Todo lo que siempre quiso hacer
pero no había podido...

MAPSERVER

FOR

DUMMIES®

Primera y unica edición

No más dolores de cabeza.

Guía súper útil y sin enredos de algún tipo.

Andres Herrera

Advertencia:

Este documento no pretende ser una guía de estudio, una guía de bolsillo, ni mucho menos un documento al cual le rindas devoción como material bibliográfico obligatorio de consulta, aquí tan solo pretendo explicar por medio de una serie de sencillos pasos la forma mas fácil de poner en marcha un servidor de datos espaciales con Mapserver, phpMapscript y PostgreSQL.

Esta es la primera y única versión de este documento, el autor no se hace responsable sobre el uso indebido de la información aquí consignada.
La iconografía usada pertenece a sus correspondientes autores.

Introducción

En esta guía debido a su contenido informal, esta concebida bajo el principio de KISS (Keep It Simple Stupid[1]), así que no entraremos en formalismos y en definiciones complejas de cada uno de los componentes de software y las metodologías a utilizar a lo largo del desarrollo, por otro lado; se requiere una serie de conocimientos básicos lograr un completo entendimiento, de lo contrario es recomendable consultar con "San Google[2]" amo y señor de nosotros los mortales.
Esta guía se desarrollara tratando de abordar el mayor número interrogantes que se presentan a lo largo del aprendizaje de este tipo de tecnologías, junto a esta guía se entregan **14** ejemplos básicos, algunas herramientas y utilerías.
Algunas de mi autoría y otras los cuales he recopilado de diferentes fuentes, realizando ligeros cambios. A manera de "PLUS" introduciremos algunas utilidades disponibles en el paquete FWTools[3], las cuales nos permitirán conocer otras formas no tradicionales de realizar las cosas y otras cosillas mas que se entregan a lo largo de la guía como TIPS.

Herramientas

En esta guía marcharemos de la mano de herramientas de código abierto, de libre acceso y disponibles en la red, en particular, usaremos un paquete llamado (MS4W) el cual instala un ambiente pre-configurado de servidor Web, en este caso Apache además crea una completa instalación de Php5, MapServer, MapScript y otra serie de componentes útiles, igualmente para el almacenamiento de datos espaciales usaremos el motor PostgreSQL con la extensión espacial PostGIS,

[1] Principio KISS, http://es.wikipedia.org/wiki/Principio_KISS
[2] Google, http://www.google.com
[3] FWTools, http://fwtools.maptools.org/

Instalación

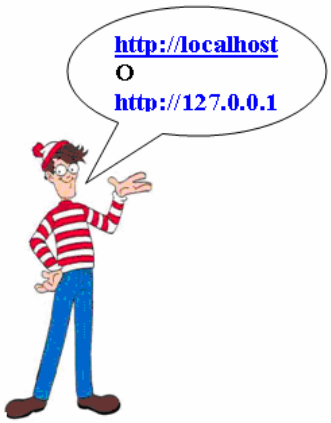
MS4W (MapServer para Windows): descargamos el instalador de la URL: http://www.maptools.org/dl/ms4w/ms4w-2.2.7-setup.exe ,la instalación de este paquete se resume a unos simples clicks en otras palabras "Siguiente -> Siguiente".

Nota de instalación: instalamos en el directorio raíz (C: o D:) en la ruta **C:\ms4w**

PostgeSQL / PostGIS: instalaremos motor de bases de datos PostgreSQL con la extensión de soporte de datos espaciales PostGIS, descargando el instalador de la URL: ftp7.us.postgresql.org/pub/postgresql//binary/v8.3.1/win32/postgresql-8.3.1-1.zip , la instalación igualmente se resume en unos simples clicks, debemos tener un particular cuidado al realizar la instalación, con los datos proporcionados para el nombre de usuario y la contraseña que le asignaremos a la cuenta de usuario Postgres puesto que estas serán las que posteriormente usaremos para acceder a nuestro repositorio de datos, en la versión actual postgresql 8.3.1 la extensión de PostGIS ya hace parte del release oficial. Simplemente la debemos activar dirigiéndonos a **Inicio -> Programas -> PostgreSQL 8,3 -> Application Builder Stack**. Seleccionando de la lista La última versión de PostGIS y haciendo clic en "Siguiente". Una vez instalada esta aparecerá en el marco del "Spatial Extensions"

FWTools : la instalación es igualmente sencilla, descargamos el instalador de la siguiente URL: http://home.gdal.org/fwtools/FWTools210.exe y seguir la instalación guiada.

Configuración



Comprobamos que la instalación de MapServer se ha realizado con éxito, digitando en el navegador. (http://localhost o http://127.0.0.1) esto nos debe dirigir a la pantalla de bienvenida la cual nos indica que esta corriendo el servidor Apache y confirmándonos la correcta instalación de MapServer y sus componentes, de igual forma comprobamos que efectivamente mapserver este funcionando como CGI, digitando en el navegador (http://localhost/cgi-bin/mapserv.exe), mapserver nos debería responder de la siguiente forma:

No query information to decode. QUERY_STRING is set, but empty.

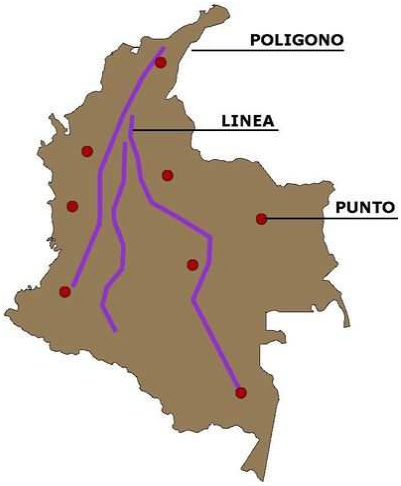
Una vez instalada la base de datos PostgreSQL checkeamos su correcta instalación, mediante pgAdmin III[4] verificando la existencia de las tablas con información espacial:(**geometry_columns, spatial_ref_sys),** además podemos realizar un pequeño script el cual nos

[4] PgAdmin III, http://www.pgadmin.org/

permitirá checkear la conexión de postgres con php, hay que aclarar que se debe establecer los parámetros correctos de conexión, por ejemplo: con animo de probar el funcionamiento se ha creado una base de datos de nombre (prueba), y se han establecidos los parámetros de conexión user y password, y en su defecto el host y el puerto cuando hallamos realizado instalaciones personalizadas.

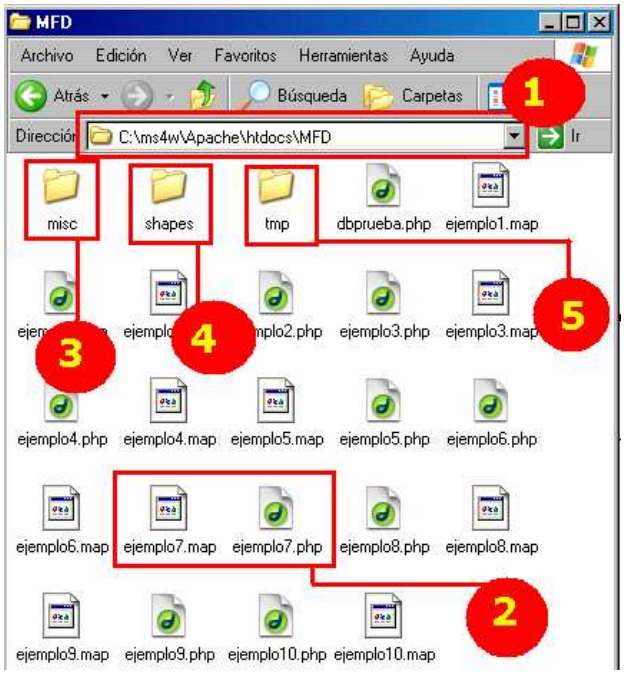
```
//Guardar como : dbprueba.php
<?php
        $connection = "host=localhost port=5432 dbname=prueba user=postgres password=1234567";
        $conecta=pg_connect($connection);

        if(!$conecta)
        {
                die("No se pude establecer la conexión con la base de datos.");
        }
else
        {
                print("Conexión establecida con éxito !!");
        }
?>
```

Lo básico



Referente a la información espacial; necesitamos 3 coberturas de tipo vectorial (punto, línea, polígono), las cuales están referenciadas en el sistema de referencia WGS84, esta información la he incluido en al archivo que acompaña esta guía (**MFD.zip**) el cual debe ser descomprimido en el fólder (**C:\ms4w\Apache\htdocs**).

Para simplicidad de los ejemplos, definiremos una estructura de directorios de trabajo.



(1) Ubicación donde descomprimir archivo MFD.zip
(2) Archivos .php y .map correspondientes a los ejemplos.
(3) Fólder donde ubicaremos símbolos, fuentes, utilidades y otra serie de archivos herramientas.
(4) Información de ejemplo en formato ESRI shp.
(5) Fólder temporal, aquí una vez mapserver empiece a realizar su función, generara muchos archivos.

Mapas Estáticos

Una vez instalado ms4w, y MapServer corriendo en nuestra maquina local, desarrollaremos nuestra primera aproximación a la generación de mapas con mapserver y phpmapscript, se ha consolidado en el Ejemplo1[5] en donde elaboraremos un mapa estático donde representaremos las 3 entidades básicas (punto, línea, polígono) que hemos definido anteriormente.

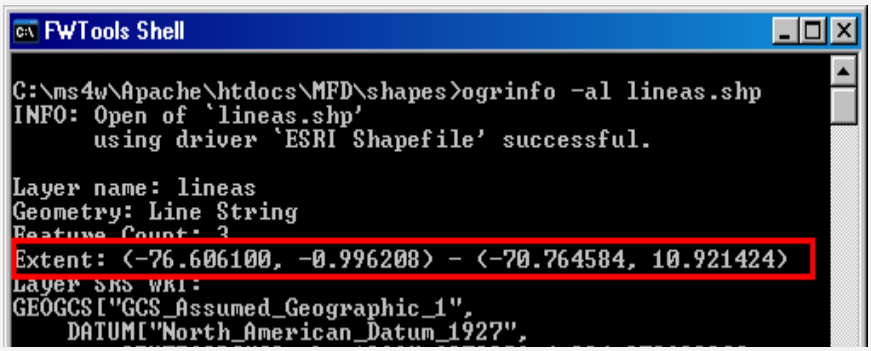
Echémosle un vistazo al archivo ejemplo1.map, este coincide con la estructura de mapfile propia de mapserver. Puntos importantes a destacar son:

Bloque: **EXTENT -88 -5 -62 13** corresponde a las coordenadas **[minx], [miny], [maxx], [maxy]** de nuestra información espacial, podemos visualizar estos limites, apoyándonos en herramientas como son Quantum GIS, GRASS, gvSIG, o cualquier tipo de software que nos permita visualizar información geográfica en formato ESRI shape.

.

