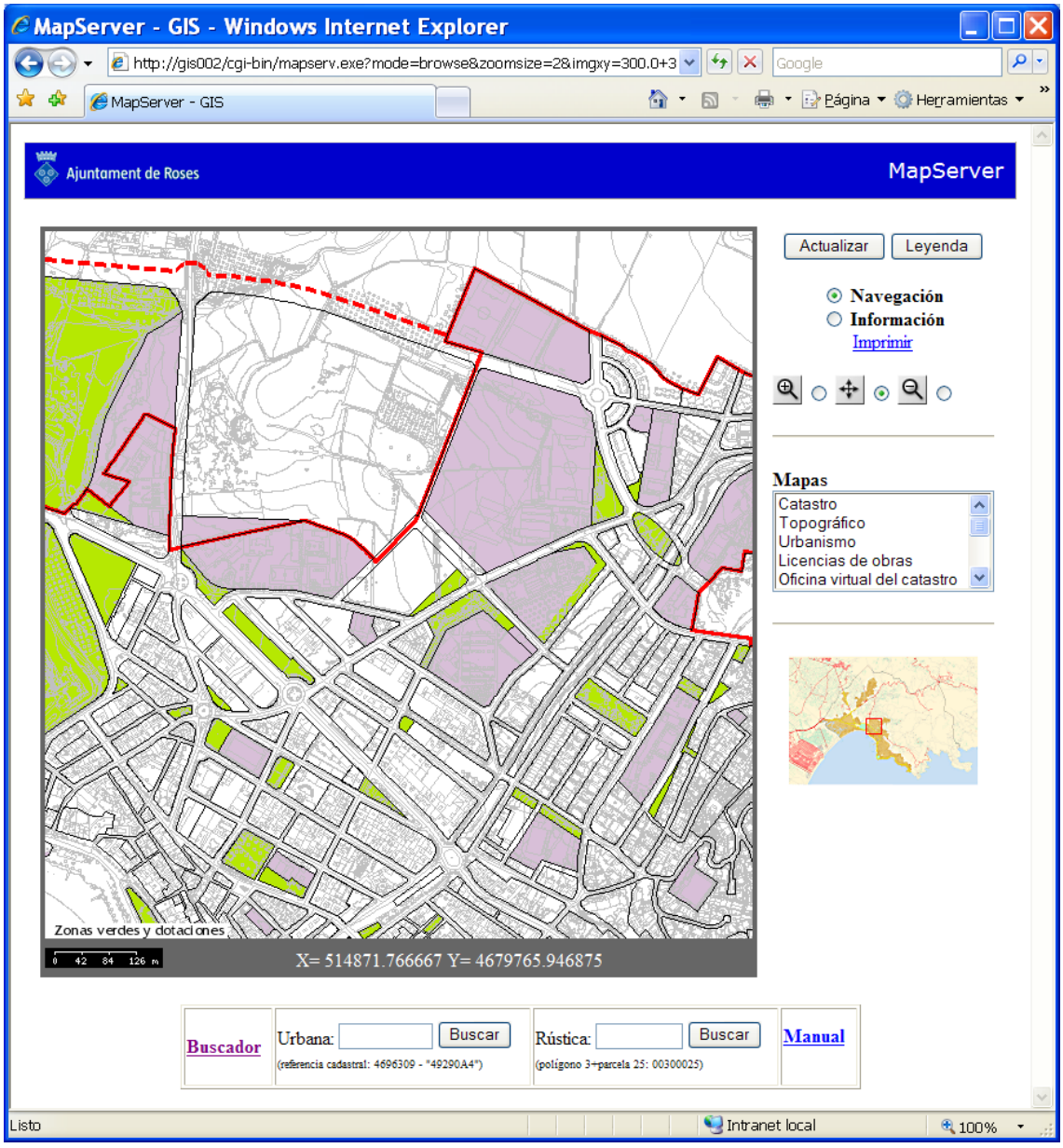


Figura 4.24 Vista del tema zonas verdes y dotaciones.

4.3.9. Servicios urbanísticos

Por servicios urbanísticos entendemos los elementos y redes de los servicios básicos municipales: alcantarillado, electricidad, agua, gas y telefonía. Actualmente se dispone de la información de todos estos servicios en formato dgn y se están dotando de topología y atributos con Geomedia. La red y elementos del alcantarillado ya está en su última fase, la de definición de simbología y revisión general.

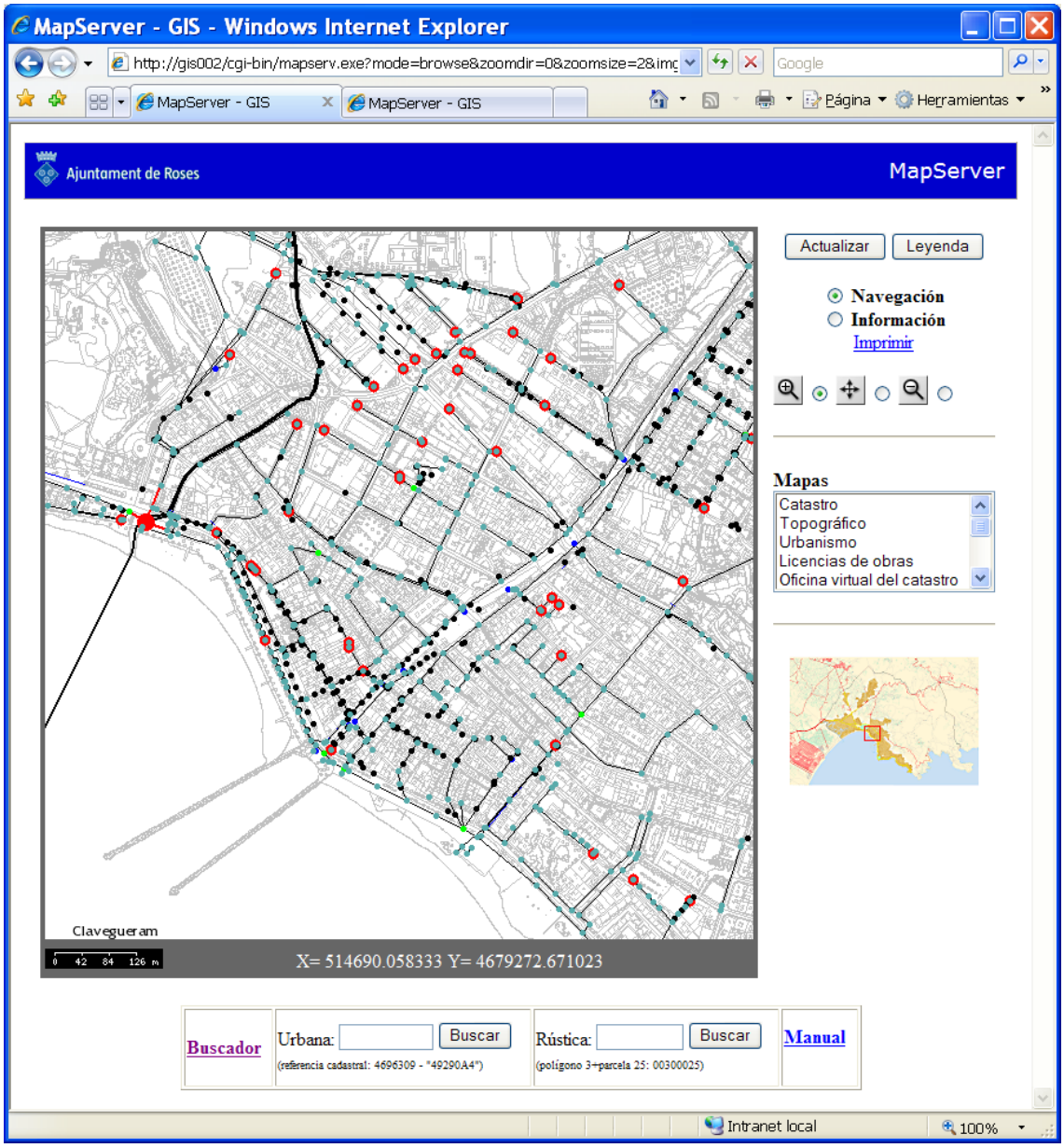


Figura 4.25 Vista del tema servicios urbanísticos.

Los ficheros que utilizamos son:

clavegueram.map

Ficheros shp	Tipo	Descripción
Topoline5000	Líneas	Topográfico 1:5000 de la zona rústica
lavilaline	Líneas	Topográfico 1:1000 de la zona 'la vila'
Buscafum line	Líneas	Topográfico 1:1000 de la zona 'busca-fumats'
Stamarga line	Líneas	Topográfico 1:1000 de la zona 'sta margarita'
Puigrom line	Líneas	Topográfico 1:1000 de la zona 'puigrom'
Conexioebar1	Líneas	Conexiones a ebars
Impulsió1	Líneas	Tuberías de impulsión
Reixes1	Líneas	Rejillas
Tram_tub1	Líneas	Tramos de tubería
Inici_tram1	Puntos	Inicio de tramos de tuberías
Abocaments1	Puntos	Pozos de vertido
Ebars1	Puntos	Estaciones de bombeo de aguas residuales
Embornals11	Puntos	Imbornales
Embornals21	Puntos	Imbornales
Pous_ficticis1	Puntos	Pozos de registro fictícios
Pous1	Puntos	Pozos de registro

Tanto los elementos lineales como los puntuales tienen mucha información asociada. Si pedimos por ejemplo información de un tramo de tubería:

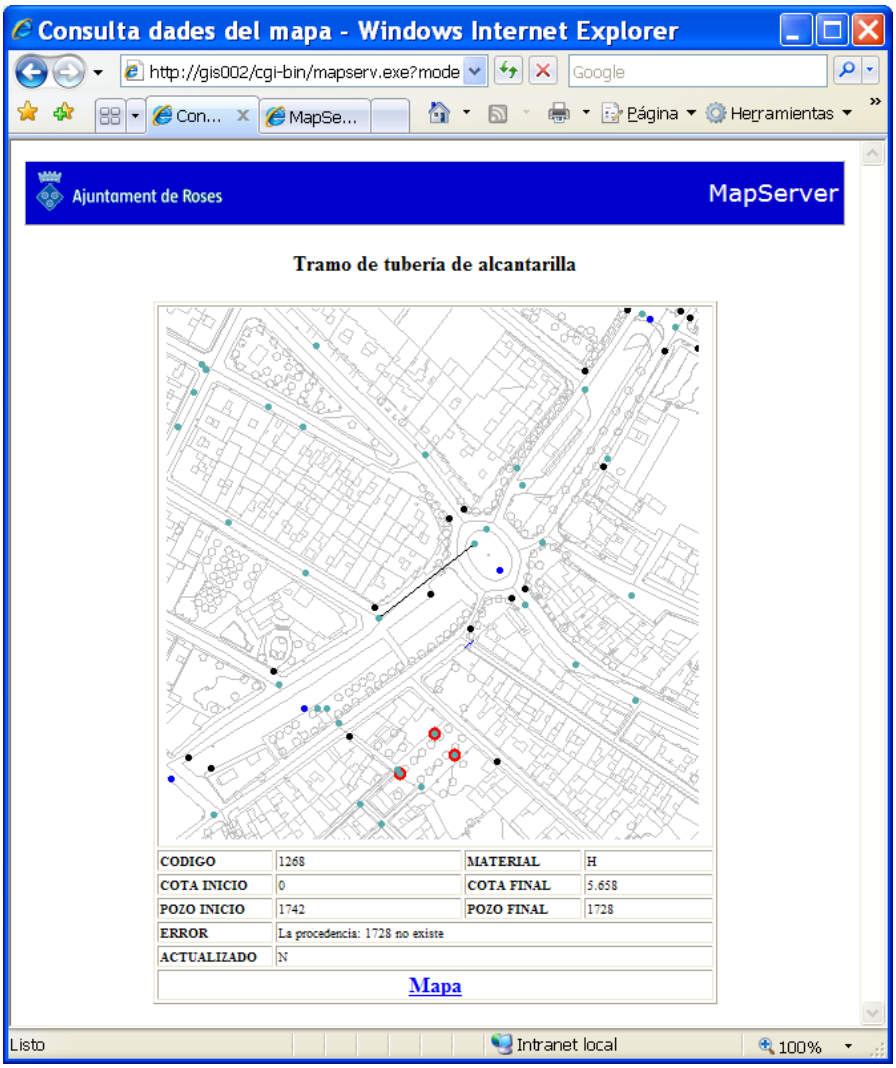


Figura 4.26 Página de información de un tramo de tubería de alcantarilla.

Toda la información de estos elementos está en los ficheros dbf. No tienen asociada ninguna base de datos.

4.3.10. Información gráfica WMS

Cuando definíamos el diseño conceptual de la aplicación al inicio de este mismo capítulo, en el cuadro resumen de la información que íbamos a tratar señalábamos algunos temas como tipo WMS. Estos temas eran el catastral, el topográfico y el ortofotográfico. Los dos primeros se han definido ya con datos locales, pero también es conveniente añadir otras fuentes de cartografía.

WMS Catastro
La Dirección General de Catastro, a través de la Oficina Virtual publica la cartografía catastral digital utilizando el estándar Web Map Service. Aunque es el ayuntamiento el que hace el mantenimiento de esta cartografía, y, por lo tanto, dispone de manera directa de las últimas actualizaciones, a veces se resuelven recursos que afectan a la cartografía

y se modifica en la Gerencia Territorial. La manera más rápida es acceder a la cartografía que han modificado. MapServer es capaz de actuar como cliente para acceder a los servidores WMS de cartografía. En concreto, para acceder a la información catastral, el objeto LAYER es el siguiente:

```
LAYER
NAME "Cartografia"
TYPE RASTER
STATUS default
CONNECTION "http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx?"
CONNECTIONTYPE WMS
METADATA
    "wms_srs" "EPSG:23031"
    "wms_name" "Cartografia"
    "wms_server_version" "1.1.0"
    "wms_formatlist" "image/gif,image/png,image/jpeg,image/wbmp"
    "wms_format" "image/png"
END
END
```

Todos estos parámetros de conexión los proporciona el mismo servidor.
Para que funcionen correctamente estas capas WMS es necesario añadir un objeto más a parte de la capa correspondiente; el objeto PROJECTION.
Hasta ahora no ha sido necesario ya que trabajábamos exclusivamente en un ámbito local, y toda la cartografía que hemos usado está en el mismo sistema de coordenadas y de referencia.
Los parámetros de este objeto PROJECTION para la zona donde estamos trabajando son los siguientes:

```
PROJECTION
        "proj=utm"
        "ellps=intl"
        "zone=31"
        "units=m"
        "north"
END
```

Todos estos parámetros estan agrupados en un fichero de proyecciones con el código ‘EPSG:23031’.
El resultado de esta conexión será el siguiente:

