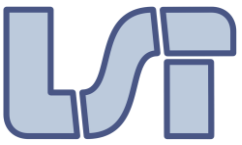
Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Universidad de Granada



Departamento de Lenguajes y Sistemas Informáticos.

Aplicación SRS

Proyecto fin de carrera curso 2008/09

Autor: Álvaro Ortega Cabeza

Director: D. Juan Carlos Torres Cantero

Contenido

1. Introducción.

Un Sistema de Información Geográfica (GIS) es la integración en un mismo sistema del conjunto de datos geográficos así como del software necesario para su manipulación y consulta. Dichos datos geográficos es de suma importancia que estén georreferenciados, de lo contrario sería prácticamente imposible para el sistema software interpretar la información que éstos pretenden mostrar. Es decir, un mapa no se compone únicamente de la imagen, además va acompañada de un conjunto de atributos geográficos asociados a la zona que se pretende representar. De hecho un mapa realmente se compone de este conjunto de datos mencionados. El dibujo del mapa no es más que una representación gráfica de dichos datos para facilitar así su comprensión.

Para la realización del proyecto todas las herramientas que se han utilizado son de libre distribución.

MapServer es una aplicación CGI distribuida de código libre bajo licencia del MIT, que se ejecutará del lado del servidor proporcionando las imágenes de los mapas requeridas por el cliente.

Una aplicación CGI (Common Gateway Interface) se ejecuta a través de una página web y permite a un cliente solicitar datos de un programa ejecutado en un servidor web. El CGI especifica un estándar para transferir datos entre el cliente y el programa.

Además MapServer se encargará de realizar operaciones tales como el zoom o el desplazamiento sobre el mapa siempre, por supuesto, tras una petición por parte del cliente.

El otro programa fundamental para el desarrollo de la aplicación es GRASS bajo licencia GPL, también de código libre.

GRASS se encargará de guardar los mapas vectoriales que compondrán el mapa de rutas y de integrarlos en este último. Además se utilizará para realizar las consultas pertinentes sobre los mapas.

El tratamiento y análisis de mapas es un problema de tiene una gran repercusión en la actualidad siendo numerosos los sistemas que implementan funcionalidad para este tipo de aplicaciones.

En este trabajo, pretendemos afrontar la configuración e implementación de un sistema web, basado en GIS, desde el principio. Es decir, comenzando con la instalación y configuración del software necesario en el servidor para a continuación centrarnos en la programación de las aplicaciones que servirá nuestro sistema.

En este caso se ha optado por la implementación de un servidor de mapas en el que poder consultar y crear rutas orientadas a la práctica del deporte del senderismo. Es el mencionado apartado de la creación de rutas lo que le aporta el carácter más innovador a este trabajo. La mayoría de las aplicaciones actuales son simples visores de mapas en los que se nos permite consultar datos sobre ellos y nada más. En este caso iremos un paso más en lo que a la interactividad del usuario con el sistema se refiere. Dicha interacción es fundamental, evidentemente, en la visualización de mapas ya que es el usuario el que debe seleccionar las capas pertinentes. Sin embargo, es al visualizar la capa de las rutas donde se le proporciona al usuario libertad absoluta para consultar los datos de la ruta que desee. Así mismo podrá efectuar consultas acotadas de acuerdo a las características que desea que tengan las rutas.

Por último, y quizá lo más novedoso de este sistema, es que se le da al usuario la posibilidad de editar el mapa de rutas del servidor creando él sus propias rutas las cuales por supuesto serán visibles también al resto de los usuarios.

Como última característica destacable del trabajo es que no se requiere al usuario que tenga nada instalado en su máquina más que un navegador correctamente configurado para la ejecución de código Javascript y, por supuesto, una conexión a internet.

Siguiendo la filosofía del software libre la aplicación servidora está optimizada para su utilización bajo el navegador Mozilla Firefox.

1.1. Objetivos.

El objetivo de este trabajo es la implementación de un sistema de actualidad en estos tiempos como son los Sistemas de Información Geográfica (SIG) a la vez que se trata de incluir algunos aspectos que aporten algo novedoso a este campo.

En esta ocasión se dotará al usuario de la capacidad de editar los mapas de rutas creando sus propias rutas y almacenándolas en el servidor.

Para ello los primeros pasos a dar será la instalación y configuración del servidor, para pasar posteriormente a la implementación del resto del sistema.

En primer lugar, como ya se ha mencionado, será necesaria la instalación y configuración del servidor. Esto es la instalación y configuración de MapServer que como se ha anunciado anteriormente se encargará de mostrar las imágenes oportunas al usuario así como de realizar las operaciones que tengan que ver con la visualización de los mapas (zoom, pan,…).