Ventana de Propiedades de la capa, software Quantum GIS

En la consola FWTools; ubicándonos en el path donde se encuentra la información espacial, digitamos: Orginfo **–al archivo.shp**



[5] Ver archivo MFD.zip que acompaña esta guía.

Bloque:

```
SHAPEPATH "shapes/"
FONTSET "misc/fonts/fonts.txt"
SYMBOLSET "misc/symbols/symbols.sym"
```

Aquí definimos la ubicación de nuestra información especial (shapes), el archivo de fuentes y la librería de símbolos.

Bloque:

```
WEB
        IMAGEPATH "C:/ms4w/Apache/htdocs/MFD/tmp/"
        IMAGEURL "tmp/"
END
```

Aquí definimos el path temporal donde mapserver renderizara las imágenes.

Bloque:

```
UNITS dd
```

Aquí definimos la unidad de medida con la que nuestra información espacial esta representada.

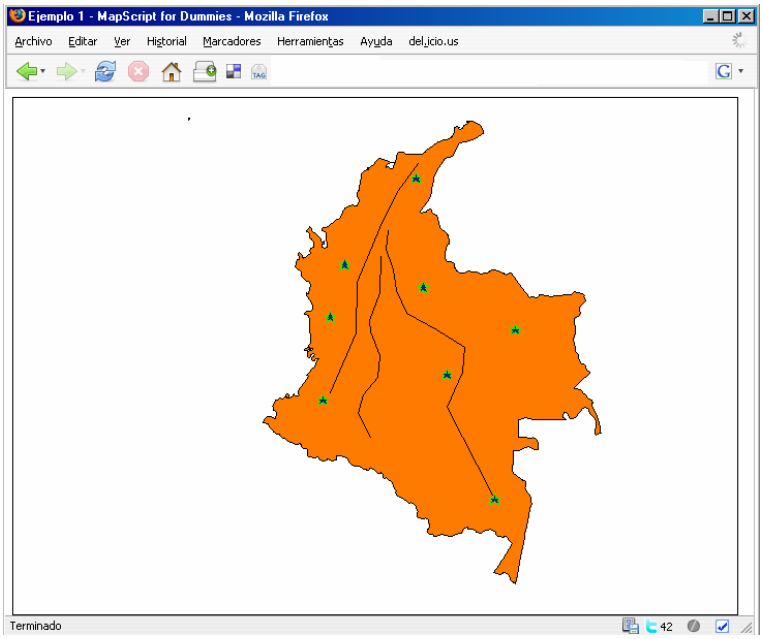
Este primer ejemplo contiene dos archivos importantes Ejemplo1.map y Ejemplo1.php el primero correspondiente al MAPFILE donde se definen las propiedades del mapa a generar y cada capa (LAYER) de información espacial a ser representado, el segundo;

```
LAYER
        NAME "Poligonos"
        STATUS ON
        DATA "poligono.shp"
        TYPE POLYGON
        CLASS
                STYLE
                        COLOR 255 123 0
                        OUTLINECOLOR 0 0 0
                END
        END
END
```

```
<?php
        if (!extension_loaded("MapScript"))
        {
        dl('php_mapscript.'.PHP_SHLIB_SUFFIX);
        }

        $mapObject = ms_newMapObj("ejemplo1.map");
        $mapImage = $mapObject->draw();
        $urlImage = $mapImage->saveWebImage();
?>
<img src="<?php echo $urlImage; ?>" border="1" >
```

ejemplo1.php, corresponde a un script en php que usa la Liberia mapscript para crear el objeto del mapa e invocar el mapserver para que genere la imagen, una vez renderizada la imagen es llamada desde un bloque de código en HTML y desplegada en cualquier navegador, el resultado al digitar la url:

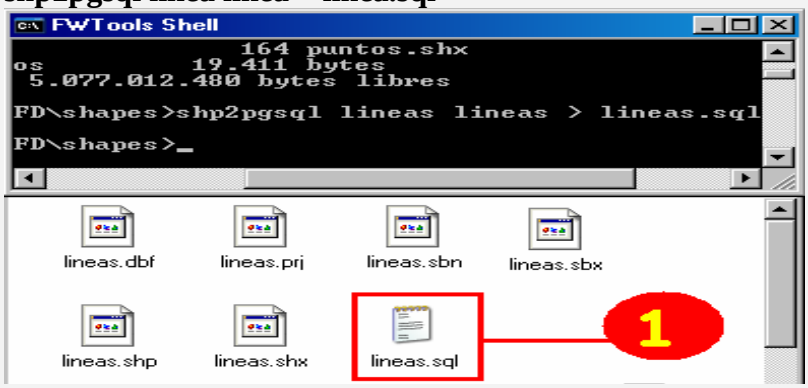
http://localhost/MFD/ejemplo1.php



No más shapefiles

En este ejemplo no usaremos más los archivos shapes, y crearemos un repositorio remoto (base de datos), el cual contendrá la información vectorial, como primer paso convertiremos nuestros archivos fuente ESRI shp a SQL, siguiendo las siguientes recomendaciones.
Algunos aspectos a tener en cuenta cuando trabajemos con archivos:
No usar espacios nombres de archivo, No usar caracteres espaciales, Usar nombres cortos y de fácil recordación.

En la consola FWTools; ubicándonos en el path donde se encuentra la información espacial, digitamos siguiendo la siguiente sintaxis: **shp2pgsql [shp] [shx] > salida.sql ,** sin indicar la extensión de los archivos de entrada, por ejemplo si nuestra misión es convertir el shape de **linea.shp**, digitaríamos en la línea de comandos: **shp2pgsql linea linea > linea.sql**



Como resultado obtenemos una salida en formato .SQL, archivo (**1**) que próximamente introduciremos en nuestra base de datos.

¡Trabajo Tedioso!

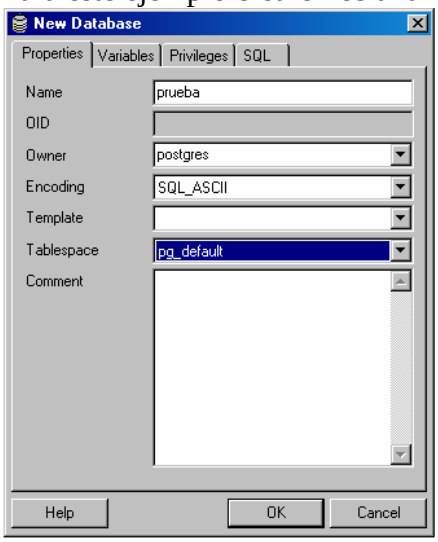
Cuando tenemos una gran cantidad de coberturas en formato ESRI Shape, las cuales deseamos insertar en nuestra base de datos, resulta ser una tarea bastante tediosa convertir una a una, escribiendo una sencilla rutina en batch podemos ahorrarnos unos cuantos minutos de nuestras vidas y convertir grandes volúmenes de información usando del comando shp2pgsql para luego ser insertadas en nuestra base de datos geográfica.

Crear un archivo **all_shapes.bat** y adicionar las siguientes líneas:

```
@echo *************************
@echo all_shapes to pgsql v0.1
@echo .
@echo fandresherrera(at)hotmail(dot)com
@echo .
@echo *************************
@echo off
color 20
for %%x in (*.shp) do shp2pgsql %%~nx %%~nx > %%~nx.sql
pause
exit
```

Creando el repositorio

Para este ejemplo crearemos una nueva base de datos en PostgreSQL, la cual llamaremos (**prueba**)



Los pasos para crear esta;

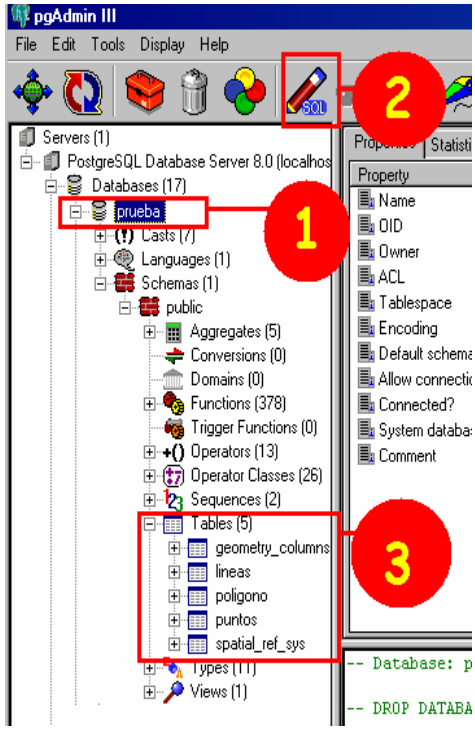
1) Iniciamos pgAdmin III[6]

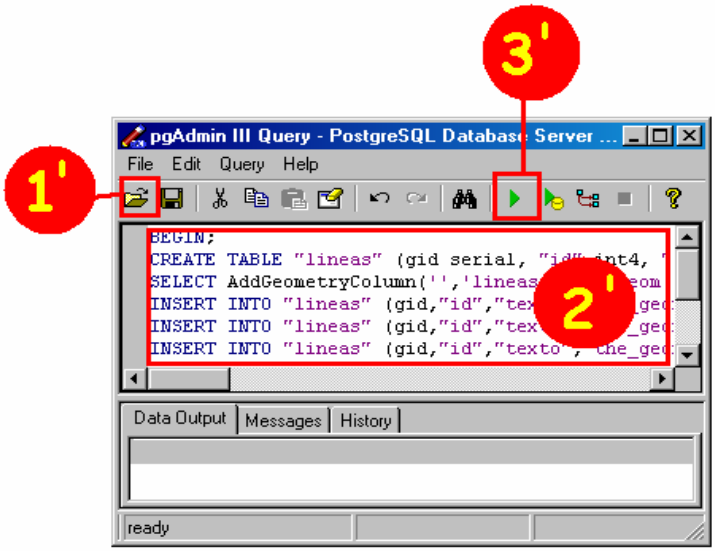
2) Nos conectamos al servidor de PostgreSQL(indicamos usuario y contraseña).

3) Vamos a **Edit** -> **New Database**

4) En la ventana de **New Database,** configuramos la base de datos escogiendo encoding **SQL_ASCII.**

Una vez creada la base de datos procedemos a volcar el contenido de cada una de nuestras geometrías convertidas a SQL, **(1)** primero es conveniente seleccionar la base de datos creada, seguidamente picando el icono de **(2)** (Execute arbitrary SQL query's). La ventana emergente nos permitirá seleccionar el archivo que deseamos volcar a través del icono (**1'**) (open file), aquí seleccionamos el archivo .SQL el cual deseamos introducir en la base de datos, en (**2'**) pre-visualizamos el archivo correspondiente a la geometría a introducir… para la el **ejemplo2** introduciremos los archivos correspondientes a **poligonos.sql , lineas.sql , puntos.sql** . Con (**3'**) ejecutamos la sentencia SQL. Verificamos la correcta creación de las tablas en la base de datos en (**3**).