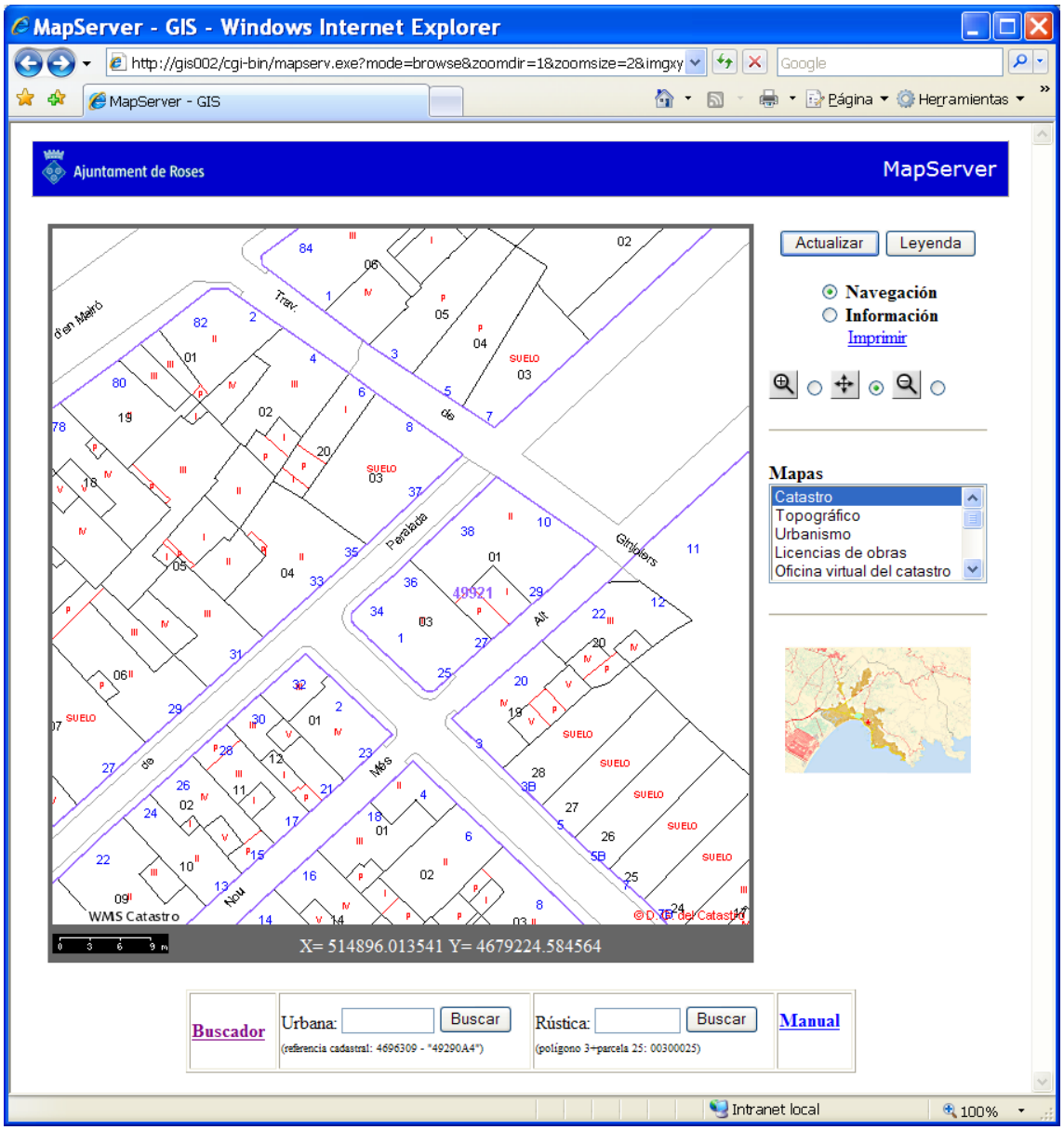


Figura 4.27 Vista del tema WMS Catastro

WMS Ortofotografías

Al igual que el Catastro, el Institut Cartográfic de Catalunya ofrece mucha de su información a través de servicios WMS.

En este caso vamos a acceder a dos ortofotografías: una a escala 1:25.000 y la otra a escala 1:5.000. Configuraremos el fichero 'mapa' de tal manera que nos active una u otra dependiendo de la escala. Las dos LAYERS serán las siguientes:

```
LAYER
NAME "Ortofoto_5000"
TYPE RASTER
STATUS default
CONNECTION "http://galileo.icc.es/wms/servlet/icc_orto5m_r_r?"
CONNECTIONTYPE WMS
maxscale 6000
```

```
METADATA
    "wms_srs" "EPSG:23031"
    "wms_name" "Ortofoto_5000"
    "wms_server_version" "1.1.0"
    "wms_formatlist" "image/png,image/jpeg"
    "wms_format" "image/png"
END
END

LAYER
NAME "Ortofoto_25000_color"
TYPE RASTER
STATUS default
CONNECTION "http://galileo.icc.es/wms/servlet/icc_orto25m_r_r?"
CONNECTIONTYPE WMS
minscale 6000
METADATA
    "wms_srs" "EPSG:23031"
    "wms_name" "Ortofoto_25000_color"
    "wms_server_version" "1.1.0"
    "wms_formatlist" "image/png,image/jpeg"
    "wms_format" "image/png"
END
END
```

Como en el caso anterior, deberemos insertar también un objeto PROJECTION en este fichero.

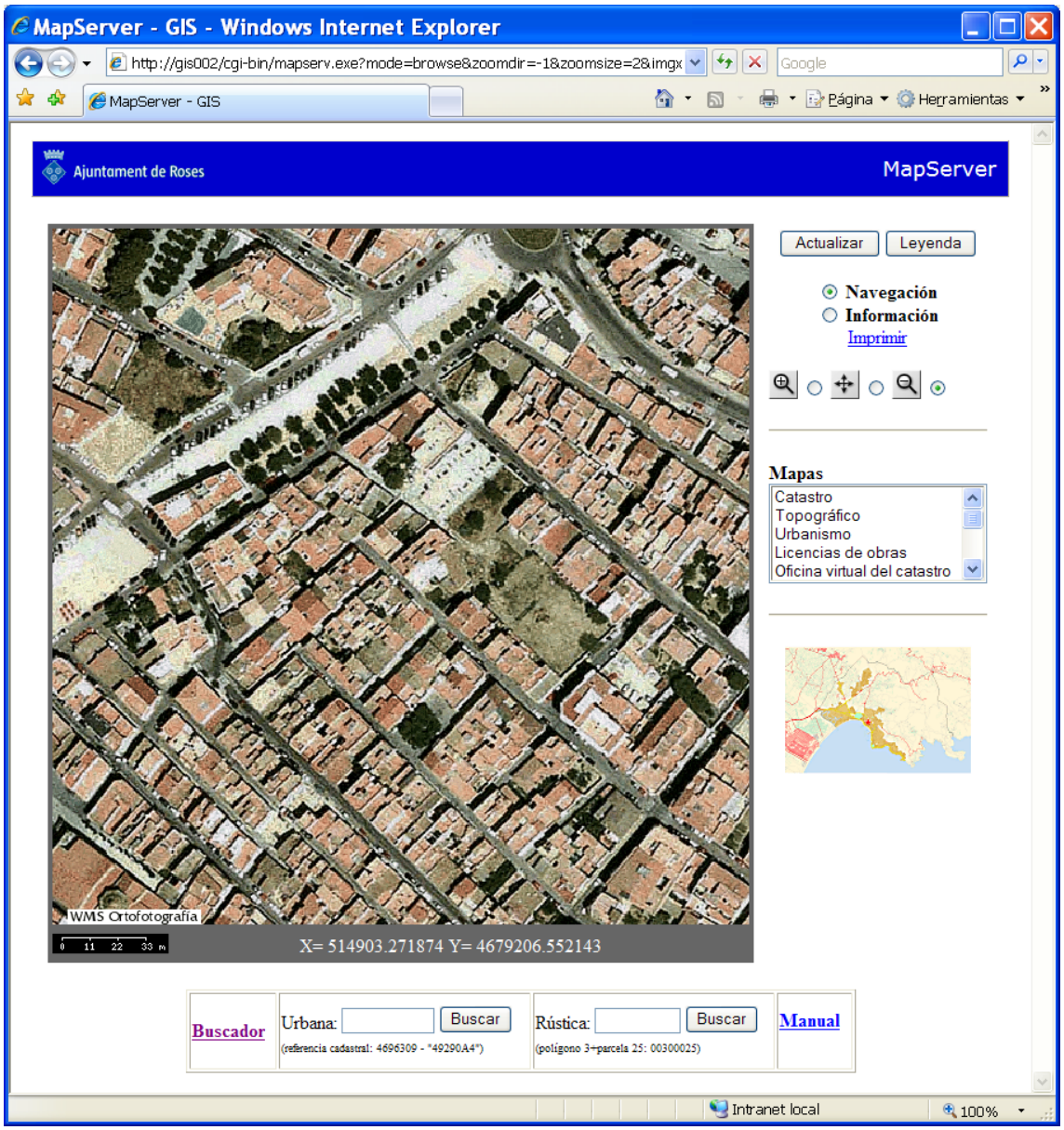


Figura 4.28 Vista del tema WMS Ortofotografía

4.4. Las consultas

Hemos visto que en algunos de los mapas anteriores, al pedir información de los elementos, nos mostraba algunos datos en la plantilla de consulta. Estos datos están contenidos en el fichero dbf de las entidades. En algunas de estas plantillas teníamos acceso a otro tipo de datos que están en bases de datos externas. Estas bases de datos son documentos mdb a los que se accede usando el lenguaje ASP.
Otro tipo de consulta es la que aparece en el fichero plantilla de la aplicación (en la parte inferior) que nos permite localizar parcelas rústicas y urbanas por su referencia catastral. Es del tipo ITEMQUERY, como vimos en el capítulo 3.7 que trataba de consultas.
Vamos a analizar las consultas ASP. El objetivo es interconectar MapServer con MS Access. Uno de los mayores problemas que se planteó en el momento de utilizar la cartografía con MapServer fue el de localización de entidades (parcelas en primera instancia). Se solucionó parcialmente añadiendo la funcionalidad ITEMQUERY que hemos visto ya, pero no era suficiente. Normalmente la gente no utiliza referencias catastrales para localizar elementos, sinó la dirección postal. De esta manera se fueron proponiendo soluciones posibles como:

- Crear una base de datos que relacione las referencias catastrales con la dirección.
- Insertar en las parcelas el nombre de la calle y el número en dos campos.

Pero ninguno de los dos ofrecía una solución definitiva. La mejor opción, que además permite seleccionar usando otros parámetros, es consultar directamente a la base de datos de IBI a través de un formulario ASP, pero esto generó otro problema: ASP y Apache no son directamente compatibles. El lenguaje ASP ha sido creado para funcionar con IIS de Microsoft. Aunque se ha llegado a utilizar este lenguaje con Apache a través de un módulo Perl, no llega a ser completamente compatible.
Hay dos modos de solventar este contratiempo: usando dos servidores, uno con Apache y el otro con IIS, ejecutando MapServer en el primero y el formulario ASP en el segundo, y el que se ha implementado que es instalar Apache y IIS en el mismo servidor.
Esto se hace siguiendo estos pasos:

1. Parar Apache.
2. Ir al Panel de control>herramientas administrativas>servicios de IIS
3. Pinchar en Sitio Web predeterminado con el botón derecho y elegir ‘propiedades’.
4. Entrar el valor ‘8080’ en Puerto TCP.
5. Aceptar e iniciar los dos servidores.

Todos reconocemos http://www.roses.cat como una dirección web, pero lo que normalmente no se sabe es que toda dirección web tiene un puerto. Si no se especifica, toma el puerto por defecto que es el 80 (la inmensa mayoría). Así la dirección anterior y http://www.roses.cat:80 es exactamente la misma. En nuestro caso, siempre hemos utilizado la dirección de intranet http://gis002/. Apache por defecto tenía el puerto 80, y hemos configurado IIS para que utilice el 8080. Cuando queramos usar Apache no hará falta indicar ningún puerto, pero cuando queramos utilizar IIS deberemos acceder usando la siguiente dirección http://gis002:8080.

Normalmente IIS utiliza un sistema de carpetas por defecto distinto al de Apache. Para no mezclar archivos de uno y otro servidor, colocaremos todo lo relativo a IIS en la carpeta 'C:\Inetpub\wwwroot\gis002-8080\'.
En esta carpeta tenemos una base de datos Acces (dadescadastre.mdb) que contiene varias tablas. Una de estas tablas, DadesUTM, está formada por varias columnas de la base de datos de IBI:

UTM: referencia catastral
SG: sigla de la calle
Nom: nombre de la calle
Num: número postal
Esc: escalera
Pis: piso
Pta: puerta
DNI: dni del propietario
Prop: propietario

DadesUTM : Tabla

UTM	SG	Nom	Num	Esc	Pis	Pta	DNI	Prop
2489601	AV	DE LA BOCANA	0008	1	-1	45		
2489601	AV	DE LA BOCANA	0008	1	00	01		
6969024	C	VAN GOGH	0017	1	00	01		
6969024	C	VAN GOGH	0017	1	00	02		
6969024	C	VAN GOGH	0017	1	00	03		
3092903	PS	MARITIM	0060	1	04	34		
3092903	C	PORT-REIG	0008	1	08	50		
1698003	AV	PORT CANIGO	0015	1	02	41		
1698003	AV	PORT CANIGO	0015	1	02	42		
3394001	C	SANT PAU	0005	1	00	06		
3394001	C	MIGUEL DE CERVANTES	0024	3	00	10		
5188209	C	PUIG ROM	0118	1	02	01		
5188209	C	PUIG ROM	0118	1	02	02		
4988510	C	PESCADORS	0015	1	02	01		
4394202	RD	MIQUEL OLIVA PRAT	0004	1	00	01		
4394202	RD	MIQUEL OLIVA PRAT	0004	1	-1	27		
4790404	TR	DE L'ESGLESIA	0014	T	OD	OS		
4790405	C	SANT SEBASTIA	0035	T	OD	OS		
4791601	C	D'EN MAIRO	0054	1	00	01		

Figura 4.29 Vista de la base de datos catastral.

Empecemos creando el formulario de búsqueda. Este será un fichero ASP (buscador.asp) cuyo listado completo se encuentra al final del trabajo, pero lo más significativo es:

```
18  <form method="GET" action="http://gis002:8080/gis002-
    8080/resultados3.asp">
```

Esta línea nos enviará las variables que rellenemos al fichero 'resultados3.asp' que es el que accede a la base de datos, procesa la consulta y presenta los resultados. El fichero 'buscador.asp' visualmente es así:

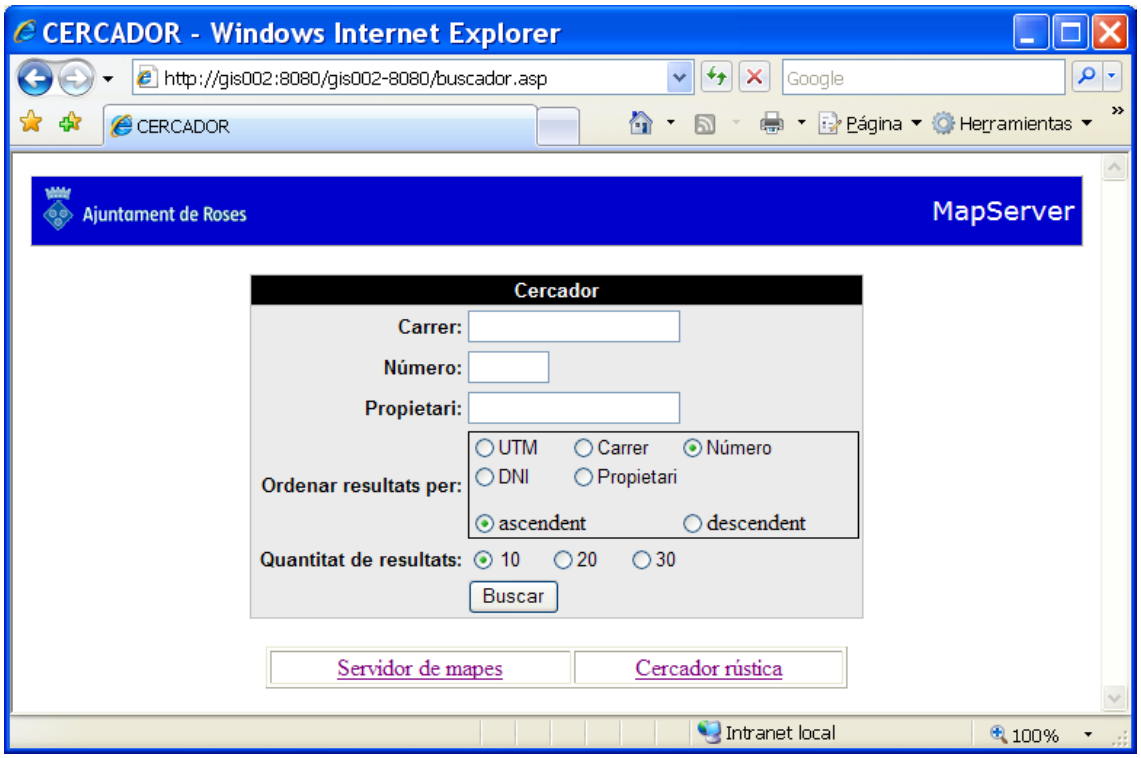


Figura 4.30 Vista la aplicación 'buscador'.