A continuación se procederá a la implementación de la funcionalidad necesaria para la edición de rutas sobre los mapas. Esto es, el marcado de dichas rutas, su georreferenciación y posterior almacenado como mapa. El concepto de georreferenciación aludido es de crucial importancia en tanto que cuando se dibuja una ruta sobre el mapa esta viene dada en coordenadas sobre la web. Dichas coordenadas habrá que convertirlas, con el proceso que se mencionará posteriormente, en coordenadas terrestres con el fin que su posterior interpretación sea correcta. El encargado de guardar los mapas en el formato adecuado será GRASS.

Una vez se cuente con las rutas se pretende que se puedan consultar detalles relacionados con ellas como pueden ser la altura media, la distancia media de la ruta, la dureza de la misma (calculada a partir de la distancia y el desnivel), el desnivel medio.

Además se implementará la funcionalidad necesaria para poder crear gráficas con los perfiles de las rutas que relacionen altitud y distancia recorrida.

Por último el usuario podrá seleccionar las rutas que desee. Una vez seleccionadas las mismas se le presentarán en una nueva página junto con todos los datos de interés. Página que el usuario podrá imprimir.

1.2. Contenidos

Esta documentación consta de doce de los cuales el presente corresponde a la introducción al trabajo quedando el resto distribuidos de la siguiente manera:

1. Sistemas de información geográfica. En él se hace una introducción a este tipo de sistemas con el fin de que el lector pueda comprender un poco mejor los términos en los que se moverá este documento al tiempo que le sirve de "referencia histórica" para poder comprobar desde una mejor perspectiva la novedad que este trabajo pretende introducir.

2. Diseño e implementación del sistema. En este capítulo se abordan todas las cuestiones referentes al diseño de la arquitectura sobre la que se implementará el sistema.

3. Instalación y configuración del servidor. Aquí se expondrá como deberá instalarse y configurarse el sistema para su correcto funcionamiento.

4. Análisis de requisitos. En esta ocasión se expondrán los requisitos funcionales y no funcionales del sistema.

5. Descripción y diagramas de casos de uso. Como su nombre indica se especificarán los diagramas de casos de uso así como su descripción para de esta forma proporcionar una descripción más formal de los requisitos del sistema.

6. Especificación de los contratos del sistema. En este caso el objetivo será la especificación formal de las operaciones del sistema.

7. Descripción de las operaciones. Descripción en lenguaje natural de las operaciones del sistema.

8. Manual de uso. Descripción en lenguaje natural del funcionamiento completo del sistema con el objetivo de que el usuario no experimentado pueda manejar el software sin ningún tipo de problemas.

9. Conclusiones. En este capítulo se expondrá una valoración personal del trabajo realizado.

10. Bibliografía.

2. Sistemas de información geográfica.

Según lo expuesto en la referencia [9], un sistema de información geográfica es una integración organizada de *hardware*, *software* y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión. En el sentido más estricto, es cualquier sistema de información capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada. En un sentido más genérico, los SIG son herramientas que permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones.

El profesor David Rhind (1989) lo ha definido en los siguientes términos: "*Es un sistema de hardware, software y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados (georeferenciados), para la solución de los problemas complejos del manejo y planeamiento territorial*".

Problemas tales como: determinar el cierre óptimo de un embalse, proponer el trazado menos costoso para una vía de comunicación, analizar el comportamiento espacial del nivel educativo o evaluar los territorios más propicios para cultivar tal o cual especie, constituyen las típicas interrogaciones a que son sometidas los SIG en su fase operacional. Ellas nunca podrían ser contestadas por los sistemas CAD/CAM, paquetes estadísticos y otros de aplicaciones, aunque todos ellos, apoyan sustancialmente a los SIG en calidad de herramientas de trabajo.

Este tipo de sistemas ha sufrido un enorme auge en los últimos tiempos. Diversas son las causas expuestas para explicarlo

Muchos programas de aplicaciones trabajan con operadores espaciales, sin embargo no se consideran un SIG pues no son capaces de realizar búsquedas y análisis espaciales, condición indispensable para tal consideración. Este es el caso de los sistemas, especialmente implementados sobre web, que se limiten simplemente a implementar meros visores que se encarguen de interpretar los mencionados datos georreferenciados para mostrarlos al usuario de forma clara y en un formato comprensible. La mayoría además permiten la consulta de dichos datos sobre el mapa.