Posibles problemas que se pueden presentar en este paso:

Que la información alfanumérica correspondiente a los shapes tengan un encoding extraño y una vez convertido a SQL presente problemas al tratar de ingresarlo a la base de datos; esto se soluciona escogiendo el encoding correcto para la base de datos, o normalizando la información alfanumérica realizar la conversión nuevamente en SQL.

[6] **NOTA**: Iconos y Estilo de ventanas, suelen cambiar de versión en versión de pgAdminIII

¡Manos a la obra ¡

Nuestro ejemplo2, corresponde a la generación de un mapa estático en mapserver y mapscript, similar al anterior, salvo una serie de ligeros cambios en la estructura MAP contenidos en el ejemplo2.map, cambios que nos permitirán conectarnos a la base de datos previamente creada.
Echémosle un vistazo al archivo ejemplo2.map, este coincide con la estructura de mapfile propia de mapserver. Puntos importantes en este archivo a destacar son:

Bloque:

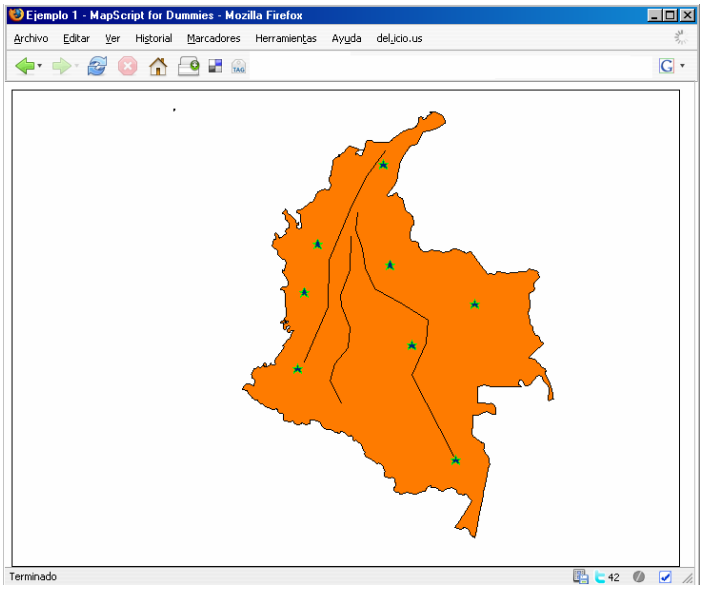
```
LAYER
        CONNECTIONTYPE postgis
        NAME "Poligonos"
```

Aquí definimos el tipo de conexión; postgis para nuestro ejemplo

Bloque:

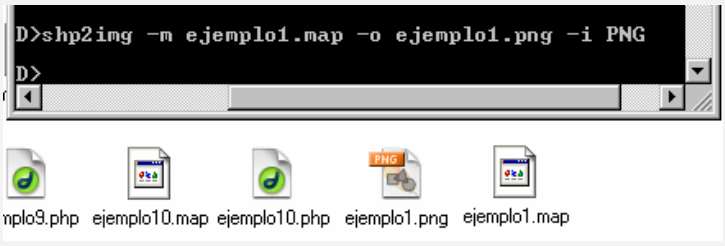
```
CONNECTION "user=postgres password=1234567 dbname=prueba host=localhost"
DATA "the_geom FROM poligono as poligono using unique gid using SRID=-1"
```

Para realizar la conexión con la base de datos, indicamos los parámetros correspondientes, y realizamos un simple query (SELECT) a la tabla deseada, trayendo el campo de **the_geom** el cual correspnde a la geometría en formato **WKT**[7].



Para finalizar, el resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo2.php** como vemos el resultado corresponde al despliegue de un mapa con las entidades básicas (punto, línea, polígono), las cuales se encuentran en la base de datos postgresSQL.

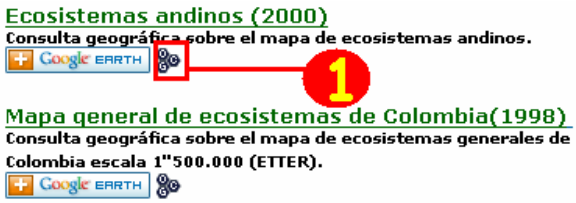
En la consola FWTools; ubicándonos en el path en donde se encuentran nuestros archivos MAP, digitamos la siguiente línea: **shp2img -m ejemplo1.map -o ejemplo1.png -i PNG**, Esta es otra forma de generar un mapa estático partiendo de la estructura .map.

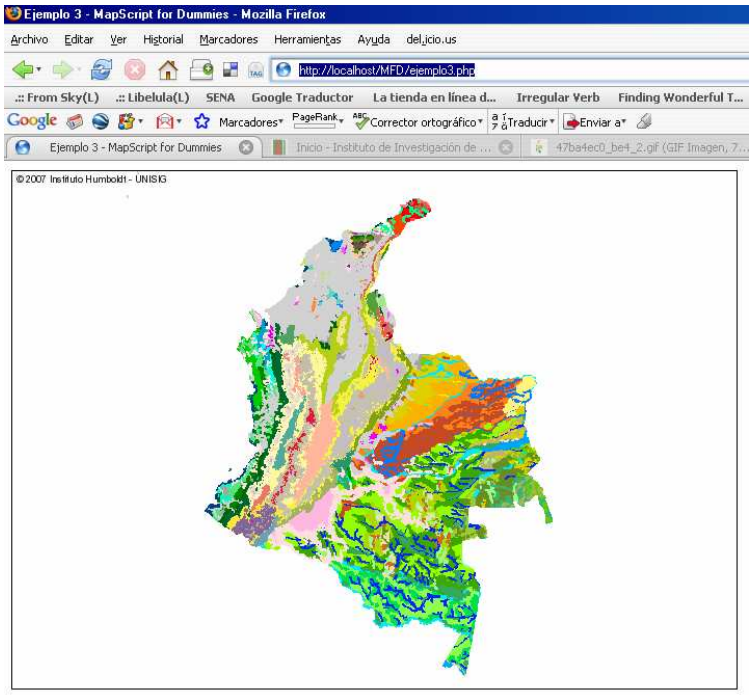


[7]Well Know Text - http://en.wikipedia.org/wiki/Well-known_text

Accediendo a Repositorios distribuidos en WMS

En el ejemplo3 vamos a conectarnos a un repositorio OGC WMS, para este ejemplo usaremos el recurso proporcionado por el instituto humboldt[8], el cual nos proporciona 3 tipos de mapas **(1)** diferentes servidos a través de WMS a los cuales tenemos libre acceso.





En ejemplo3.map observemos los cambios, para poder realizar la respectiva conexión WMS. Puntos importantes en este archivo a destacar son:

Bloque:

```
PROJECTION
        "init=epsg:4326"
END
```

Forzó la proyección. Según humboldt; el mapa esta referenciado en WGS84, para esto consultamos el código EPSG[9] equivalente a WGS, este corresponde al **4326**.

Bloque:

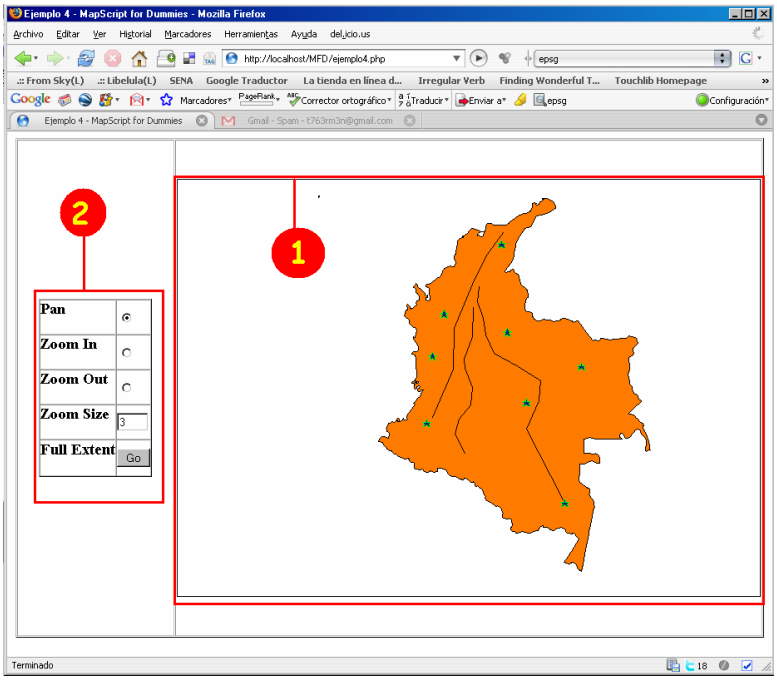
```
LAYER
        NAME "Ecosistemas"
        TYPE RASTER
        STATUS ON
        CONNECTION
"http://www.humboldt.org.co/unisig/ogc/wxs?service=wms&servicename=Ecosistemas_de_Colombia&request=getcapabilities"
        CONNECTIONTYPE WMS
        METADATA
            "wms_srs"               "EPSG:4326"
            "wms_name"              "Ecosistemas"
            "wms_server_version"    "1.1.1"
            "wms_format"            "image/png"
        END
END
```

Como podemos ver clara mente la estructura del mapfile se conserva, la definición de layer cambia en su contenido, puesto que WMS nos entrega una capa renderizada en formato raster, debemos definir esta en **TYPE RASTER**, igualmente los parámetros de conexión y el tipo de conexión cambia drásticamente con respecto a ejemplos anteriores. Un nuevo bloque de atributos **METADATA** se define, aquí definiremos parámetros propios de la conexión, ***wms_rsr***: tipo e proyección, ***wms_name***: nombre de la capa que deseamos obtener, ***wms_server_version***: la versión del servidor remoto wms del cual estamos obteniendo la capa, ***wms_format*** : formato en el que deseamos que el servidor nos entregue la imagen renderizada, para finalizar, el resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo3.php**

[8] http://www.humboldt.org.co/humboldt/
[9] http://www.epsg.org/CurrentDB.html

Añadiendo Controles

En el ejemplo4 añadiremos una serie de controles básicos (**zoom in , zoom out, pan**), con los cuales nuestro mapa ya nunca más será un aburrido mapa estático.
Nota: para la comprensión de este archivo es necesario conocimientos básicos de php y html (manejo y envió de formularios).



Para nuestro ejemplo, estructura del archivo .map se conserva (**ejemplo4.map**) igual a los anteriores, los cambios se han realizado sobre el archivo **ejemplo4.php**, donde mediante phpMapScript e interacciones de formularios en HTML, podemos lograr el efecto de zoom.
El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo4.php**
Donde **(1)** corresponde al mapa renderizado, **(2)** a las herramientas compuestas por elementos de formulario HTML (radiobutton), analizando el archivo **ejemplo4.p**hp Puntos importantes a destacar son:

Bloque:

```
$pointObject = ms_newpointObj();
$pointObject->setXY($HTTP_POST_VARS["mapa_x"],$HTTP_POST_VARS["mapa_y"]);
```

Creación de objeto punto, con información de posición (x,y) capturada a partir de la interacción Mouse -> ventana de despliegue **(1)** .