Podemos buscar por calle, número y propietario por separado o usando cualquier combinación de estos campos. Por ejemplo introduciendo el valor 'oslo' en calle, seleccionaremos todos los registros de la base de datos que contienen esa cadena en el campo calle, y así tendremos todas las parcelas de esa calle. Si introducimos 'juan' en el campo 'propietario' obtendremos todas las parcelas cuyo campo 'propietario' contenga dicha cadena.
También enviaremos variables que transmitan de que manera presentar los resultados, como la cantidad que se mostrarán por página, y su orden.
Estos campos de búsqueda se guardan en variables para enviarlos. Por ejemplo, lo que introduzcamos en la caja de texto 'Carrer' nos lo guarda en la variable 'Nom' como se aprecia en la línea correspondiente:

```
28  <font face="Arial"><input type="text" name="Nom" size="20"></font></td>
```

Igualmente, las otras variables de presentación de resultados se envían usando variables:

```
49  <font face="Arial"><input type="radio" value="UTM"
    name="orden"></font></td>
```

Si introducimos 'oslo' en la calle y 'juan' en propietario y pinchamos en 'buscar' se creará la siguiente URL:

http://gis002:8080/gis002-8080/resultados3.asp?Nom=oslo&Num=&Prop=juan&orden=Num&alf=asc&cantidad=10

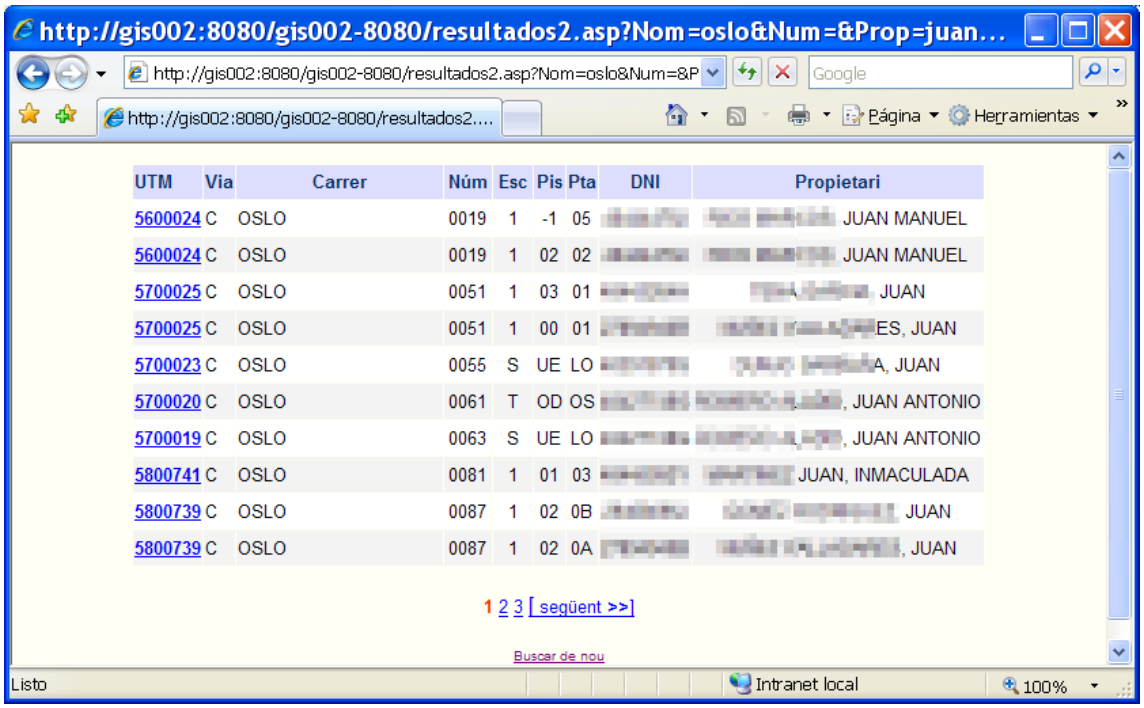


Figura 4.31 Resultados de la búsqueda.

Parcelas de la calle Oslo y cuyo nombre de propietario contiene la cadena ’juan’ ordenados por número de calle, ascendentemente, y de diez en diez.

Cuando encontremos la parcela deseada, al pinchar en su referencia catastral invocará a MapServer en modo ITEMQUERY, le pasará el valor de la referencia y nos mostrará la plantilla de consulta de la parcela. Vamos a ir por partes a ver como hacemos todo esto. El listado completo del código ASP está detallado al final del texto, pero lo más interesante es lo siguiente:

```
34  strsql = "SELECT * FROM DadesUTM where Trim(Esc)<>" AND
    UCase(Nom) like '%" & UCase(Request("Nom")) & "%' AND UCase(Num)
    like '%" & UCase(Request("Num")) & "%' AND UCase(Prop) like '%" &
    UCase(Request("Prop")) & "%' order by "& orden & " "&alf
35  Set oConn = Server.CreateObject("ADODB.Connection")
36  oConn.Open "DRIVER={Microsoft Access Driver (*.mdb)};
    DBQ="&Server.MapPath("DadesCadastre.mdb")
```

Estas tres líneas definen los parámetros de conexión y la consulta SQL a la base de datos. Unas líneas mas abajo, se ejecuta la sentencia que abre la base de datos y aplica las condiciones de consulta especificadas en la variable ‘strsql’:

```
44  RS.Open strSQL, oConn,3,1
```

Buena parte del código es usado para la paginación de los resultados y en el listado completo está comentado, por lo que no incidiremos en esta característica de la aplicación.

Entre la línea 70 y la 97 se crean los encabezados de las columnas, que serán fijos. Y entre las líneas 110 y 121 se insertan los resultados. El más interesante es el de la referencia catastral:

```
111     <td width="6%" height="26" style="border-style:none; border-
        width:medium; background-color: <%= color %>; "><font face="Arial"
        size="2"><b><a href="http://gis002/cgi-
        bin/mapserv.exe?mode=itemquery&qlayer=PARCELA&qitem=UTM&qstring
        =%22<%=RS("UTM")%>%22&map=C%3A%5Cms4w%5Capps%5Ctfc%5C
        catastro.map&mapext=shapes&map_web_template=mapserver48.html"><%=
        RS("UTM")%></a></b> </td>
```

A parte de insertar el valor de la referencia catastral, a este se le aplica un link que invoca a MapServer en modo ITEMQUERY y con el valor de qstring igual a la referencia catastral y los otros parámetros necesarios para hacer la consulta.
Puede parecer extraña la notación de este link por los caracteres especiales utilizados del tipo %5C. Estos caracteres son códigos hexadecimales de símbolos ASCII, y son descodificados por el navegador. Los que aparecen en la sentencia son:

ASCII	Símbolo
%22	“
%3A	:
%5C	\

Al pinchar en una referencia catastral obtendremos la plantilla de consulta de la parcela seleccionada:

Figura 4.32 Plantilla de consulta de las parcelas.

Que es la misma que nos abriría si pedimos información pinchando sobre la parcela.

4.4.1. Catastro

A esta plantilla hemos accedido a través del buscador, pero si lo hacemos pidiendo información en la ventana de mapa, podremos acceder a la información de los propietarios entre otras, como vemos en la parte izquierda de la imagen anterior.
El link 'Propietaris' es:

http://gis002:8080/gis002-8080/resultados3.asp?referencia=5800739&Nom=&Num=&Prop=&orden=Num&alf=asc&cantidad=10

Ejecutamos la aplicación ‘resultados3.asp’ utilizando como filtro la referencia catastral, para obtener un listado de todos los propietarios de la finca con esa referencia.

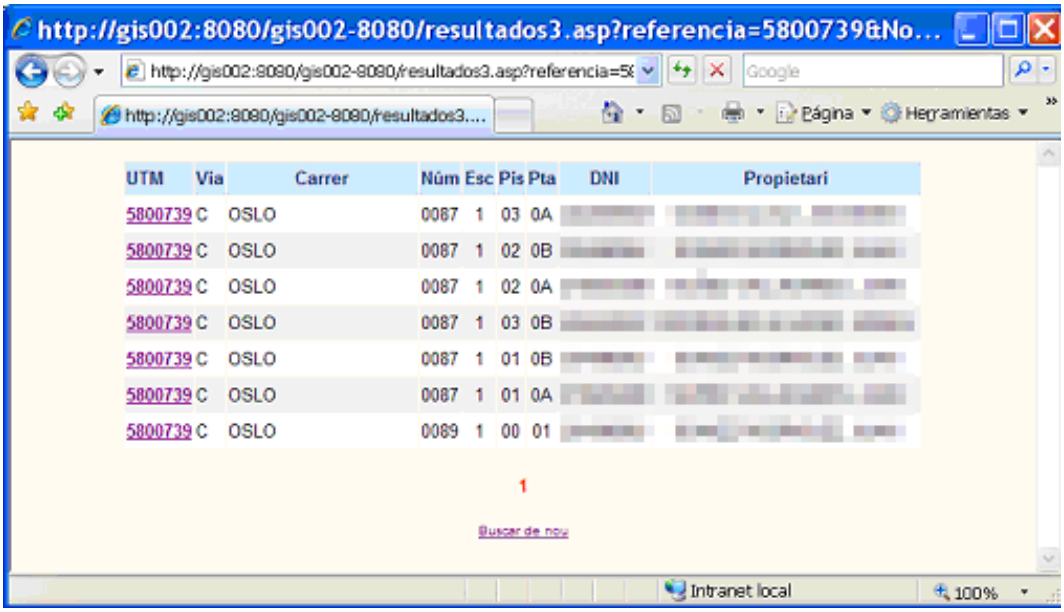


Figura 4.33 Acceso a los datos de propietarios de una parcela.

Otro tipo de datos a los que podemos acceder a través de la planilla de consulta de parcelas son los relativos a las superficies de las construcciones. El link ‘Construccions’ es:

http://gis002:8080/gis002-8080/construccions.asp?referencia=5800739&orden=Num&alf=asc&cantidad=10

Ejecuta la aplicación ‘construccions.asp’ y filtra los resultados utilizando la referencia catastral, como en el ejemplo anterior. El código, que se puede ver al final del trabajo, es muy similar al de la aplicación anterior. Como nota más destacada tenemos que accedemos a dos tablas diferentes que están en la misma base de datos:

```
20  strsql = "SELECT * FROM superficies where UCase(UTM) like '%" &
    UCase(Request("referencia")) & "%'"
21  strsql1 = "SELECT * FROM constru where UCase(UVC_CODI_UTM) like
    '%" & UCase(Request("referencia")) & "%'"
22  Set oConn = Server.CreateObject("ADODB.Connection")
23  oConn.Open "DRIVER={Microsoft Access Driver (*.mdb)};
    DBQ="&Server.MapPath("DadesCadastre.mdb")
```

Y, por lo tanto, debemos crear dos cadenas de consulta SQL para acceder a los datos de las dos tablas.
El resultado de las dos consultas es:

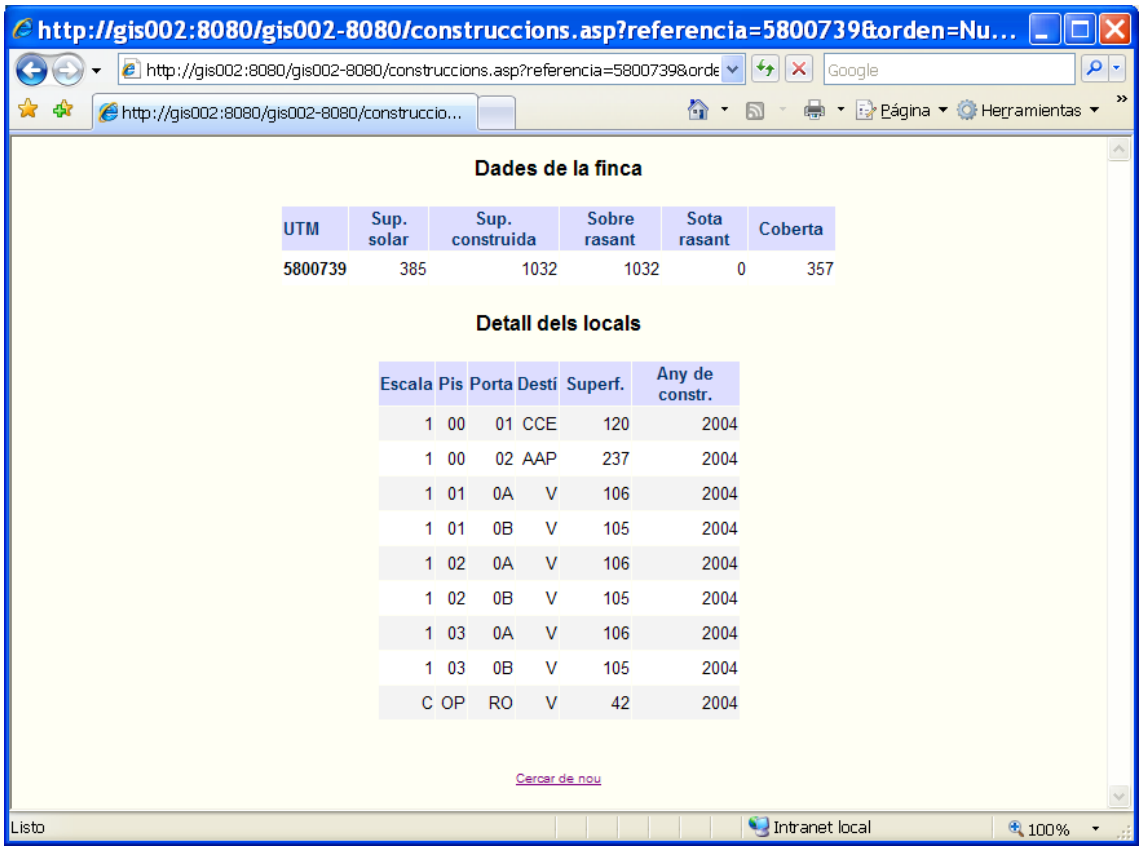


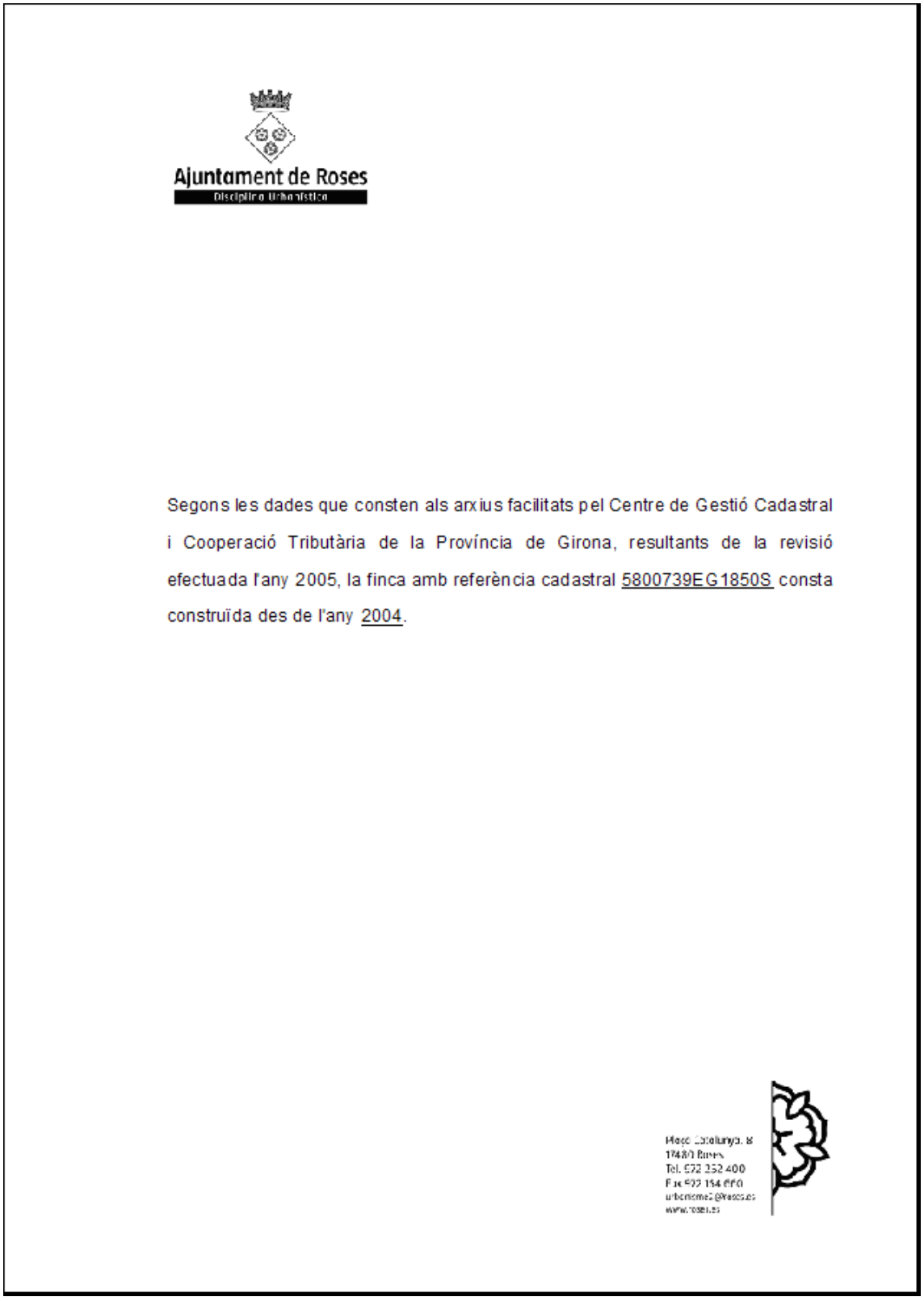
Figura 4.34 Acceso a los datos de construcciones de una parcela.