Bloque:

```
$mapObject->zoompoint($zoomFactor,$pointObject,$mapObject->width,$mapObject->height,$extentRectObject);
```

Paso de parámetros a través de la función **zoompoint** hacia objeto correspondiente al mapa **(2).**

Usted puede aplicar trasparencias a las capas definiendo dentro de la estructura de layer el atributo **TRANSPARENCY** e indicando un valor entre (**0-100**) donde 100 es opaco y 0 totalmente transparente.

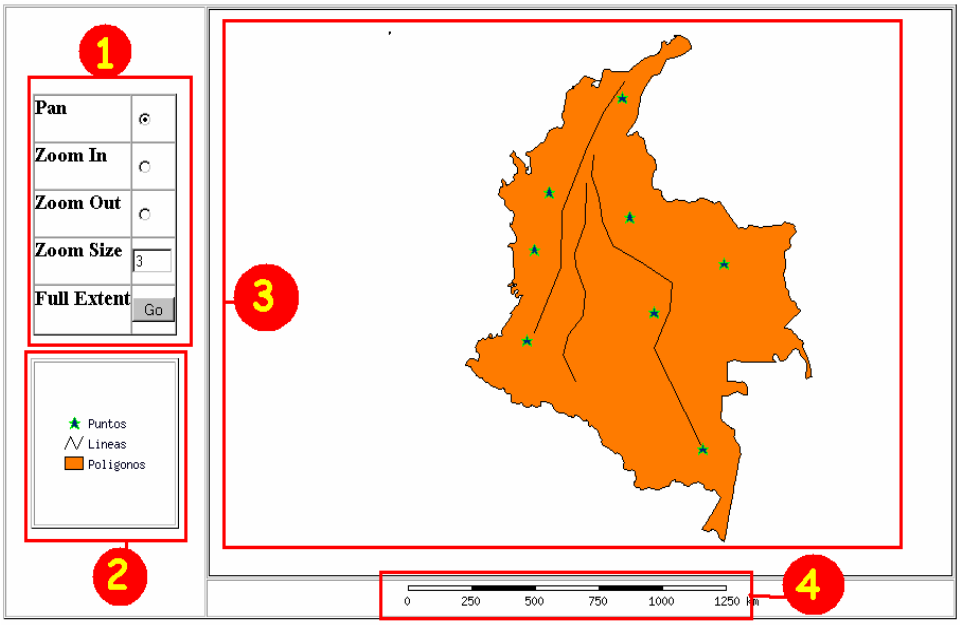
```
LAYER
        NAME "Poligonos "
        STATUS ON
        DATA "poligonos.shp"
        TRANSPARENCY 90
        TYPE  POLYGON
        CLASS
                STYLE
                  COLOR 255 250 250
                  OUTLINECOLOR 100 150 155
                END
        END
END
```

Más Dinamismo

En el ejemplo5 añadiremos más controles como: (escala grafica y leyenda), el resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo5.php**
Donde **(1)** corresponde a la barra de herramientas, exactamente igual a la del ejemplo anterior, **(2)** leyenda, **(3)** mapa renderizado**, (4)** escala grafica. Puntos importantes a destacar sobre los archivos **ejemplo5.map** y **ejemplo5.php** son:

Bloque:

```
LEGEND
  IMAGECOLOR 255 255 255
  KEYSIZE 18 12
  KEYSPACING 5 5
    LABEL
      SIZE SMALL
      TYPE BITMAP
      BUFFER 0
      COLOR 0 0 30
      FORCE FALSE
      MINDISTANCE -1
      MINFEATURESIZE -1
      OFFSET 0 0
      PARTIALS TRUE
    END
POSITION LL
STATUS ON
END
```



Definición de las propiedades de la leyenda a visualizar este bloque de codigo se encuentra definido en el .map.

Bloque:

```
$mapLegend = $mapObject->drawLegend();
$urlLegend = $mapLegend->saveWebImage(MS_GIF, 0, 0, -1);
```

Creación y renderizado del objeto leyenda a través de phpmapscript.

Bloque:

```
<img src="<?php echo $urlLegend; ?>" alt="leyenda" border="0" >
```

Despliegue de leyenda a través de HTML.

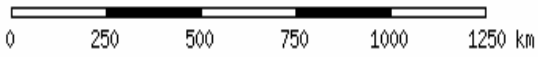
Bloque:

```
LAYER
        NAME "Poligonos"
        STATUS ON
        DATA "poligono.shp"
        TYPE POLYGON
        CLASS
        NAME "Poligonos"
                STYLE
                        COLOR 255 123 0
                        OUTLINECOLOR 0 0 0
                END
        END
END
```

Debemos prestar un especial cuidado, dentro de cada layer al cual le deseemos generar leyenda, en la definición de la clase debemos agregar el atributo NAME, este es aquel que se mostrara una vez mapserver renderice la leyenda.

Bloque:

```
SCALEBAR
    IMAGECOLOR 255 255 255
     LABEL
          COLOR 0 0 0
          SIZE SMALL
     END
     SIZE 350 4
     COLOR 255 255 255
     BACKGROUNDCOLOR 0 0 0
     OUTLINECOLOR 0 0 0
     UNITS KILOMETERS
     INTERVALS 5
    STATUS ON
END
```



Bloque:

```
$mapScale = $mapObject->drawScaleBar();
$urlScale = $mapScale->saveWebImage(MS_GIF, 0, 0, -1);
```

Creación y renderizado del objeto de escala a través de phpmapscript**.**

Bloque:

```
<img src="<?php echo $urlScale; ?>" border="0" >
```

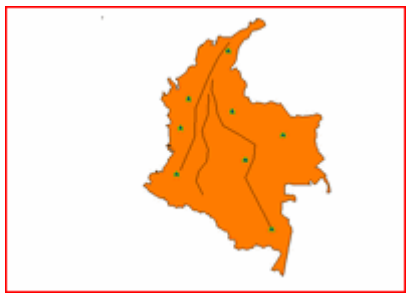
Despliegue de la escala grafica a través HTML.

Añadiendo más objetos interesantes

En el ejemplo6 añadiremos más controles como: (mapa de referencia y rosa de los vientos).
Puntos importantes a destacar sobre los archivos **ejemplo6.map** y **ejemplo6.php** son:

Bloque:

```
REFERENCE
          IMAGE mapaclave.gif
          EXTENT -88 -5 -62 13
          STATUS ON
          COLOR -1 -1 -1
          OUTLINECOLOR 255 0 0
          SIZE 200 143
END
```



En nuestro mapa definimos un nuevo bloque de código el cual indica que usaremos una imagen **IMAGE** llamada mapaclave.gif como mapa de referencia, para nuestro ejemplo esta imagen fue una pre-visualización de nuestro mapa, grabada como un archivo adicional de nombre ***mapaclave.gif*** cuyas dimensiones son: 200 x 143px , debemos tener especial cuidado en proporcionar el correcto extent.

Bloque:

```
$keyMapImage = $mapObject->drawreferencemap();
$urlKeyMap = $keyMapImage->saveWebImage(MS_GIF, 0, 0, -1);
```

Creación y renderizado del objeto de referencia.

Bloque:

```
<img src="<?php echo $urlKeyMap; ?>" border="0" >
```

Despliegue del mapa de referencia a través HTML.

Bloque: en **misc/symbols/symbols.sym**

```
SYMBOL
        NAME 'rosavientos'
        TYPE PIXMAP
        IMAGE 'norte.gif'
END
```

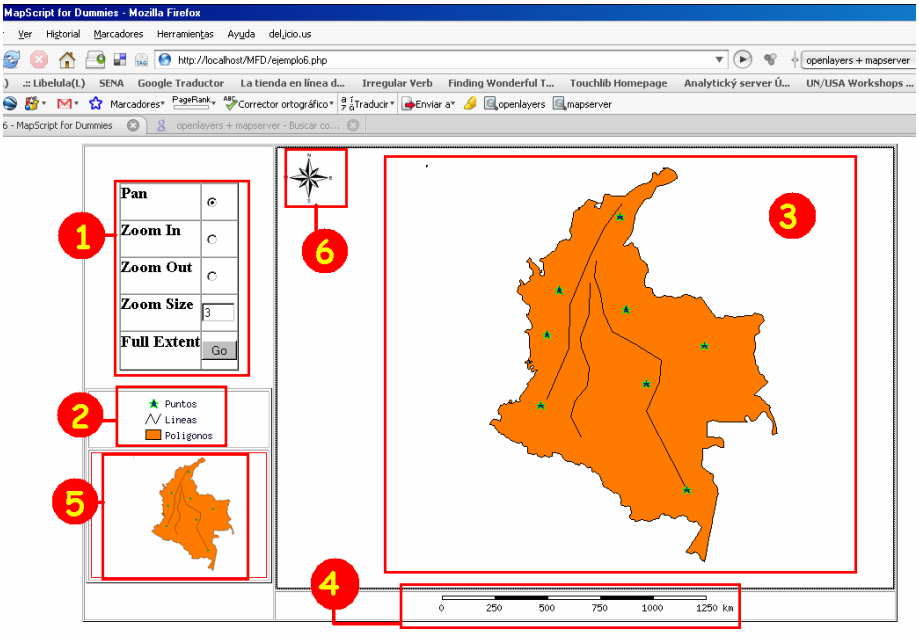
Agregaremos unas nuevas líneas al archivo symbols.sym, donde creamos un símbolo a través del renderizado de una imagen ***norte.gif***

Blaque:

```
LAYER
        NAME "Norte"
          TYPE POINT
          STATUS ON
          TRANSFORM OFF
          POSTLABELCACHE TRUE
                FEATURE
                        POINTS
                                35 35
                        END
                END
          CLASS
                SYMBOL 'rosavientos'
                COLOR 0 0 0
                OUTLINECOLOR 0 0 0
                STYLE END
           END
END
```

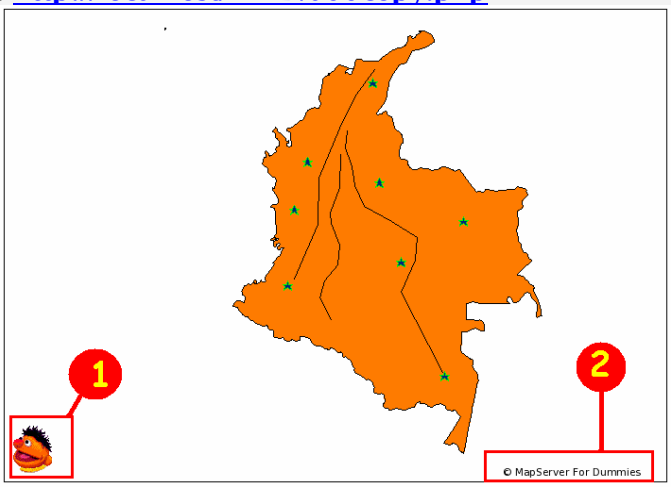
Definimos un nuevo layer de tipo punto (**POINT**) en el .map el cual usaremos para asignar un símbolo (**rosavientos**) en el punto de anclaje (**POINTS**).

El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo6.php**



Donde **(1)** corresponde a la barra de herramientas, exactamente igual a la del ejemplo anterior, **(2)** leyenda, **(3)** mapa renderizado**,** **(4)** escala grafica, **(5)** mapa de referencia, **(6)** rosa de los vientos .

Usted puede añadir marcas de copyright de texto **(2)**, o logo símbolos **(1)** a la imagen renderizada, mira los siguientes archivos. **addcopy.map** y **addcopy.php**, el resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/addcopy.php**



Manejo de Capas

En el ejemplo7 añadiremos el control de capas (layer selector como en alguno de los software GIS).
Puntos importantes a destacar sobre los archivos **ejemplo7.map** y **ejemplo7.php** son:

Bloque:

```
$item = $_REQUEST["layerselector"];
$allLayersObject=$mapObject->getAllLayerNames();
foreach ($allLayersObject as $idx => $layer)
{
        $layerObject=$mapObject->getLayerByName($layer);
        if( sizeof( $item ) > 0 )
        {
                if (in_array( $layerObject->name, $item ))
                {
                        $layerObject->set( "status", MS_ON );
                }
        else
                {
                        $layerObject->set( "status", MS_OFF );
                }
        }
}
```

☑ Poligonos
☑ Lineas
☑ Puntos

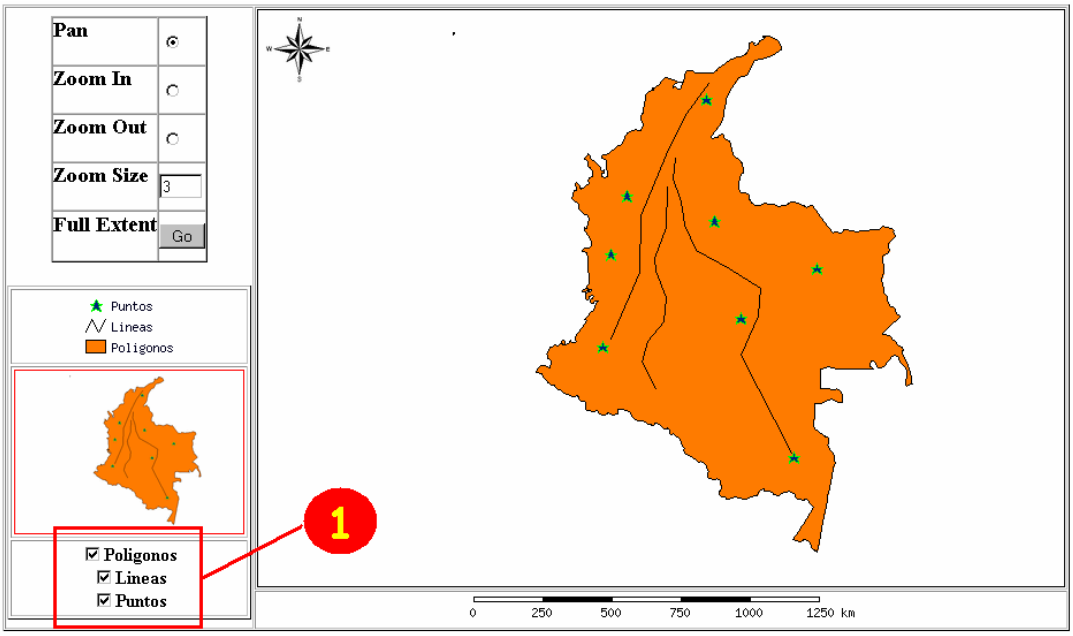
En este bloque le solicitamos al objeto de mapa que nos indique todas las capas por el cual esta compuesto **getAllLayerNames()** .Luego según la petición (encender / apagar) proveniente de los check box en HTML, **getLayerByName($layer)** encendemos **$layerObject->set("status", MS_ON)** o apagamos **$layerObject->set("status", MS_OFF)** el layer correspondiente.

Bloque:

```
<?php
    $allLayersObject=$mapObject->getAllLayerNames();
    foreach ($allLayersObject as $idx => $layer)
    {
        $layerObject=$mapObject->getLayerByName($layer);
        if ($layerObject->status==MS_ON)
        {
                if(($layerObject->name != "Norte") )
                {
                    for($i=0;$layerObject->getClass($i);$i++) {
                       $Class = $layerObject->getClass($i);
                       echo "<input type='checkbox' value='{$layerObject->name}'
                       name='layerselector[]' onClick='document.frmlayerselector.submit();' checked>";
                       echo "<span>{$Class->name}</span><br /> "; }
                }
        }
    else
        {       if(($layerObject->name != "Norte") )
                {
                   for($i=0;$layerObject->getClass($i);$i++) {
                       $Class = $layerObject->getClass($i);
                       echo "<input type='checkbox' value='{$layerObject->name}'
                       name='layerselector[]' onClick='document.frmlayerselector.submit();'  > ";
                       echo "<span>{$Class->name}</span><br /> ";  }
                }
        }
    }
?>
```

En este bloque generamos código HTML **(1)** el cual será representado por el navegador mediante las cajas de checkeo (checkboxes), siempre mantendrá actualizadas las cajas que se encuentren encendidas y apagadas, adicionalmente agregamos un condicional para que nuestro layer de nombre Norte, siempre mantenga encendido.

El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo7.php**



Añadido eventos

Bloque:

```
$polLayer = $mapObject->getLayerByName("Poligonos");
$polLayer->set("status",MS_ON);
$punLayer = $mapObject->getLayerByName("Puntos");
$punLayer->set("status",MS_ON);
$linLayer = $mapObject->getLayerByName("Lineas");
$linLayer->set("status",MS_ON);
```

A través de mapscript traigo una instancia de las capas.

Bloque:

```
$dfKeyMapXMin = $mapObject->extent->minx;
$dfKeyMapYMin = $mapObject->extent->miny;
$dfKeyMapXMax = $mapObject->extent->maxx;
$dfKeyMapYMax = $mapObject->extent->maxy;
$dfWidthPix = doubleval($mapImage->width);
$dfHeightPix = doubleval($mapImage->height);
$nClickGeoX = GMapPix2Geo($_REQUEST['mapa_x'], 0, $dfWidthPix,  $dfKeyMapXMin, $dfKeyMapXMax, 0);
$nClickGeoY = GMapPix2Geo($_REQUEST["mapa_y"], 0, $dfHeightPix, $dfKeyMapYMin, $dfKeyMapYMax, 1);
$my_point = ms_newpointObj();
$my_point->setXY($nClickGeoX,$nClickGeoY);
```

Genero punto con coordenadas mapa, invoco función que convierte de coordenadas píxel a mapa.

Bloque:

```
<?php
    function GMapPix2Geo($nPixPos, $dfPixMin, $dfPixMax, $dfGeoMin, $dfGeoMax, $nInversePix)
    {
        $dfWidthGeo = $dfGeoMax - $dfGeoMin;
        $dfWidthPix = $dfPixMax - $dfPixMin;
        $dfPixToGeo = $dfWidthGeo / $dfWidthPix;
        if (!$nInversePix)
            $dfDeltaPix = $nPixPos - $dfPixMin;
     else
            $dfDeltaPix = $dfPixMax - $nPixPos;
            $dfDeltaGeo = $dfDeltaPix * $dfPixToGeo;
            $dfPosGeo = $dfGeoMin + $dfDeltaGeo;
     return ($dfPosGeo);
}
?>
```

Función para convertir de coordenadas píxel, a coordenadas mapa.

Bloque:

```
//Query a Puntos
if(@$punLayer->queryByPoint($my_point, MS_SINGLE, 200) == MS_SUCCESS)
{
            $results = $punLayer->{resultcache};
            $punLayer->open();
            $rslt = $punLayer->getResult(0);
            $shape = $punLayer->getShape($rslt->tileindex, $rslt->shapeindex);
            $resultadoConsluta  = $shape->values["texto"];
            echo "<center><br><br>Resultado de la Consulta Sobre Capa Puntos: <b> ".
            $resultadoConsluta . "</b></center>";
} else{ echo "No pudo realizar la consulta, vuelva a intentar !!"; }
```

Mediante esta función desplegamos el resultado de abrir el layer, consultar el atributo **$shape->values["texto"]** alfanumérico correspondiente al hacer **queryByPoint** del objeto seleccionado.

Bloque:

```
//Query a Poligonos
if(@$polLayer->queryByPoint($my_point, MS_SINGLE, 200) == MS_SUCCESS)
{
        $results = $polLayer->{resultcache};
        $polLayer->open();
        $rslt = $polLayer->getResult(0);
        $shape = $polLayer->getShape($rslt->tileindex, $rslt->shapeindex);
        $resultadoConsluta  = $shape->values["CNTRY_NAME"];
        echo "<center>Resultado de la Consulta Sobre Capa Poligonos:<b> ".$resultadoConsluta . "</b></center>";
}else{ echo "No pudo realizar la consulta, vuelva a intentar !!"; }
```

Mediante esta función desplegamos el resultado de abrir el layer, consultar el atributo **$shape->values["CNTRY_NAME "]** alfanumérico correspondiente al hacer **queryByPoint** del objeto seleccionado.