Otra funcionalidad relativa a las consultas es la creación de informes sobre el año de construcción de las fincas.
Esta consulta se hace utilizando un componente de IIS, llamado Internet Database Conector (IDC). Con el IDC se puede acceder a bases de datos que utilicen SQL como lenguaje de interrogación, como SQL Server o Access, y siempre utilizando los Open Data Base Conectivity (ODBC) de que vienen provistas las bases de datos.
Para que funcione el IDC sólo es necesario escribir un fichero con extensión .idc que contenga una instrucción SQL para la base de datos. Dentro de este fichero, que puede tener cualquier nombre, y entre otros parámetros, se indica el nombre de otro fichero con extensión .htx que hace las veces de plantilla, y da formato a los datos devueltos por la instrucción SQL del primer fichero. En nuestro caso, el fichero IDC es el siguiente:

```
01  Datasource:Prop
02  Template:any.htx
03  SQLStatement:SELECT * FROM [superficies] WHERE UTM='%UTM%'
04  Password:
05  Username:
```

La línea 01 indica el origen de datos (ODBC). Este se configura en Panel de control -> Herramientas administrativas -> Orígenes de datos (ODBC). Se crea uno nuevo (con el nombre ‘Prop’ en este ejemplo) que apunte a la base de datos a la que queremos acceder. La línea 02 identifica la plantilla que se usará para presentar los resultados y la 03 indica la instrucción SQL que usaremos para filtrar los resultados (el listado del fichero plantilla any.htx se encuentra al final del texto).

Si pinchamos en el botón ‘Informe constr.’ Obtenemos como resultado un informe sobre el año de construcción del edificio, con el formato de papel del departamento de disciplina urbanística, que es quien emite estos informes.


Ajuntament de Roses
Disciplina Urbanística

Segons les dades que consten als arxius facilitats pel Centre de Gestió Cadastral i Cooperació Tributària de la Província de Girona, resultants de la revisió efectuada l'any 2005, la finca amb referència cadastral 5800739EG1850S consta construïda des de l'any 2004.

Plaça Catalunya, 8
17480 Roses
Tel. 972 252 400
Fax 972 154 670
urbanisme2@roses.cat
www.roses.cat

Figura 4.35 Certificado de la fecha de construcción.

Para cerrar este capítulo de consultas de datos de parcelas, el link ‘Inspecció’ de la plantilla de consulta, nos abre una carpeta cuyo nombre es la referencia catastral y donde los inspectores de la vía pública colocan fotografías e informes de la parcela.

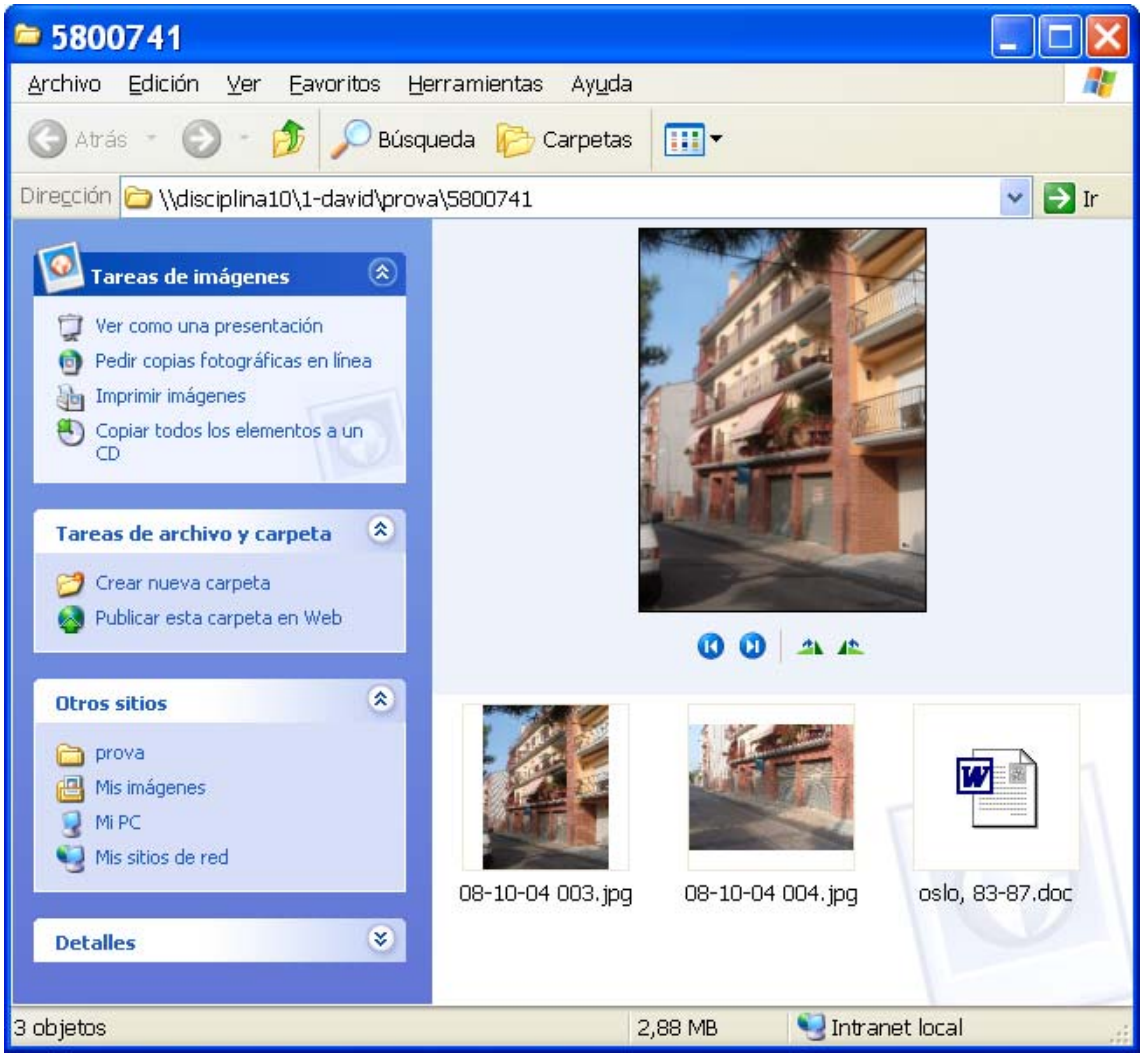


Figura 4.36 Acceso a los documentos de inspección de una parcela.

El código del fichero plantilla es:

```
43  <a href="\\disciplina10\1-david\prova\[UTM]">Inspecció</a>
```

La información está en un ordenador (disciplina10) en el departamento de disciplina urbanística.

Catastro de rústica

Si ahora pedimos información de una parcela rústica, se nos muestra la plantilla de consulta propia de las parcelas rústicas:

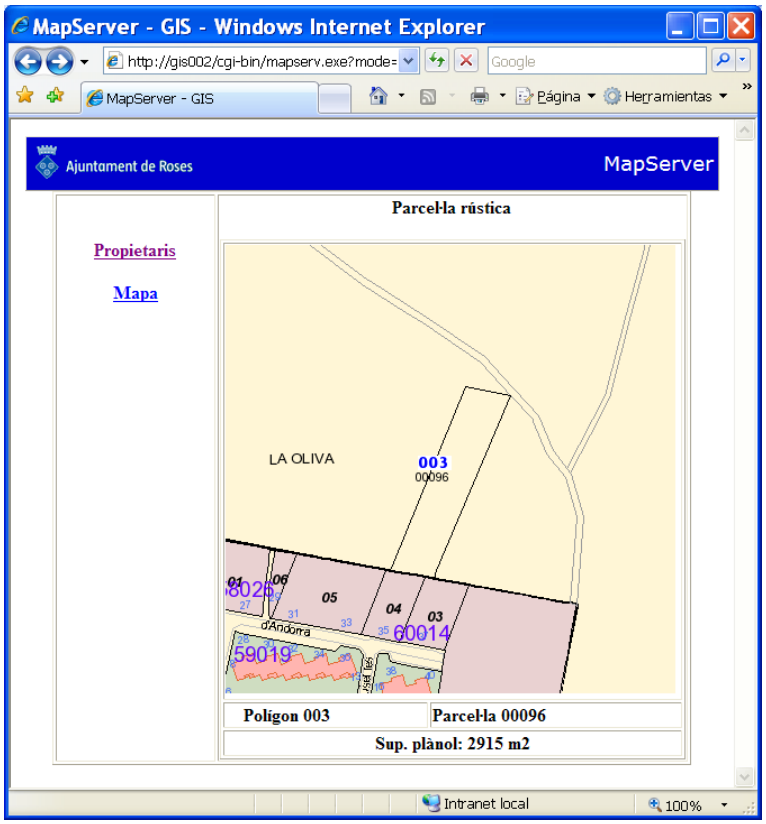


Figura 4.37 Acceso a los datos de una parcela rústica.

El link ‘Propietaris’ es:

http://gis002:8080/gis002-8080/resultados_rustica.asp?referencia=00300096&nif=&prop=&orden=Prop&alf=asc&cantidad=10

Y vemos que es muy similar al que hemos visto cuando tratábamos con parcelas urbanas: ejecuta la aplicación ‘resultados_rustica.asp’ y le pasamos como filtro la referencia catastral, que en este caso está compuesta por el polígono y la parcela. El resultado es también del mismo estilo.

Figura 4.38 Acceso a los datos de propietarios de una parcela rústica.

Con los datos propios de las parcelas rústicas.

4.4.2. Actividades

Los datos a los que podemos acceder están almacenados en la base de datos 'activitats.mdb'. Lo haremos utilizando otra variante ASP. Si solicitamos información de una actividad accedemos a su plantilla de consulta:

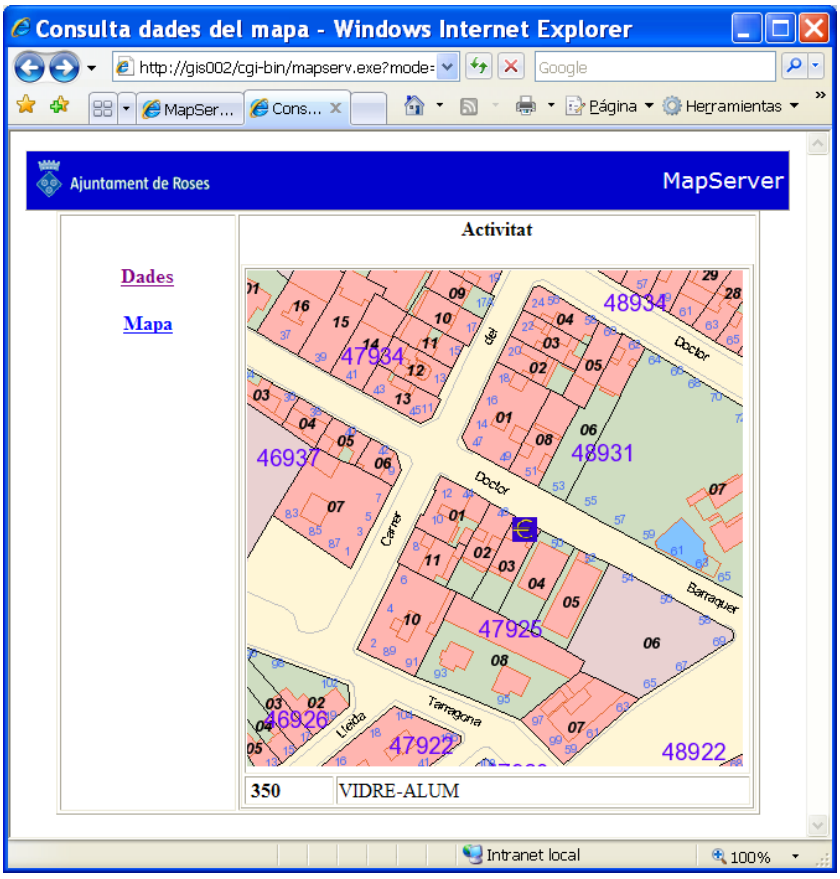


Figura 4.39 Página de información de las actividades económicas.