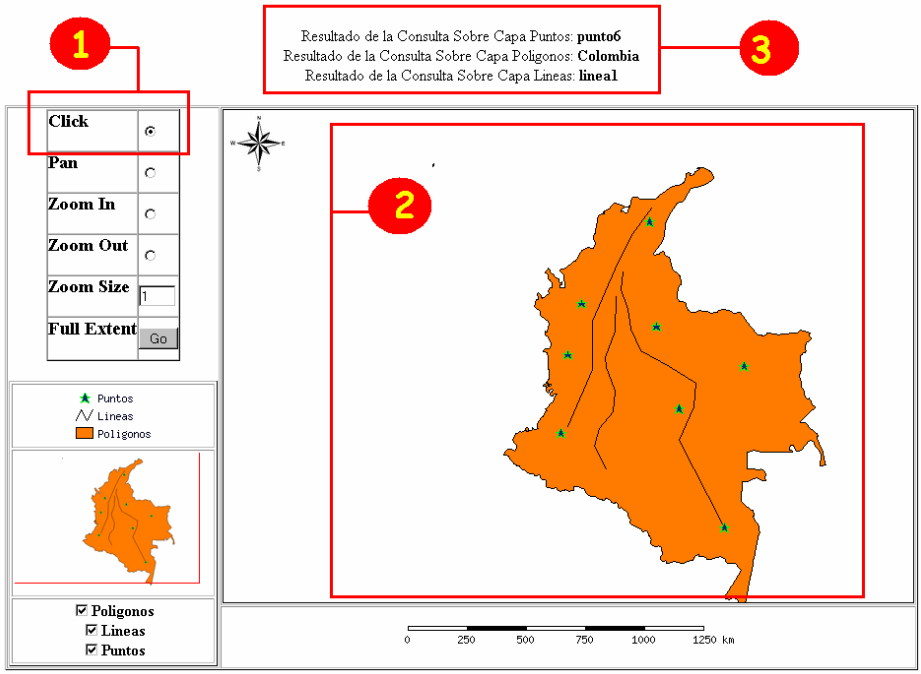
Bloque:

```
//Query a Lineas
if(@$linLayer->queryByPoint($my_point, MS_SINGLE, 200) == MS_SUCCESS)
{
        $results = $linLayer->{resultcache};
        $linLayer->open();
        $rslt = $linLayer->getResult(0);
        $shape = $linLayer->getShape($rslt->tileindex, $rslt->shapeindex);
        $resultadoConsluta  = $shape->values["texto"];
       echo "<center>Resultado de la Consulta Sobre Capa Lineas:<b> ".
       $resultadoConsluta . "</b><br><br></center>";
}else{ echo "No pudo realizar la consulta, vuelva a intentar !!"; }
```

Mediante esta función desplegamos el resultado de abrir el layer, consultar el atributo **$shape->values["texto "]** alfanumérico correspondiente al hacer **queryByPoint** del objeto seleccionado.

Bloque:

```
LAYER
  NAME "Poligonos"
  STATUS ON
    DATA "poligono.shp"
    TYPE POLYGON
     CLASS
      TEMPLATE 'poligonos.html'
      NAME "Poligonos"
           STYLE
              COLOR 255 123 0
              OUTLINECOLOR 0 0 0
           END
      END
     TOLERANCE 20
END
```



Para cada layer, debemos agregar Atributo **TEMPLATE** y **TOLERANCE**.

El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo8.php** , primero seleccionamos la herramienta click **(1),** luego picamos sobre algunas de las entidades graficas **(2)** y por ultimo visualizamos el resultado en **(3).**


```
LAYER
  NAME "LayerMapInfo"
   TYPE LINE
   STATUS ON
    CONNECTIONTYPE OGR
    CONNECTION "archivomapinfo.TAB"
   CLASS
     COLOR 255 0 0
     NAME "LayerMapInfo"
   END
END
```

Usted puede añadir capas no solo de ESRI shape, también puede añadir capas en formato TAB de MapInfo.

No más archivos .MAP

En el ejemplo9 dibujaremos nuestro mapa, sin necesidad de usar un archivo .map, puntos importantes a destacar este archivo **ejemplo9.php** son:

Bloque:

```
$mapObject = ms_newMapObj("");
$mapObject->set("name","Pruebas");
$mapObject->set("shapepath","C:/ms4w/Apache/htdocs/MFD/shapes/");
$mapObject->setSize(700,500);
$mapObject->setExtent(-88 ,-5 ,-62 ,13);
$mapObject->web->set("imagepath" ,"C:/ms4w/Apache/htdocs/MFD/tmp/" );
$mapObject->web->set( "imageurl", "tmp/" );
```

Defino las propiedades basicas de creación de objeto del mapa.

Bloque:

```
$layerPoligono = ms_newLayerObj($mapObject);
$layerPoligono->set( "name", "Poligonos");
$layerPoligono->set( "type", MS_LAYER_POLYGON);
$layerPoligono->set( "status", MS_ON);
$layerPoligono->set( "data", "poligono.shp");
$clasePoligono = ms_newClassObj($layerPoligono);
$estiloPoligono = ms_newStyleObj($clasePoligono);
$estiloPoligono->color->setRGB(255,123,0);
$estiloPoligono->outlinecolor->setRGB(0, 0, 0);
```

Defino las propiedades de cada layer. Clases y estilos correspondientes a esta igual como en la estructura del MAPFILE (.map).

Bloque:

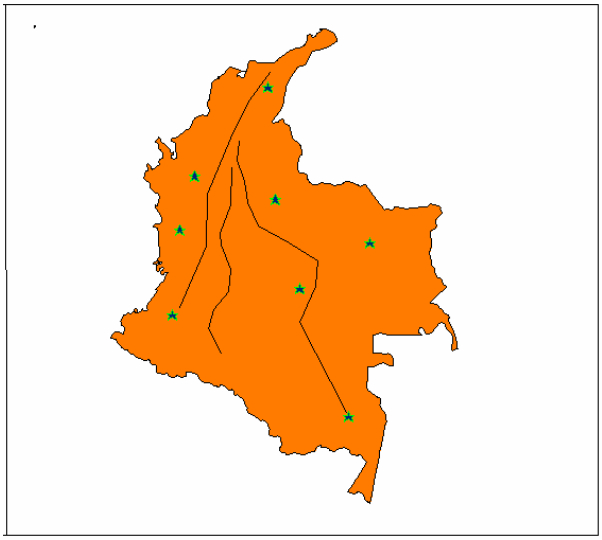
```
$symbolid = ms_newSymbolObj($mapObject, "star");
$oSymbol = $mapObject->getsymbolobjectbyid($symbolid);
$oSymbol->setpoints(Array(0 ,.375 , .35, .375,.5, 0,.65, .375,1 ,.375,.75, .625,.875 ,1,.5 ,.75,.125, 1,.25 ,.625));
$oSymbol->set("filled",MS_TRUE);
$oSymbol->set("inmapfile", MS_TRUE);
```

Creamos símbolos y otras estructuras complejas.

Bloque:

```
$estiloPuntos->color->setRGB(0 ,34 ,125);
$estiloPuntos->outlinecolor->setRGB(0 ,255, 0);
$estiloPuntos->set("symbolname", "star");
 $estiloPuntos->set("size", "10");
```

Finalmente estructuras complejas, las podemos asignar a las entidades graficas atravez de los estilos.



El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo9.php** representa nuestro mapa de ejemplo el cual ha sido renderizado sin necesidad de definir un archivo externo .map.

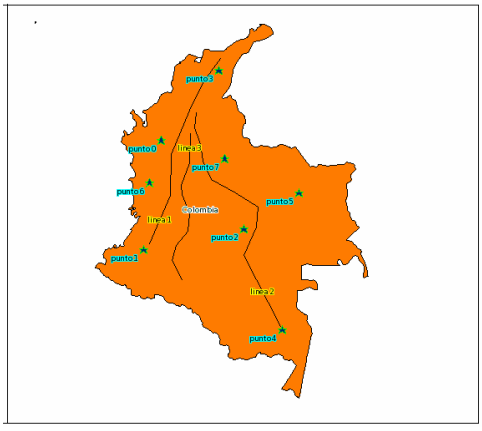
De igual manera podemos acceder a repositorios remotos (bases de datos PostgreSQL) , el ejemplo10, presenta la misma salida grafica, ligeros cambios se realizan en el archivo **ejemplo10.php**, donde rescatamos el siguiente bloque de código.

Bloque:

```
$layerLineas->set("connectiontype",MS_POSTGIS);
$layerLineas->set("connection","user=postgres password=1234567 dbname=prueba host=localhost");
$layerLineas ->set("data","the_geom FROM lineas as lineas using unique gid using SRID=-1");
```

El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo10.php**


```
LABELITEM "texto"
LABELCACHE ON

LABEL
        COLOR 0 0 0
        FONT sans
        TYPE TRUETYPE
        POSITION CC
        PARTIALS TRUE
        SIZE 7
        BUFFER 1
       OUTLINECOLOR 255 255 0
END
```



Si desea añadir etiquetas (**labels**) a las entidades graficas, puedes ver un ejemplo digitando en el navegador la url: **http://localhost/MFD/labels.php** el cual usa **labels.map .**

Creando un Servidor WMS

Nuestro mapfile nos permite publicar mapas a través de wms, en el ejemplo11.map lograremos esto realizando unos ligeros cambios en la estructura, trozos a destacar los veremos a continuación.

Bloque:

```
WEB
        METADATA
        "wms_title"             "Ejemplo WMS de MapServer for Dummies"
        "wms_onlineresource"    "http://localhost/cgi-bin/mapserv.exe?map=../htdocs/MFD/ejemplo11.map&"
        "wms_srs"               "EPSG:4326"
        END
END
```

Ya no es necesario definir un path de salida, ahora tan solo basta definir un bloque de **METADATA.**

Bloque:

```
PROJECTION
        "init=epsg:4326"
END
```

Al igual que en el ejemplo de acceso a servicios WMS, para la publicación de nuestra información espacial es necesario definir una proyección de salida, en este caso usaremos WGS84 con su código EPGS equivalente.

Bloque:

```
LAYER
        NAME "Poligonos"
        METADATA
                "wms_srs"           "EPSG:4326"
                "wms_name"          "Poligonos"
                "wms_server_version"  "1.1.1"
                "wms_format"        "image/png"
                "wms_transparent"     "true"
        END
        STATUS ON
        DATA "poligono.shp"
        TYPE POLYGON
        CLASS
                STYLE
                        COLOR 255 123 0
                        OUTLINECOLOR 0 0 0
                END
        END
END
```

Debemos definir para cada layer que deseemos hacer visible mediante WMS basta tan solo definir el bloque **METADATA** indicando los atributos de tipo de proyección (wms_srs), nombre accesible para esa capa (wms_name), versión del servidor wms (wms_server_version), formato de publicación (wms_format), si deseamos que nuestra capa sea publicada con soporte de transparencias (wms_transparent) indicamos trae, de lo contrario indicamos el atributo de false.

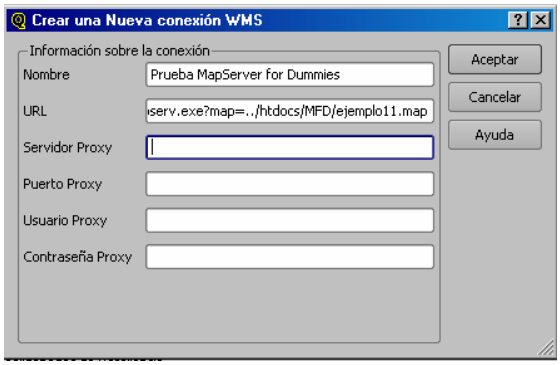
Consumiendo servicios WMS

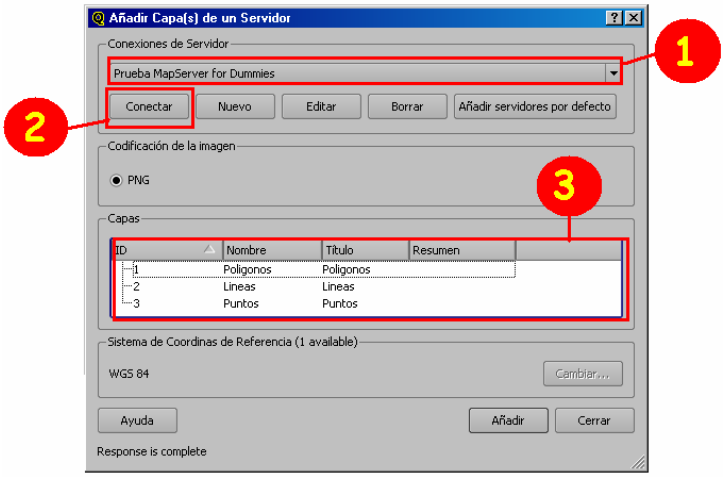
Accederemos a las capas servidas en WMS a través de un cliente pesado, en este caso se usara Quamtum GIS, pero en general puedes usar cualquier tipo de software GIS el cual soporte acceso a datos de fuente WMS.

Quantum GIS es un proyecto OpenSource multiplataforma, el cual podemos obtener desde la siguiente URL. **http://www.qgis.org/.**

Para añadir una capa WMS en QuantumGIS, nos dirigimos al menú **Capas –> Añadir capa WMS** , en esta ventana emergente procedemos a crear una nueva conexión; indicando un nombre y la URL de acceso correspondiente a nuestro servicio previamente creado.

URL: **http://localhost/cgi-bin/mapserv.exe?map=../htdocs/MFD/ejemplo11.map**





Una vez creada la conexión **(1),** procedemos a conectarnos con el servicio **(2).**

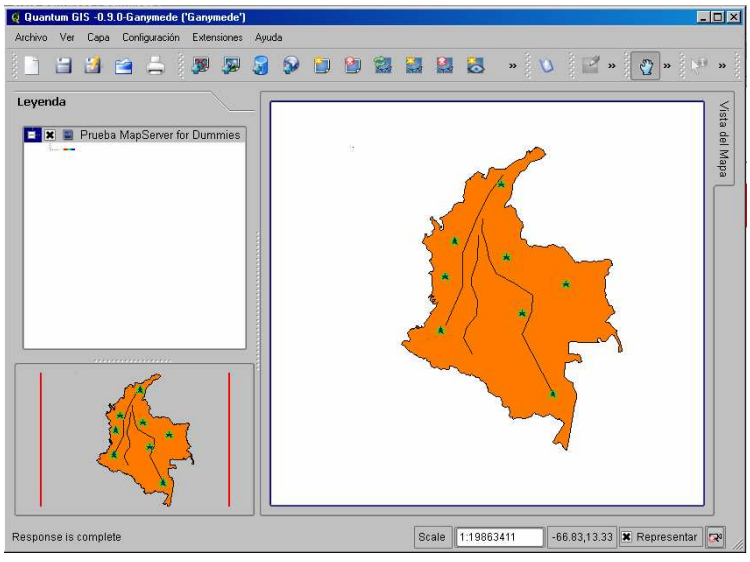
El software internamente se encarga de realizar la petición **getCapabilities** al servidor.

Devolviendo las capas disponibles servidas **(3),** para nuestro ejemplo; capa: Polígonos, Líneas, Puntos.

http://localhost/cgi-bin/mapserv.exe?map=../htdocs/MFD/**ejemplo11.map**&service=**WMS**&request=**getcapabilities**

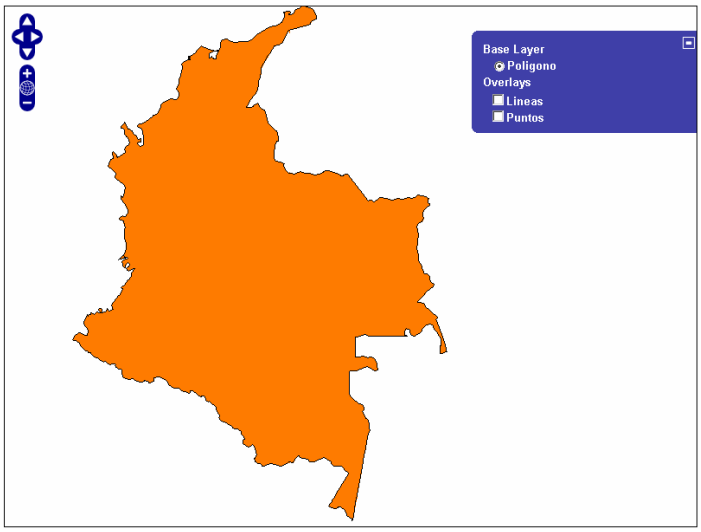
Añadimos las capas disponibles **(3),** el resultado es un mapa compuesto entregado por el mapserver. Igualmente podemos acceder a cada capa de forma independiente, y combinar estas con otro tipo de información espacial.

En **acceso_wms.qgs** se guardo el proyecto de acceso al ejemplo presentado.



OpenLayers

Al igual que podemos acceder a servicios wms, a través de clientes pesados, Openlayers nos permite acceder a estos mediante la interfaz web.



Openlayers es una librería OpenSource en JavaScript, la cual podemos obtener en la siguiente URL: **http://www.openlayers.org/** , OpenLayers hace muy fácil el despliegue de mapas dinámicos en la WEB, podemos acceder a nuestro WMS previamente creado digitando la URL: **http://localhost/MFD/ejemplo12.html**

De este archivo podemos rescatar los siguientes bloques de código.

Bloque:

```
<script src="misc/lib/OpenLayers.js"></script>
```

Iniciamos la librería JavaScript OpenLayers.

Bloque:

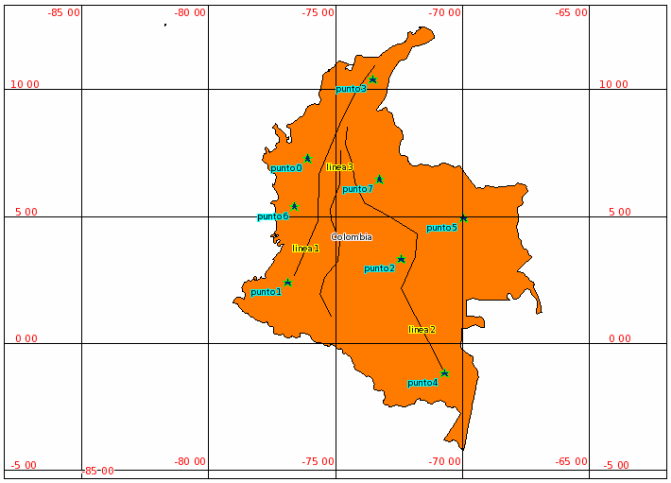
```
map = new OpenLayers.Map( 'map', {maxResolution: 'auto'} );
var layerPoligono = new OpenLayers.Layer.MapServer("Poligono",
"http://localhost/cgi-bin/mapserv.exe?map=../htdocs/MFD/ejemplo11.map",
{layers: 'Poligonos'}, {isBaseLayer: true,buffer: 1,  gutter:0} );
```

Creamos el objeto mapa y agregamos la capa de mapserver haciendo una llamada a nuestro servicio WMS).

Bloque:

```
map.addLayers([layerPoligono,layerLineas,layerPuntos]);
```

Agregamos las capas a nuestro objeto de map.



Si desea añadir etiquetas (**grillas**) a la salida grafica resultante, remítase al ejemplo grilla.map y grilla.php, puedes ver el ejemplo digitando en el navegador la url: **http://localhost/MFD/grid.php**.

Mapas Dinámicos

En la onda de las web2.0, mscross un cliente GIS Ajax OpenSource el cual lo podemos obtener en la siguiente URL: **http://sourceforge.net/projects/mscross** , escrito en JavaScript el cual nos permite realizar una implementación limpia y sencilla de nuestra información geo-espacial.

Ejemplo13.php corresponde
De este archivo podemos rescatar los siguientes bloques de código.

Bloque:

```
<script src="misc/lib/mscross-1.1.9.js" type="text/javascript">
</script>
```

Implementamos la librería JavaScript mscross.

Bloque:

```
myMap1 = new msMap( document.getElementById("dc_main"), 'standardRight' );
myMap1.setCgi( '/cgi-bin/mapserv.exe' );
myMap1.setMapFile( '/ms4w/Apache/htdocs/MFD/ejemplo13.map' );
myMap1.setFullExtent( -88 , -62, -5);
myMap1.setLayers( 'Poligonos Lineas Puntos' );
```

Una vez creada la instancia de mscross, pasaremos parámetros de configuración.

Bloque:

```
myMap1.redraw();
```

Dibujo el mapa utilizando mscross.

Bloque:

```
<input CHECKED onClick="chgLayers()" type="checkbox" name="layer[0]" value="Poligonos">
```

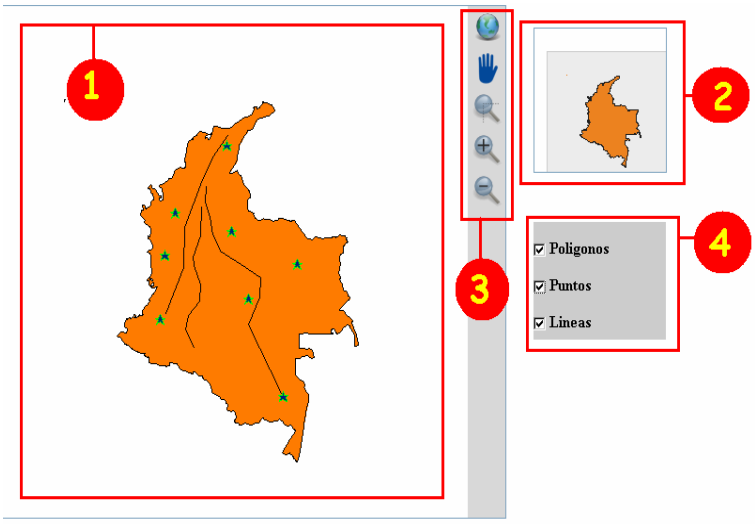
Control de capas.