El link ‘Dades’ es:

http://gis002:8080/gis002-8080/activitats.asp?Zid=350

Es una invocación a la aplicación ‘activitats.asp’ que genera una consulta a la base de datos con la condición de que el código de la actividad sea igual al que estamos solicitando (en este caso 350).
En este caso, no nos interesa mostrar los resultados en filas, ya que son muchos datos y se haría una página enorme y poco clara. La organizaremos de forma similar a un formulario. El código principal es el siguiente:

```
20  <%
21     Dim oConn,strSQL, objRS
22     Set oConn = Server.CreateObject("ADODB.Connection")
23     oConn.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
    Server.MapPath("activitats.mdb"))
24     strSQL = "SELECT * FROM ACTIVITATS where Zid = " &
    Request.QueryString("Zid")
25     Set objRS = oConn.Execute(strSQL)
26  %>
27  <%
28  Response.Write( "<TABLE  BORDER=""1""">" & vbCrlf)
29  Response.Write( "<TR>" & vbCrlf)
30  Response.Write( "<TD><B>Nom activitat: </B>" & objRS("nomactivitat") &
    "</TD>" & vbCrlf )
31  Response.Write( "<TD><B>Zid: </B>"& objRS("Zid") &"</TD>" & vbCrlf )
32  Response.Write( "<TR></TR>" & vbCrlf )
33  Response.Write( "<TD><B>Annex llei: </B>"& objRS("annexllei")
    &"</TD>" & vbCrlf )
34  Response.Write( "<TR></TR>" & vbCrlf )
35  Response.Write( "<TD><B>Codi: </B>" & objRS("codi") &"</TD>" &
    vbCrlf )
36  Response.Write( "<TD><B></B>"& objRS("descripciócodi") &"</TD>" &
    vbCrlf )
37  Response.Write( "<TR></TR>" & vbCrlf )
38  Response.Write( "<TD><B>Descripció: </B>"& objRS("descripcióactivitat")
    &"</TD>" & vbCrlf )
39  Response.Write( "<TR></TR>" & vbCrlf )
40  Response.Write( "<TD><B>Situació: </B>"& objRS("adreça") &"</TD>" &
    vbCrlf )
41  Response.Write( "<TR></TR>" & vbCrlf )
42  Response.Write( "<TD><B>Titular: </B>"& objRS("titular") &"</TD>" &
    vbCrlf )
43  Response.Write( "<TD><B>DNI: </B>"& objRS("dni") &"</TD>" & vbCrlf
    )
44  Response.Write( "<TR></TR>" & vbCrlf )
45  Response.Write( "<TD><B>Inici exp: </B>"& objRS("datainiciexp")
    &"</TD>" & vbCrlf )
```

```
46  Response.Write( "<TD><B>Concessió: </B>"& objRS("dataconcessio")
    &"</TD>" & vbCrlf )
47  Response.Write("</TABLE>")
48     oConn.Close
49     set objRS = nothing
50     set oConn = nothing
51  %>
```

Entre las líneas 21 y 25 definimos la consulta y abrimos la base de datos. Hasta la línea 47 creamos una tabla y añadimos los resultados. Finalmente cerramos la conexión y vaciamos las variables. El resultado es más adecuado a nuestras necesidades:

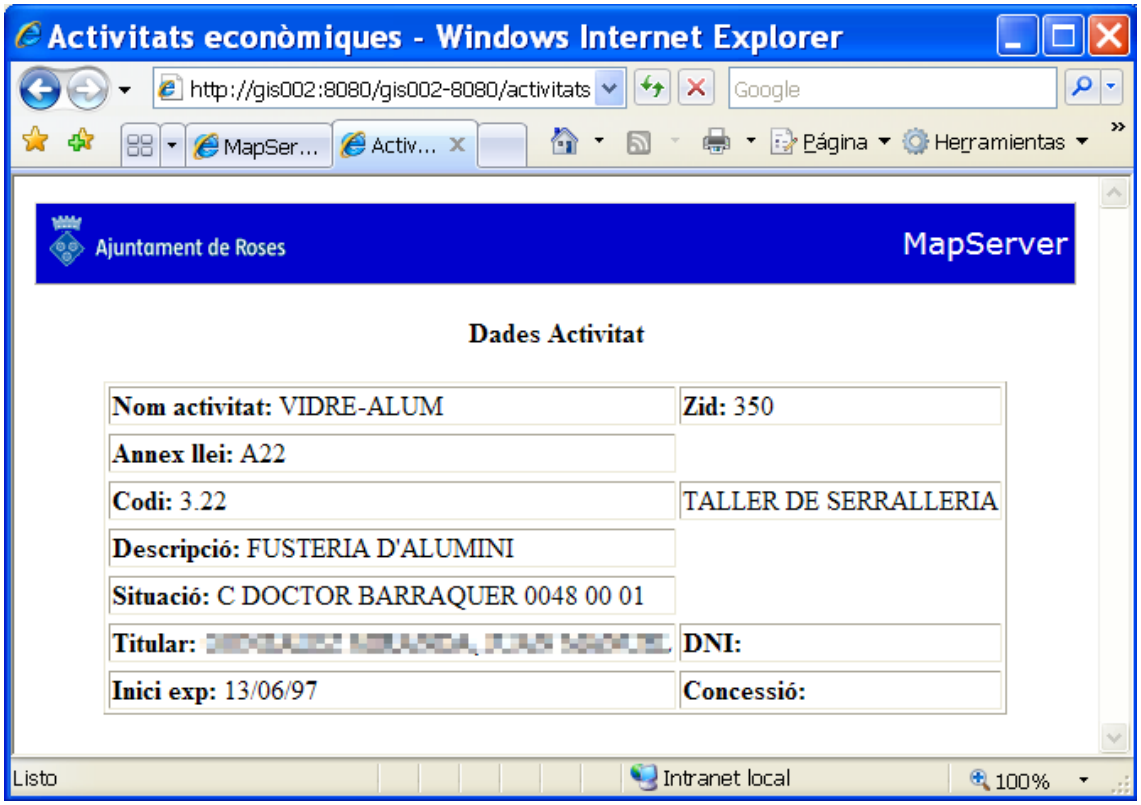


Figura 4.40 Acceso a los datos de las actividades económicas.

4.4.3. Disciplina urbanística

La consulta que efectuamos aquí es del mismo estilo que la que acabamos de ver para las actividades. Disponemos de una base de datos (disciplina.mdb) y vamos a crear un formulario con los resultados. La plantilla de consulta de las licencias de obras era:

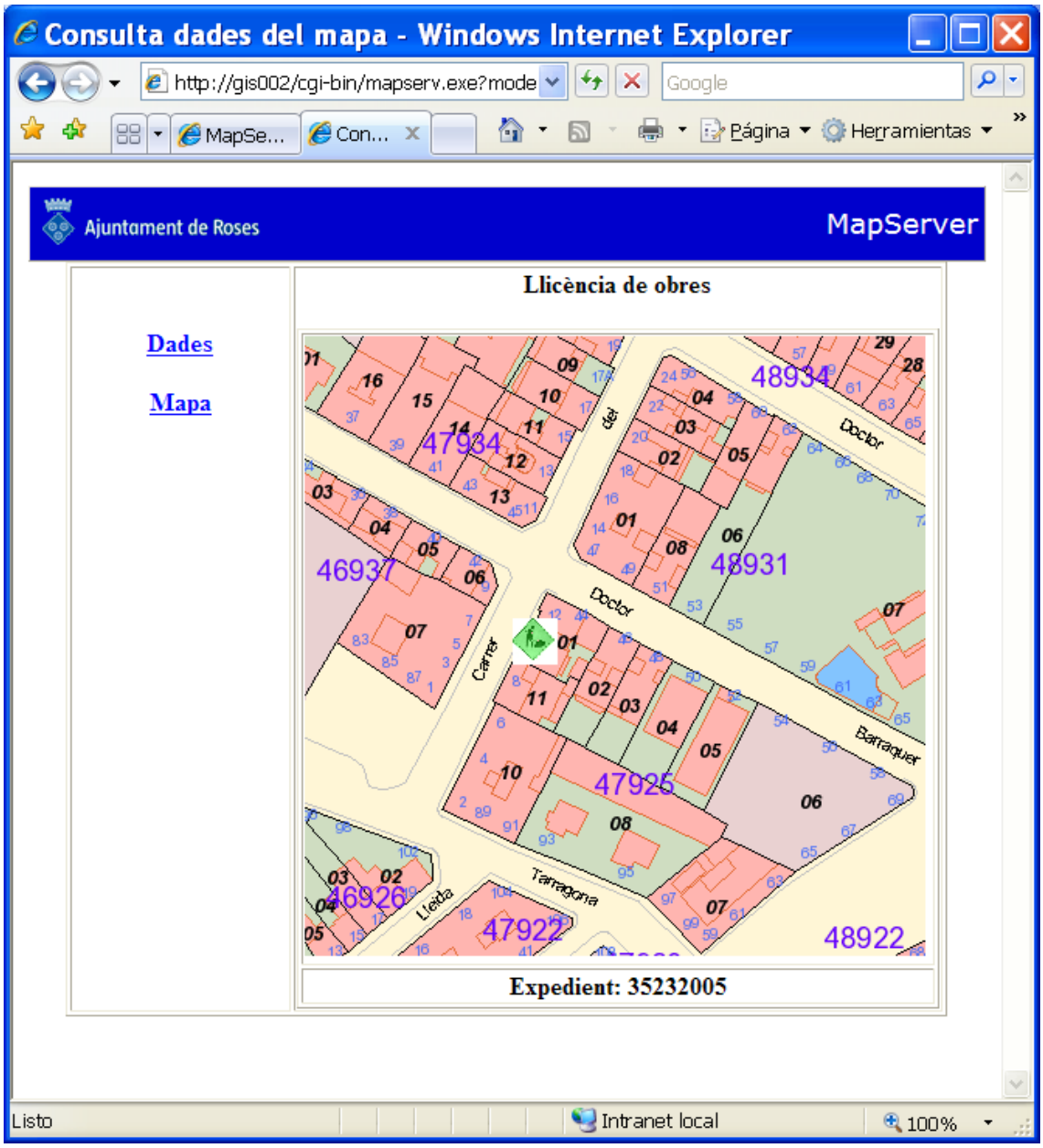


Figura 4.41 Página de información de las licencias de obras..

El link ‘Dades’ nos ejecuta la aplicación ‘obras.asp’ y con el número de expediente como parámetro de consulta:

http://gis002:8080/gis002-8080/obras.asp?expe=35232005

El código de la aplicación también es similar al de la anterior, y de igual manera se puede consultar al final del proyecto. El resultado es el que sigue:

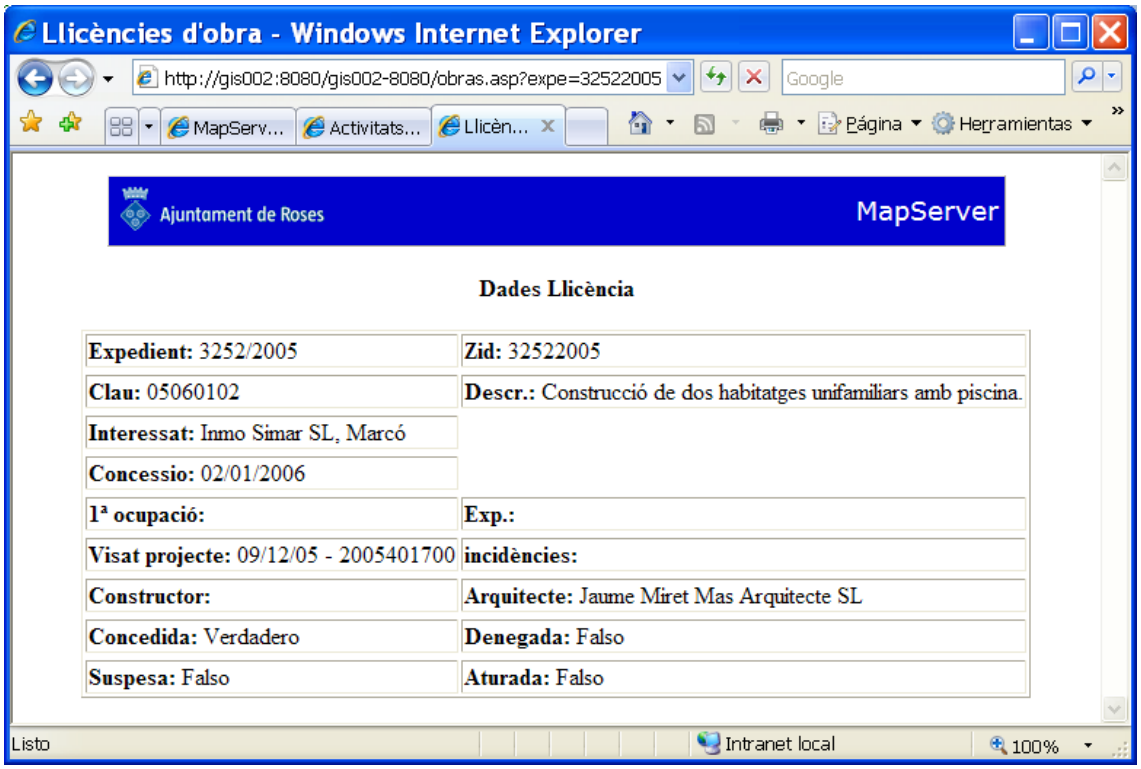


Figura 4.42 Acceso a los datos de las licencias de obras.

4.5. Funciones avanzadas

En este capítulo agruparemos una serie de funciones más sofisticadas y de gran practicidad de caras al usuario final. Estas funciones serán:

- Ventana para la leyenda
- Plantilla de impresión
- Integración con otras aplicaciones
- Edición de entidades puntuales
- Manual de usuario

Todas y cada una de ellas ha sido el resultado de sugerencias hechas por los usuarios, que en su trabajo diario necesitaban estas opciones.

4.5.1. Ventana para la leyenda

En los ejemplos tratados en el capítulo 4, integrábamos la leyenda en el fichero plantilla como se puede apreciar en el ejemplo 9:

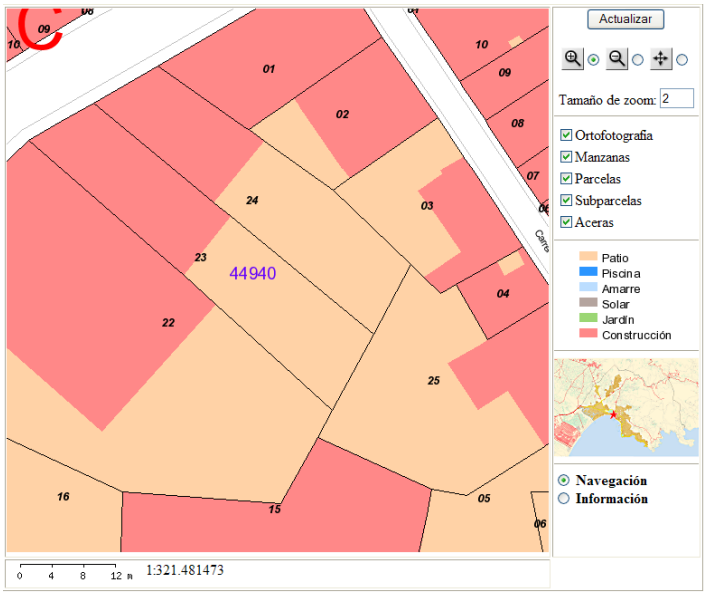


Figura 4.43 Vista del ejemplo 9.

En ocasiones, debido a la gran cantidad de elementos a representar esta leyenda puede hacerse muy extensa y puede interferir negativamente a la hora de trabajar, ya que nos quedaría una interfaz muy grande y difícil de gestionar. Para solventar este problema, se ha creado un botón en el fichero plantilla que lo que nos hace es abrir una ventana de navegador nueva e insertarnos la leyenda en esta nueva ventana.

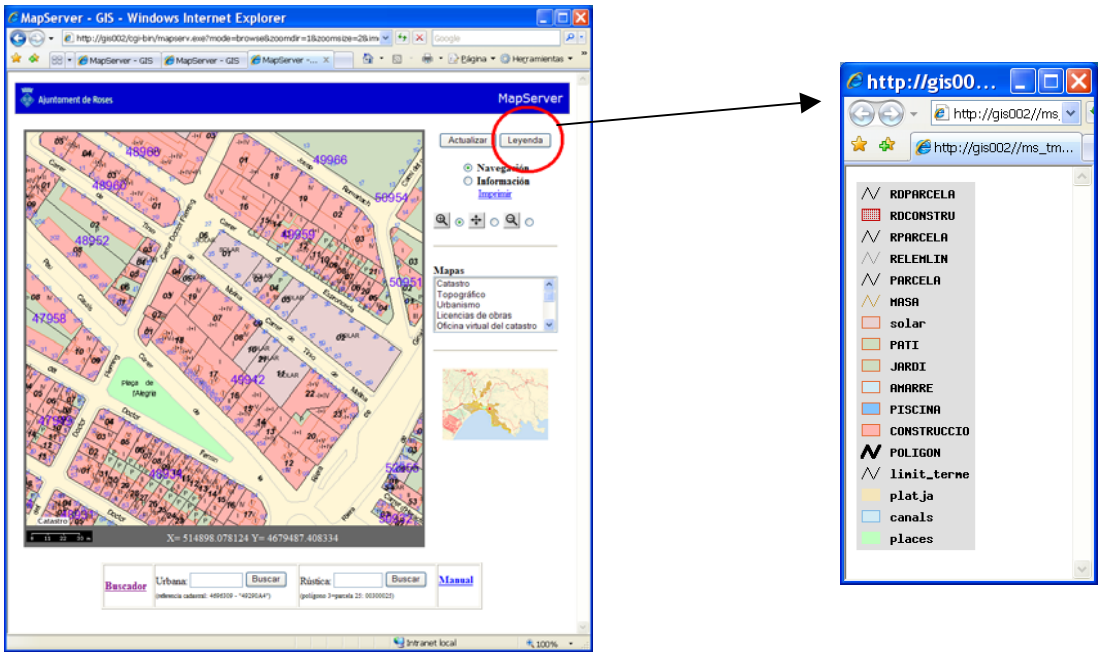


Figura 4.44 Botón para activar la leyenda.