Bloque:

```
myMap1.setLayers( list );
```

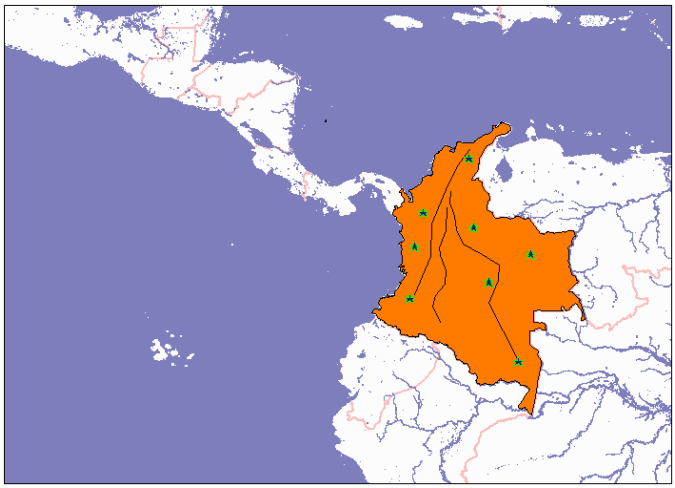
Asigno a objeto map de mscross, la lista de layers a desplegar.

El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/ejemplo13.php** representa nuestro mapa visualizado utilizando la librería Ajax de mscross.



Añadiendo Capas Raster

El resultado lo podemos ver digitando en nuestro navegador la url:
http://localhost/MFD/ejemplo14.php



Una capa WMS es una capa raster.

Pero si necesitamos agregar otro tipo de dataset de tipo raster recomendamos checkear la documentación:
http://mapserver.gis.umn.edu/docs/howto/raster_data

Bloque:

```
LAYER
    NAME landsat
    DATA "landsatimagen.tif"
    STATUS DEFAULT
    TYPE RASTER
    PROCESSING "BANDS=1,2,3"
    OFFSITE 71 74 65

    PROJECTION
      "init=epsg:4326"
    END
END
```

Agregando bandas 1,2,3 de una imagen landsat en .TIF

A través de **ogr2ogr** en FWtools, usted puede realizar múltiples conversiones de información espacial mediante la línea de comandos.
Por ejemplo; convertir de un ESRI shape a OGC GML tan solo basta digitar. **ogr2ogr -f "GML" puntos.gml puntos.shp o** convertir una cobertura a KML para ser visualizada en GoogleEarth digitando; **ogr2ogr -f KML puntos.kml puntos.shp** lo conseguiremos. Usando ogr2ogr podemos ahorrarnos unos valiosos minutos de nuestras vidas, si deseas insertar todos los shapes que se tengan en determinado fólder directamente a la base de datos PotgreSQL sin tener que realizar la tediosa labor de convertirlos inicialmente a sql y luego insertarlos uno a uno, con este script **/misc/ fast_shp2pgsql.bat** puedes insertarlos todos en segundos y sin complicaciones, debes ejecutarlo a través de **FWTools Shell.**

Borrando Cache

Cada vez que sea invocado mapserver realizándole la petición de renderizar una nueva vista o mapa este generara un archivo de cache (imagen) con la vista actual, si tenemos grandes volúmenes de consultas a la aplicación desarrollada, nuestro disco duro se llenara en un abrir y cerrar de ojos. Mediante este script borraremos el cache, digitando en la barra direcciones del navegador: **http://localhost/MFD/delcache.php ,** este script viene acompañado de una sencilla rutina en batch ubicada en **tmp/del_cache.bat** la cual es invocada para realizar el borrado.

Más Ejemplos

A manera de extra se presentan una serie de ejemplos.

Achurado

Bloque:

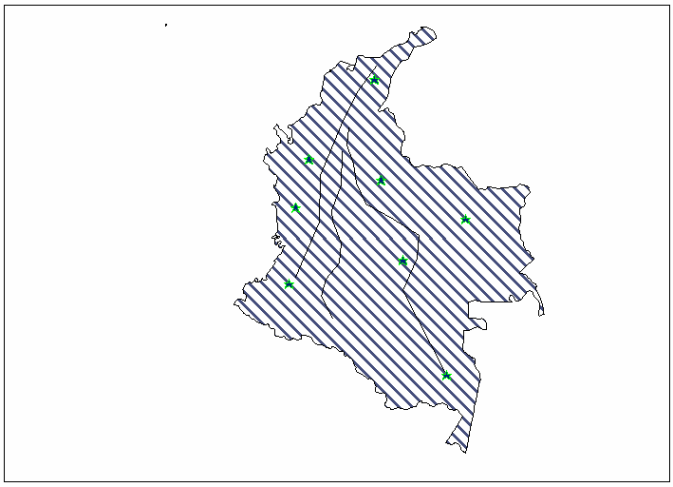
```
SYMBOL
        NAME 'achurado'
        TYPE HATCH
END
```

Defino el símbolo que será un achurado.

Bloque:

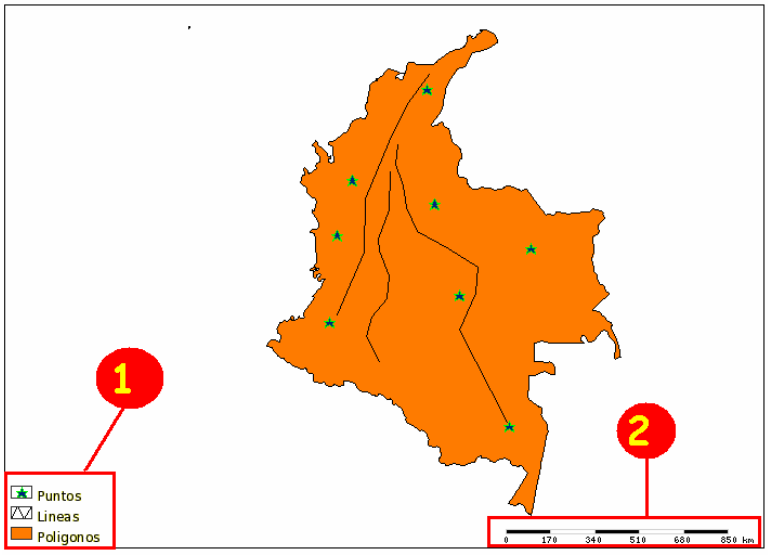
```
STYLE
        SYMBOL 'achurado'
        ANGLE -45
        SIZE 10
        WIDTH 3
        COLOR 69 78 124
        OUTLINECOLOR 0 0 0
END
```

Asigno el símbolo de achurado al estilo.



El resultado lo podemos ver digitando en nuestro navegador la url:
http://localhost/MFD/achurado.php

Escala y Leyenda Embebida



Al igual que los objetos escala **(2)** y leyenda **(1)** se renderizan por separado, podemos embeber estos dentro del mismo mapa, checkea los archivos;
escala_leyendaembebida.php y escala_leyendaembebida.map

El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/escala_leyendaembebida.php**

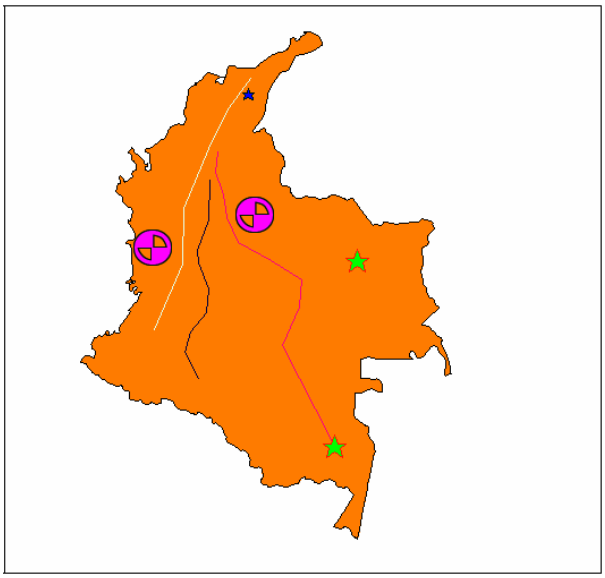
Includes

Al igual que en algunos lenguajes, la estructura MAP permite incluir trozos o fragmentos de archivos y ponerlos a funcionar dentro de la implementación, el ejemplo **includes.php** usa **includes.map** y este a su vez usa **includeslayer.map .** El resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/includes.php**

SLD

Si alguna vez nos preguntamos como extraer el SLD[10] de nuestro mapfile, la respuesta la tenemos digitando en nuestro navegador la url: **http://localhost/MFD/generarSLD.php** una vez generado el resultado, en el navegador vamos a la opción Ver Código Fuente y guardamos esta salida como un documento .xml el cual podremos utilizar para definir el SDL

Expressions



Si desea conocer mas acerca de la sintaxis y la potencialidad de expressions, remitace a **http://mapserver.gis.umn.edu/docs/howto/msexpressions** obtendra informacion con un mayor nivel de detalle.

Para este ejemplo se ha compuesto una salida grafica que usa el archivo expresiones.map indicando el uso basico de este atributo **EXPRESSION,** el resultado lo podemos ver digitando en nuestro navegador la url: **http://localhost/MFD/expresiones.php**

Otras Herramientas

misc/backup_db.bat	Realizar backup a base de datos
misc/ all_shapes.bat	Convertir todos los shp a sql.
misc/del_cache.bat	Borrar cache imágenes generados por mapserver.
misc/fast_shp2pgsql.bat	Insertar rápida y limpiamente todos los shp a una base de datos PostgreSQL.
misc/pgdb2shp.bat	Genera shp de tabla en base de datos PostgreSQL.

[10] Styled Layer Descriptor , http://www.opengeospatial.org/standards/sld

Bibliografía

Algunos enlaces y documentos de interés;

PhpExperto:	http://phpexperto.blogspot.com/
MsCross:	http://sourceforge.net/projects/mscross
Ne@Polis	http://umn.mapserver.ch/index_en.php
FWTools	http://fwtools.maptools.org/
Php/Mapscript	http://mapserver.gis.umn.edu/docs/howto/phpmapscript-byexample
Instituto Humboldt	http://www.humboldt.org.co/humboldt/
QuantumGIS	http://www.qgis.org/
MapServer	http://ms.gis.umn.edu/
GDAL	http://www.gdal.org/ogr/
PHP	www.php.net

Agradecimientos

Esta guía fue elaborada, a partir de navegación de muchos enlaces, saltando de enlace en enlace uno se topa con tutórales, ejemplos, y otra serie de artículos que clarifican el panorama, así que quiero dar gracias a esas personas de las cuales tome cierto tipo de información.

***Jaime M. Tan Nozawa** / phpExperto - "Trabajando con MapServer".*

***Tyler Mitchell** / O'Reilly Media Inc - "Web Mapping Ilustrated".*

***Jeff McKenna** / DMsolutions – "MapFile Reference"*

***Chris Tweedie** / Chris GISmos – "10 Easy steps for converting mxd to map & sld"*

Acerca del Autor

Fabio Andrés Herrera
Ingeniero Topográfico / Universidad del Valle
Santiago de Cali - Colombia

Sitio Personal:	http://andres.hrglobalideas.com
Corporativo:	http://www.hrglobalideas.com
Blog:	http://andresherreracali.blogspot.com
E-mail:	andres@hrglobalideas.com fandresherrera@hotmail.com
IM:	**t763rm3n** / en GTalk
Skype:	fabio.andres.herrera