De esta manera queda más limpio el fichero plantilla.
En el código del fichero plantilla tendremos:

```
35  <center><input type="submit" value="Actualizar"> <input
    name="imprimir" type="button" onClick="PopWindow()"
    value="Leyenda"></center>
```

Es la instrucción que crea el botón y cuando se pincha en él ejecuta la función 'PopWindow' que está unas líneas más abajo en el mismo código:

```
70  <script language="JavaScript" type="text/javascript">
71  <!--
72  function PopWindow()
73  {
74  window.open('http://GIS002/[legend]','Leyenda','width=200,height=500,m
    enubar=yes,scrollbars=yes,toolbar=yes,location=yes,directories=yes,resiza
    ble=yes,top=0,left=0');
75  }
76  //-->
77  </script>
```

Es una función JavaScript, que abre una ventana nueva con la imagen [legend] (que es la imagen de la leyenda que crea MapServer) con una serie de parámetros propios de una ventana de navegador. Si no cerramos esta ventana de leyenda y seleccionamos otro mapa al pinchar en el botón de leyenda no nos abre una ventana nueva, sinó que sustituye la leyenda antigua por la nueva.

4.5.2. Plantilla de impresión

Si bien es posible imprimir directamente el plano que vemos en el archivo plantilla, no queda demasiado atractivo, ya que nos imprime todos los botones y controles.

Para poder cambiar de plantilla, conservando el resto de parámetros (el mapa, la escala, la extensión, etc.) debemos cambiar un parámetro en todos los ficheros 'mapa' que tengamos. Por ejemplo el fichero 'mapa' del tema catastro empezaba así:

```
01  MAP
02  NAME "catastro"
03  SIZE 600 600
04  IMAGETYPE png
05  EXTENT 510720 4676215 519430 4683355
06  SHAPEPATH "shapefiles"
07  FONTSET "c:/ms4w/apps/tfc/etc/fonts.txt"
08  SYMBOLSET "c:/ms4w/apps/tfc/etc/symbols.sym"
09  UNITS METERS
```

Pues tendremos que añadir la siguiente línea:

```
36  TEMPLATEPATTERN "mapserver48"
```

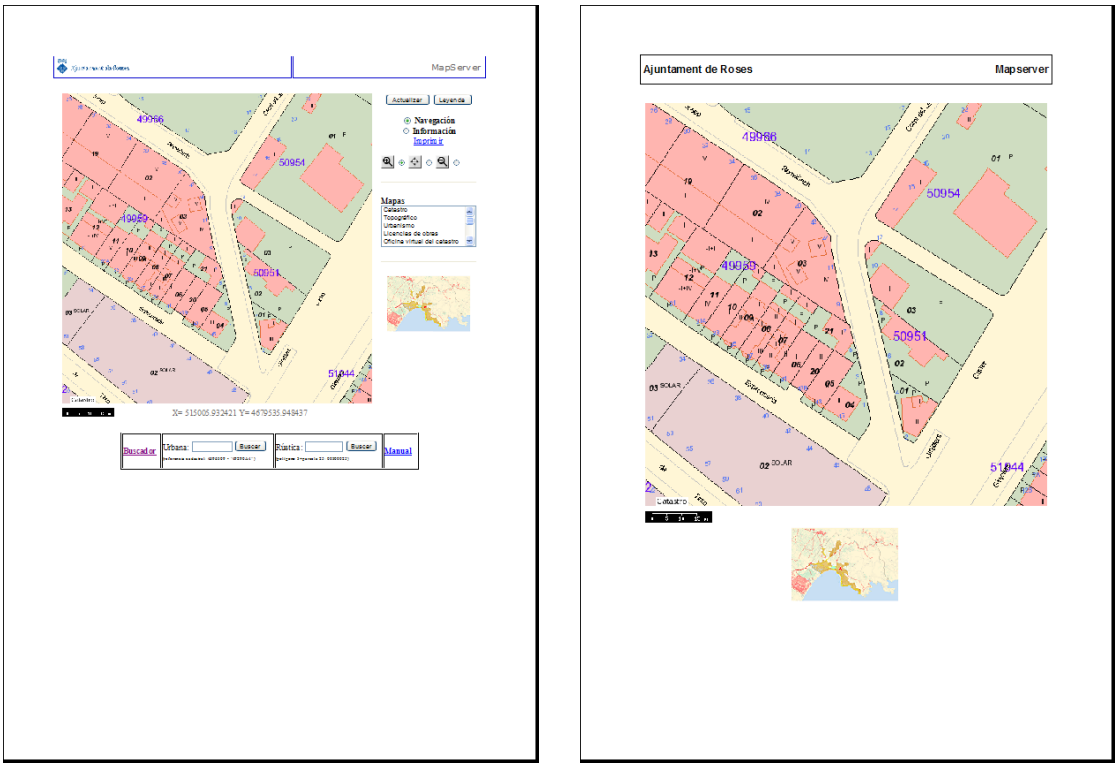
Y borrar la línea del objeto WEB que contenga el parámetro TEMPLATE:

```
37   WEB
38     TEMPLATE   "mapserver48.html" #BORRAR
39     IMAGEPATH "/ms4w/tmp/ms_tmp/"
40     IMAGEURL "/ms_tmp/"
41     EMPTY "/tfc/sin_elementos.html"
42   END
```

Estos cambios nos permiten no asociar a un fichero 'mapa' únicamente una plantilla, sinó todas las plantillas que contengan la expresión 'mapserver48'. Por lo tanto, si nos creamos una plantilla limpia de controles podremos insertar en ella el mapa y otros componentes como la escala o el mapa de referencia.
Una vez cambiados estos ficheros, creamos un link en el fichero plantilla:

```
46  <a href="http://gis002/cgi-
    bin/mapserv.exe?imgxy=300.0+300.0&imgext=[mapext]&map=[map]&ro
    ot=%2Ftfc%2F&savequery=true&program=%2Fcgi-
    bin%2Fmapserv.exe&map_web_imagepath=%2Fms4w%2Ftmp%2Fms_t
    mp%2F&map_web_imageurl=%2Fms_tmp%2F&map_web_template=ma
    pserver48_imprimir.html">Imprimir</a>
```

Vemos que invoca a MapServer utilizando la extensión geográfica que tenemos en ese momento, usando el fichero 'mapa' que tenemos también en ese momento y utilizando la plantilla 'mapserver48_imprimir.html'.
Esta plantilla de impresión es mucho más simple y aprovecha al máximo el espacio:



Sin usar plantilla de impresión | Usando plantilla de impresión

Figura 4.46 Diferencias entre usar una plantilla de impresión o no.

4.5.3. Integración con otras aplicaciones

Dada la versatilidad que ofrece el hecho de que la aplicación esté programada utilizando el sencillo lenguaje HTML, y que únicamente sea necesario un navegador convencional, sin plugins ni otros componentes, podremos enlazar con la cartografía de una forma muy sencilla. Por ejemplo, se está usando una base de datos Access para controlar y gestionar los expedientes 902, que son los que se usan para actualizar las alteraciones catastrales.

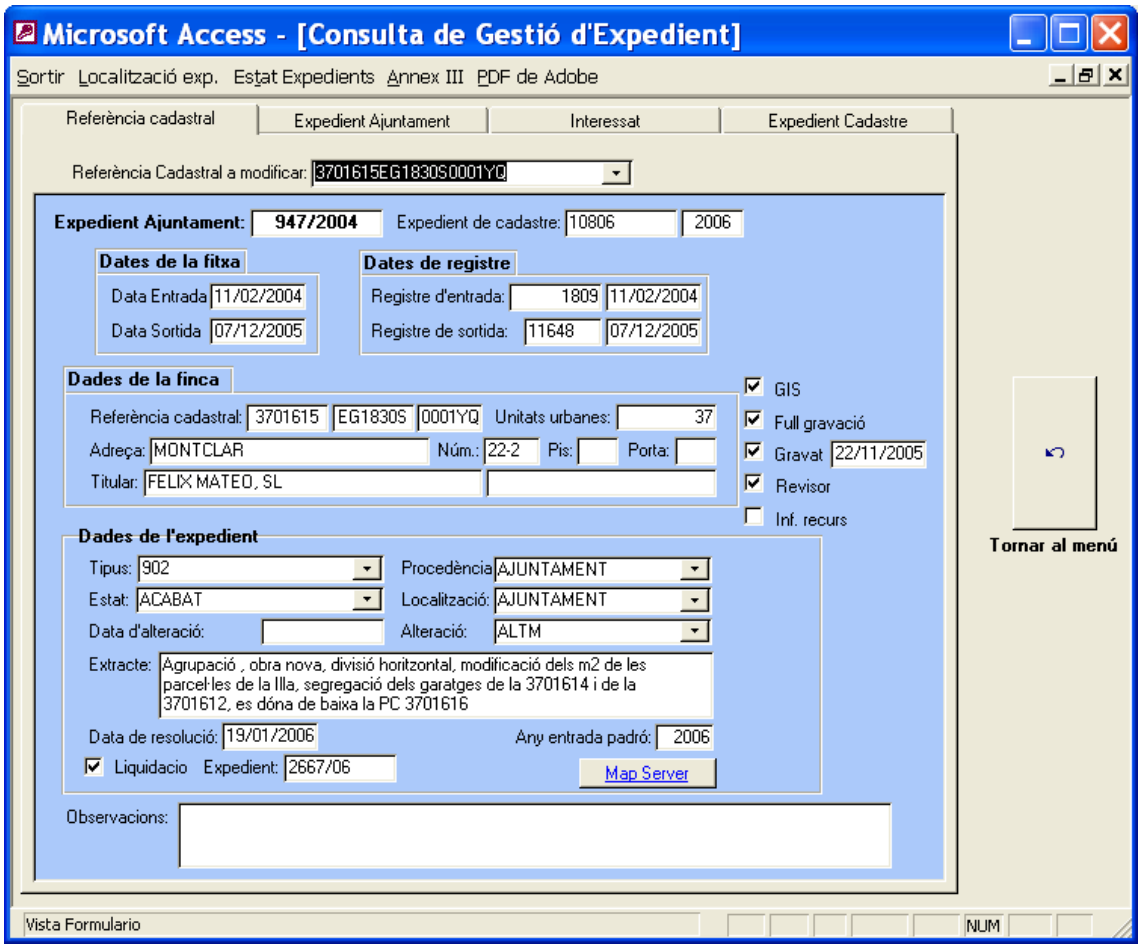


Figura 4.47 Integración con la aplicación de gestión de catastro

Vemos en la zona inferior el botón ‘MapServer’ que lo que hace es mostrarnos la parcela objeto de la modificación catastral usando la aplicación MapServer.
Usando las primeras siete posiciones del campo referencia catastral (Utm_1) creamos automáticamente el link con la función del formulario siguiente:

```
Private Sub Form_Current()
   bMapServer.HyperlinkAddress = "http://gis002/cgi-
bin/mapserv.exe?mode=itemquery&qlayer=PARCELA&" & _
                  "qitem=UTM&qstring=" & Utm_1 &
"&map=C%3A%5Cms4w%5Capps%5Croses%5Ccadastre.map&" & _
                  "mapext=shapes&map_web_template=mapserver48.html"
   DataGIS.Visible = (Estat_EXP = 1)
End Sub
```

Con esto, actualiza el campo ‘Dirección de hipervínculo’ del botón ‘bMapserver’ y al pinchar en él abrimos la plantilla consulta de la parcela:

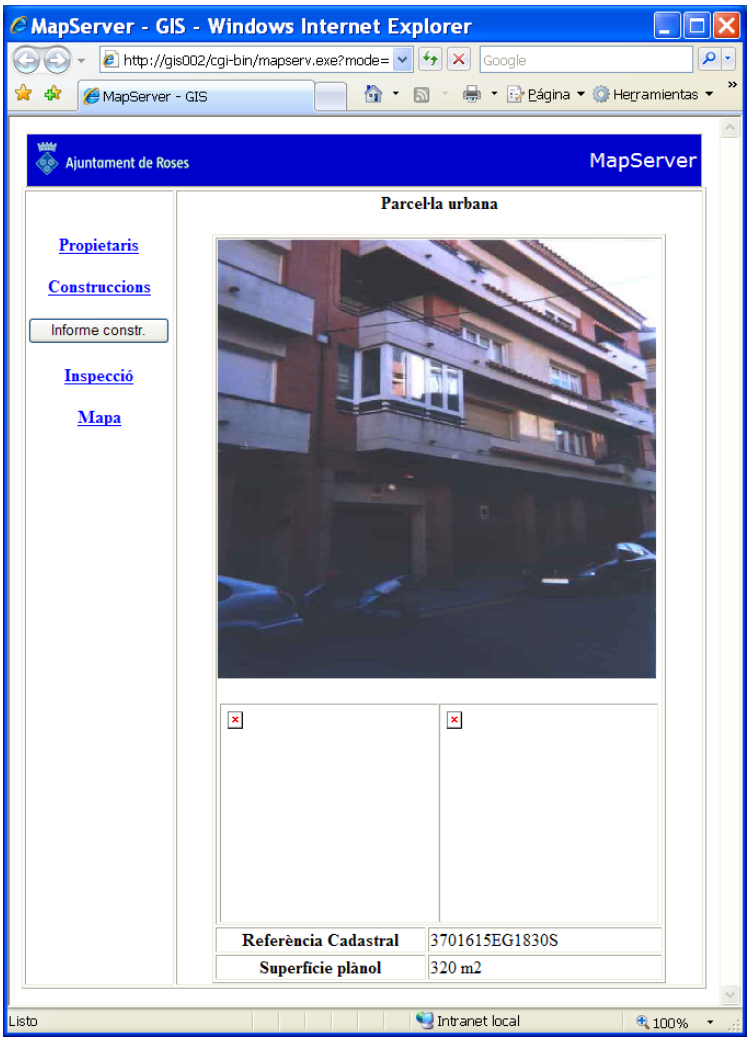


Figura 4.48 Resultado de la consulta.

En todas las bases de datos Access se vincula de la misma manera la cartografía. Otras bases de datos que contienen este vínculo son: vados permanentes, patrimonio, actividades y licencias de obra. En los ayuntamientos generalmente se usan programas más sofisticados que estos que hemos visto de Access, como son el registro de expedientes, gestión de recaudación y habitantes. En el caso del ayuntamiento que estamos tratando, estos programas han sido creados por una empresa externa especializada en este tipo de programas. Estos se han programado usando el lenguaje Speedware, pero no ha sido dificultoso añadir unos botones con los links correspondientes, iguales que en el caso de Access.
Si abrimos el de gestión de expedientes, por ejemplo y seleccionamos uno que tenga relación con el territorio como es una licencia de obras:

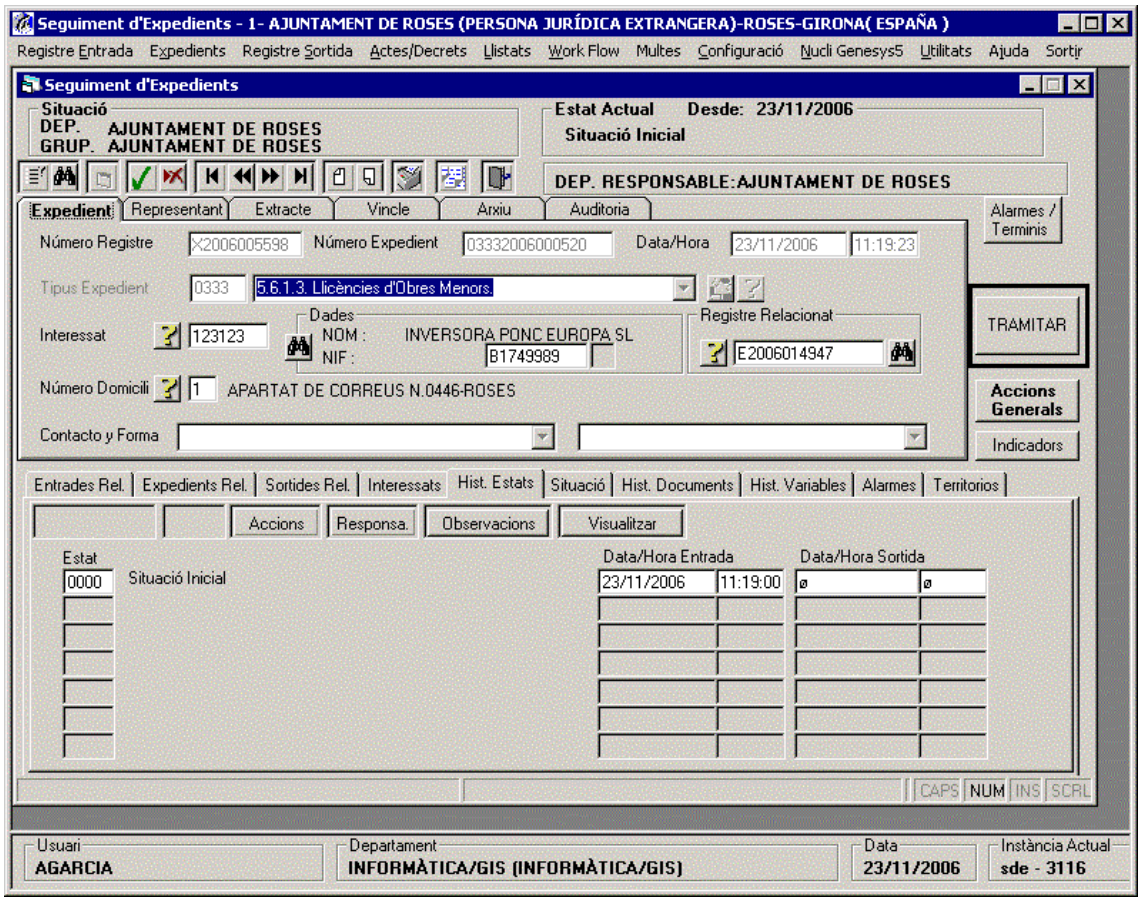


Figura 4.49 Aplicación de gestión de expedientes.

En la pestaña 'Vincle' (vínculo) tenemos el botón 'Territori' que contiene el link que usamos para invocar a MapServer con la referencia catastral de la parcela.

http://gis002/cgi-bin/mapserv.exe?mode=itemquery&qlayer=PARCELA&qitem=UTM&qstring=%225591923%22&map=C%3A%5Cms4w%5Capps%5Croses%5Ccadastre.map&mapext=shapes&map_web_template=mapserver48.html

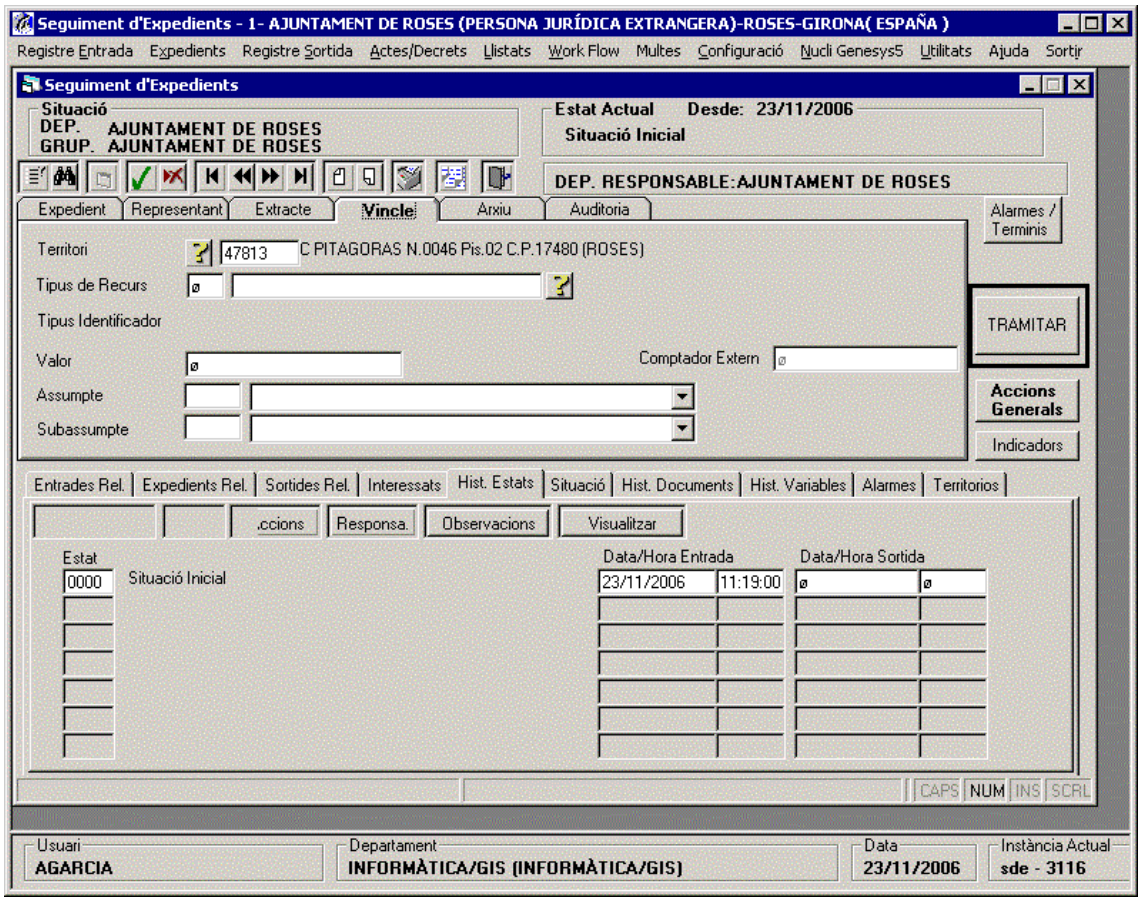


Figura 4.50 Integración con la aplicación de gestión de expedientes.

Y el resultado:

Figura 4.51 Resultado de la consulta.

4.5.4. Edición de entidades puntuales

Esta función añade la capacidad de edición a MapServer. Hasta el momento, los desarrolladores de MapServer no han implementado herramientas de este tipo, pero desde el ayuntamiento se ha considerado conveniente. Hemos visto en capítulos anteriores los temas que se gestionan a través de MapServer, pero no se ha comentado el sistema de actualización de los datos (tanto alfanuméricos como gráficos). En algunos casos, se ha resuelto que sea posible mantener la información desde el mismo departamento. Para dos de estos temas (licencias de obras y actividades) se ha añadido a la plantilla principal la capacidad de crear nuevos puntos (ya sean licencias o actividades) y la de editar los atributos de los ficheros shp.
Para lograr esta edición, se ha usado las librerías creadas por Ross Pickard llamadas "**ArcView Shapefile OCX**". Estas librerías permiten entre otras cosas:

1. Crear ficheros shp
2. Añadir registros a un fichero shp existente
3. Alterar la estructura y editar registros (de atributos y vértices) de un shp
4. Buscar registros basándose en consultas simples a la base de datos

El código se puede obtener en la página de ESRI:
http://arcscripts.esri.com/details.asp?dbid=11810.
El fichero zip contiene dos versiones de librerías: las ocx y las dll, según se quiera trabajar desde VisualBasic o fuera de él (nuestro caso).
Una vez descargado, será necesario registrar las librerías en nuestro sistema usando el comando Regsvr32:

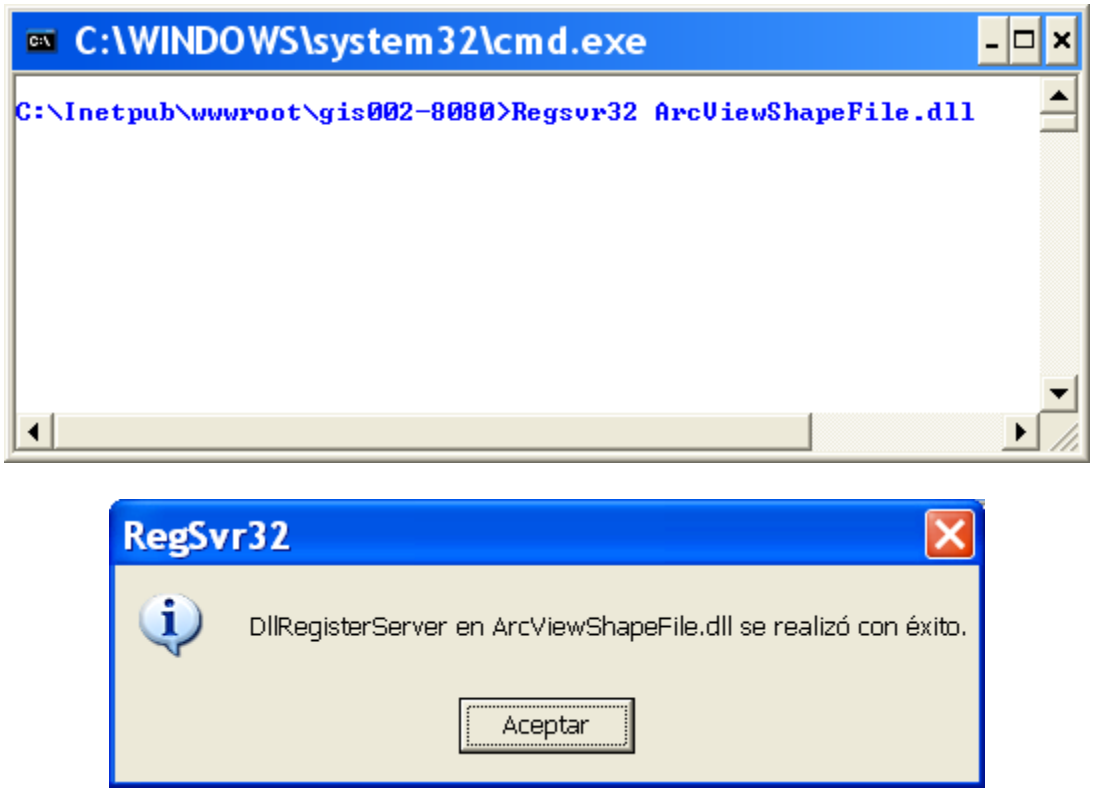


Figura 4.52 Registrando la librería.

Ahora ya podemos utilizar los comandos que nos ofrecen estos elementos.
Para el caso de las licencias de obras, se ha añadido a la plantilla dos campos de texto para introducir licencias nuevas. El funcionamiento es el siguiente:

1. Se pincha en el mapa el lugar donde se quiere añadir una licencia nueva.
2. Se rellenan los campos con el número de expediente y el estado.
3. Se pincha en el botón 'ok'
4. Se actualiza la página o se hace un zoom para ver los resultados.

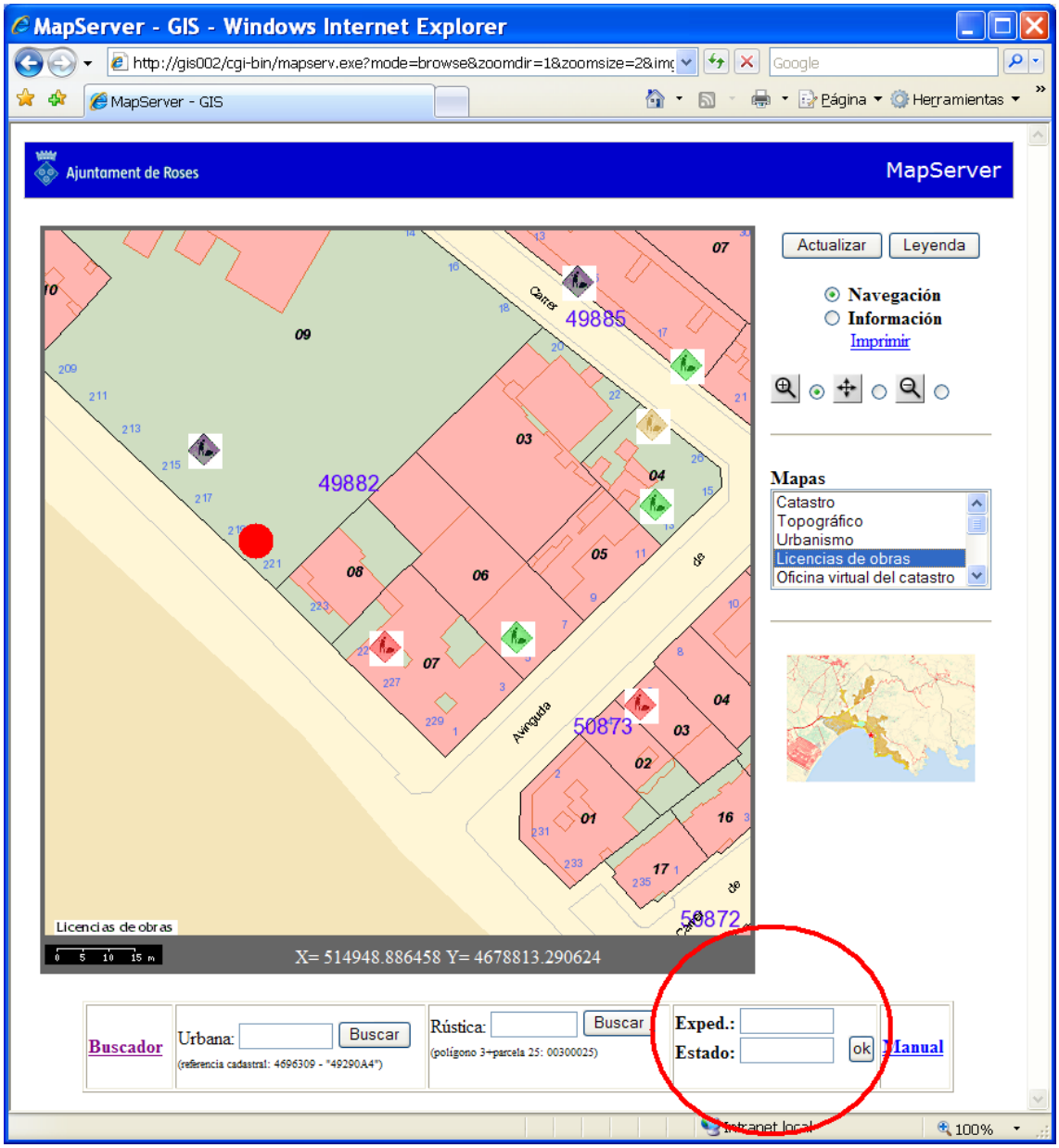


Figura 4.53 Campos de entrada de datos para la inserción de licencias de obras.

Vamos a ver el código de la página principal:

```
122  <FORM ACTION="http://gis002:8080/gis002-8080/inserta_punt.asp"
     METHOD="post">
123  <b>Exped.:</b>
```

```
124  <input name="expedient" size="9" maxlength="9">
125  <b>Estado:</b>
126 <input name="estat" size="9">
127 <input type="hidden" name="mapx" value="[mapx]">
128 <input type="hidden" name="mapy" value="[mapy]">
129 <input name="Submit" type="Submit" value="ok">
130  </FORM> </td>
```

Vemos que el formulario envía los datos que rellenamos y los valores de las coordenadas del último punto pinchado a la aplicación 'inserta_punt.asp', que es:

```
01  <html><head><title> </title></head><body>
02  <%
03  Dim EXPEDIENT,ESTAT,coordx,coordy
04  Dim MyShape
05  Set MyShape =
    Server.CreateObject("ArcViewShapeFileDLL.ShapeFiles")
06  Dim NewVert
07  Set NewVert = Server.CreateObject("ArcViewShapeFileDLL.Vertice")
08  Dim ShapeField
09  Set ShapeField =
    Server.CreateObject("ArcViewShapeFileDLL.ShapeField")
10  EXPEDIENT = request.form("EXPEDIENT")
11  ESTAT = request.form("ESTAT")
12  mapx = request.form("mapx")
13  mapy = request.form("mapy")
14  mapx= mapx/1000000
15  mapy= mapy/1000000
16       MyShape.OpenShape "C:\ms4w\apps\tfc\shapefiles\obras.shp",2,1
17       MyShape.ShapeFields("EXPEDIENT").Value = EXPEDIENT
18       MyShape.ShapeFields("ESTAT").Value = ESTAT
19       Set NewVert = MyShape.Vertices.AddVertice((mapx),(mapy))
20       MyShape.CreateShape
21       Set NewField = Nothing
22       Set NewVert = Nothing
23       Set ShapeOut = Nothing
24  %>
25  <script language="JavaScript" type="text/javascript">
26  javascript:history.back()
27  </script>
28  </body></html>
```

El código es un documento html que contiene una parte de código asp, que es el que inserta el punto, y una parte javascript que nos devuelve a la página anterior.
La parte asp es la realmente interesante. Entre las líneas 03 y 09 se definen las variables que utilizaremos. Entra la 10 y la 13 se asignan a las variables los valores que hemos enviado a través del formulario de la página principal. Con la 14 y 15 dividimos los valores de las coordenadas que provienen de MapServer por 10^6 puesto que no tiene en cuenta el punto que separa los decimales.

Con la línea 16 abrimos el fichero shape indicando dos parámetros: El modo de apertura y el tipo de datos shp. Los valores posibles son:

Modo de apertura	Tipo de datos
0: abrir un shp existente	1: puntos
1: crear un nuevo shp	3: polilíneas
2: añadir registros a un shp	5: polígonos
3: editar un shp	

Existen diez tipos más de datos shp, pero estos tres son los más comunes.
Con las líneas 17 y 18 asignamos los valores de EXPEDIENT y ESTAT al nuevo registro, y con la 19 le añadimos un vértice con las coordenadas correctas.
La línea 20 hace efectivos los cambios anteriores, y las siguientes hasta la 23 vacían las variables.
La parte JavaScript, que comprende las líneas 25 a 27 lo único que hace es devolvernos al mapa.
Así, si pinchamos en la zona del círculo rojo, e introducimos los datos de expediente y estado y pinchamos en 'ok':

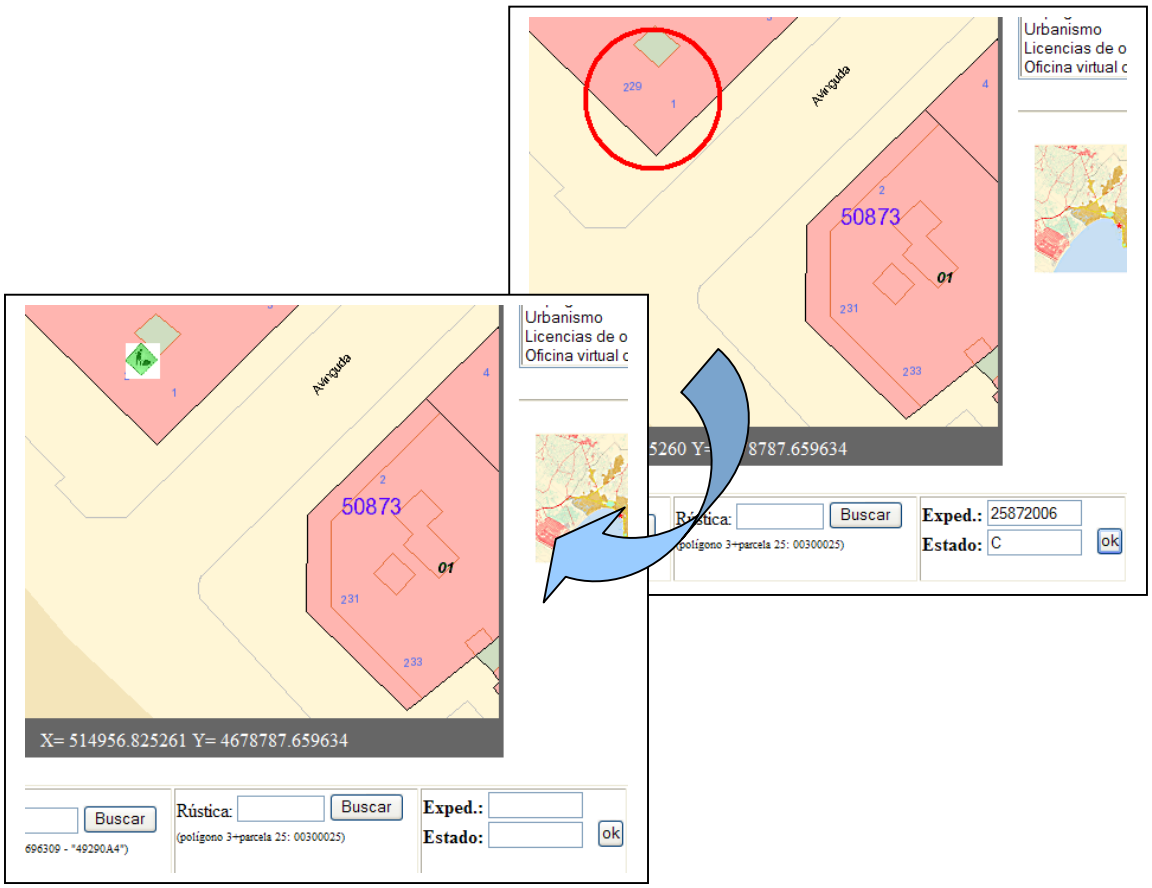


Figura 4.54 Proceso de inserción de la licencia.

Nótese que el símbolo con el que ha dibujado el punto es de color verde, ya que en el campo 'Estado' se ha introducido una C de concedida (en el capítulo anterior se explica este aspecto).

Si ahora pedimos información del punto que acabamos de crear veremos:

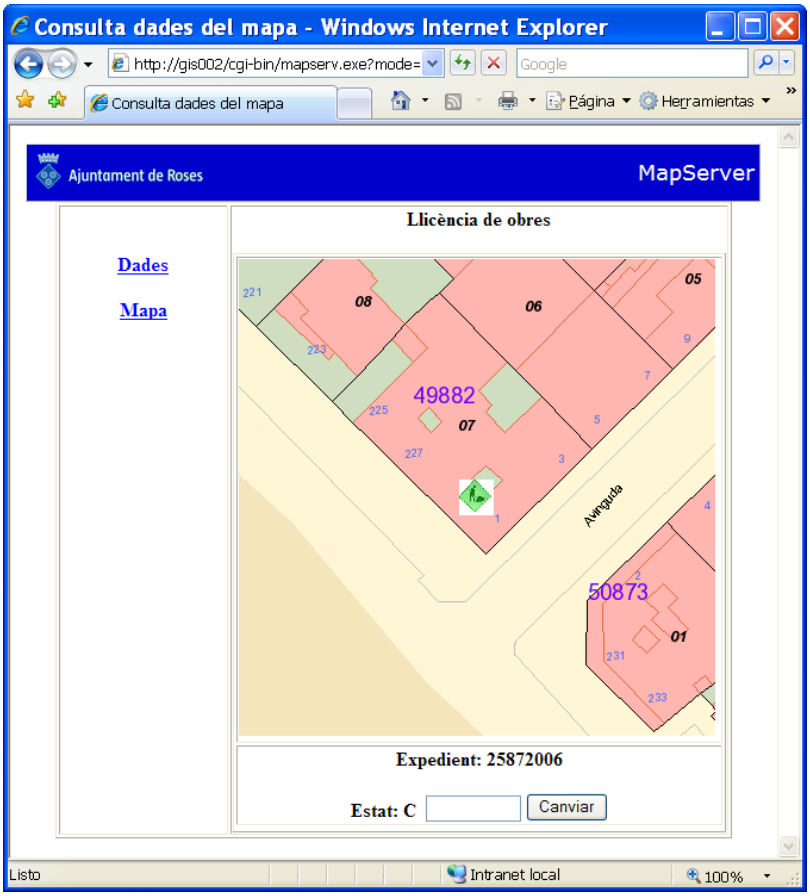


Figura 4.55 Resultado de la consulta.

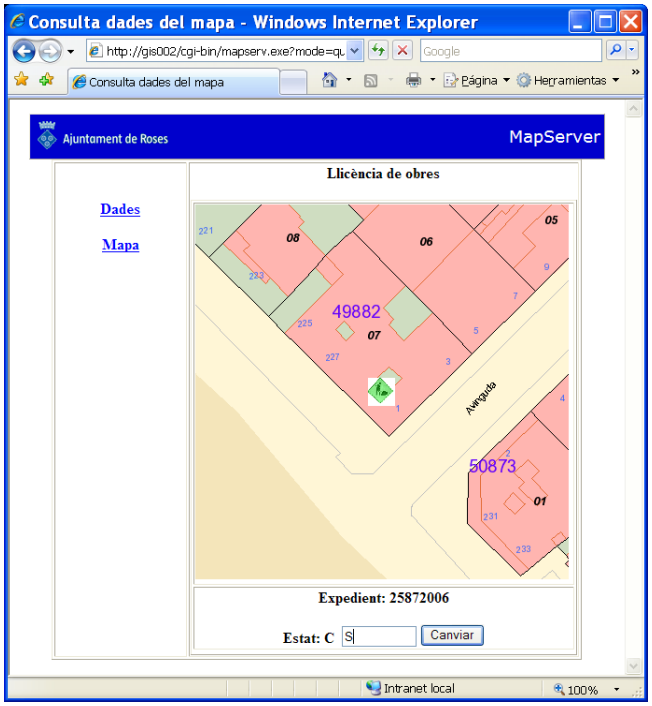
Nos informa del expediente y del estado, y nos ofrece la posibilidad de cambiar este último. Esta capacidad se ha pensado para pasar de un estado a otro los expedientes de licencias. En principio, y para que no se produzcan 'accidentes' se ha optado por no permitir la eliminación física de estos elementos. Lo que se propone para borrarlos visualmente es introducir un estado que no esté codificado (B por ejemplo), y así no será representado.
El código de esta parte de la plantilla de consulta es:

```
30  <th><p>Expedient: [EXPEDIENT]</p>
31  <p>
32  <FORM ACTION="http://gis002:8080/gis002-8080/canvia_punt.asp"
    METHOD="post">
33  Estat: [ESTAT]<input type="hidden" name="EXPEDIENT"
    value="[EXPEDIENT]">
34  <input name="ESTAT" size="9" maxlength="9">
35  <input name="Submit" type="Submit" value="Canviar">
36  </FORM> </p></th>
```

Es un formulario muy similar al visto con anterioridad, que envía los parámetros EXPEDIENT y el valor nuevo que designemos para ESTAT a la aplicación 'canvia_punt.asp'. Veámosla:

```
01  <html><head><title> </title></head><body>
02  <%
03  Dim MyShape
04  Set MyShape =
    Server.CreateObject("ArcViewShapeFileDLL.ShapeFiles")
05  Dim ShapeField
06  Set ShapeField =
    Server.CreateObject("ArcViewShapeFileDLL.ShapeField")
07  EXPEDIENT = request.form("EXPEDIENT")
08  ESTAT = request.form("ESTAT")
09  MyShape.OpenShape "C:\ms4w\apps\tfc\shapefiles\obras.shp",3,1
10  MyShape.FindFirst "EXPEDIENT = " + EXPEDIENT
11  If MyShape.NoMatch = False Then
12  MyShape.ShapeFields("ESTAT").Value = ESTAT
13  MyShape.ModifyShape
14  End If
15  %>
16  <script language="JavaScript" type="text/javascript">
17  javascript:history.back()
18  </script>
19  </body></html>
```

Como en el caso anterior primero definimos y asignamos variables y abrimos el fichero en modo edición (3). En la línea 10 usamos una herramienta que busca el primer registro cuyo valor del campo EXPEDIENT sea igual al número de expediente del punto seleccionado. Establecemos la condición en la línea 11 que no haga nada hasta encontrar el registro deseado, y en caso de encontrarlo cambiamos el valor del campo ESTAT por el nuevo valor que designamos en la plantilla de consulta. Para finalizar volvemos a la página del mapa.
Vamos a cambiar el estado del punto introducido en el caso anterior. Primero pedimos información:



Cambiamos el valor a ‘S’ de suspendida y pinchamos en ‘Canviar’. Volvemos al mapa principal y vemos los cambios realizados.

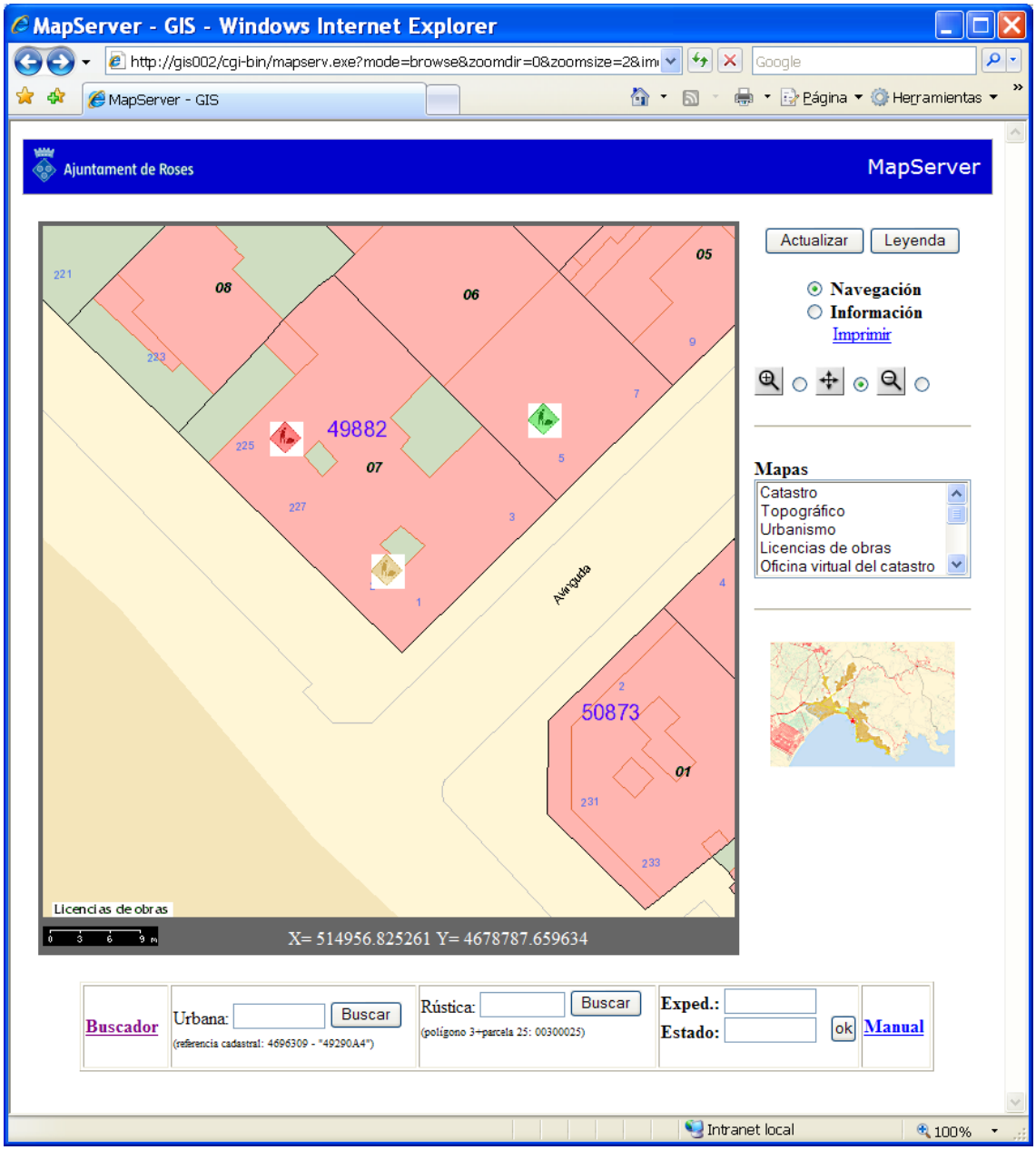


Figura 4.57 Resultado al cambiar de estado una licencia.