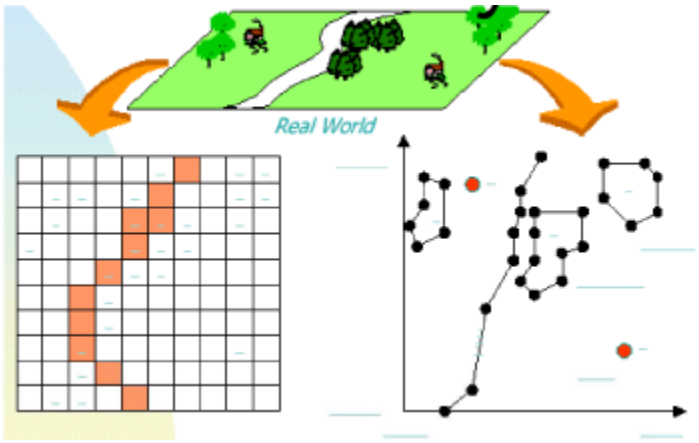
Este sistema pretende dar un paso más adelante permitiendo además la creación de nuevos mapas. Este tema se ha tratado en el punto 1.1 al hablar de los objetivos del trabajo y se valorará en el apartado de conclusiones.

Hay muchas formas de medir la calidad de un SIG. Todas ellas atienden al aspecto del sistema que se evalúa. Intentando dar buscar una medición lo más objetiva posible en [10] se puede encontrar la siguiente definición "*el éxito y la eficacia de un SIG se miden por el tipo, la calidad y vigencia de los datos con los que opera*". Esta definición se apoya en el hecho de que los datos son el principal objetivo de cualquier sistema de información siendo estos independientes del tratamiento que posteriormente se le dé.

Los SIG funcionan con dos tipos de datos geográficos. El modelo raster y el vectorial. Nuestro sistema trabajará también con ambos modelos.

El modelo raster funciona a través de una retícula que permite asociar datos a una imagen; es decir, se pueden relacionar paquetes de información a los pixeles de una imagen digitalizada.

Por su parte, en el modelo vectorial, la información sobre puntos, líneas y polígonos se almacena como una colección de coordenadas x,y. La ubicación de una característica puntual, pueden describirse con un sólo punto x,y. Las características lineales, pueden almacenarse como un conjunto de puntos de coordenadas x,y. Las características poligonales, pueden almacenarse como un circuito cerrado de coordenadas.



2.1.Aplicaciones de los sistemas de información geográfica.

En la mayoría de los sectores los SIG pueden ser utilizados como una herramienta de ayuda a la gestión y toma de decisiones, a continuación se describen brevemente algunas de sus aplicaciones principales:

Cartografía automatizada:

Las entidades públicas han implementado este componente de los SIG en la construcción y mantenimiento de planos digitales de cartografía. Dichos planos son puestos a disposición de las empresas a las que puedan resultar de utilidad estos productos con la condición de que estas entidades se encargan posteriormente de proveer versiones actualizadas de manera periódica.

Infraestructura:

Algunos de los primeros sistemas SIG fueron utilizados por las empresas encargadas del desarrollo, mantenimiento y administración de redes de electricidad, gas, agua, teléfono, alcantarillado, etc.; en este caso, los sistemas SIG almacenan información alfanumérica de servicios relacionados con las distintas representaciones gráficas de los mismos. Estos sistemas almacenan información relativa a la conectividad de los elementos representados gráficamente, con el fin de realizar un análisis de redes.

La elaboración de mapas, así como la posibilidad de realizar una consulta combinada de información, ya sea gráfica o alfanumérica, son las funciones más comunes para estos sistemas, también son utilizados en trabajos de ingeniería, inventarios, planificación de redes, gestión de mantenimiento, entre otros.

Gestión territorial:

Son aplicaciones SIG dirigidas a la gestión de entidades territoriales y permiten un rápido acceso a la información gráfica y alfanumérica, y suministran herramientas para el análisis espacial de la información. Facilitan labores de mantenimiento de infraestructura, mobiliario urbano, etc., y permiten realizar una optimización en los trabajos de mantenimiento de empresas de servicios. Tienen la facilidad de generar documentos con información gráfica y alfanumérica.

Medio ambiente:

Son aplicaciones implementadas por instituciones de medio ambiente, que facilitan la evaluación del impacto ambiental en la ejecución de proyectos. Integrados con sistemas de adquisición de datos permiten el análisis en tiempo real de la concentración de contaminantes, a fin de tomar las precauciones y medidas del caso. Facilitan una ayuda fundamental en trabajos tales como reforestación, explotaciones agrícolas, estudios de representatividad, caracterización de ecosistemas, estudios de fragmentación, estudios de especies, etc.

Equipamiento social:

Implementación de aplicaciones SIG dirigidas a la gestión de servicios de impacto social, tales como servicios sanitarios, centros escolares, hospitales, centros deportivos, culturales,

lugares de concentración en casos de emergencias, centros de recreo, entre otros y suministran información sobre las sedes ya existentes en una determinada zona y ayudan en la planificación en cuanto a la localización de nuevos centros. Un buen diseño y una buena implementación de estos SIG aumentan la productividad al optimizar recursos, ya que permiten asignar de forma adecuada y precisa los centros de atención a usuarios cubriendo de forma eficiente la totalidad de la zona de influencia.

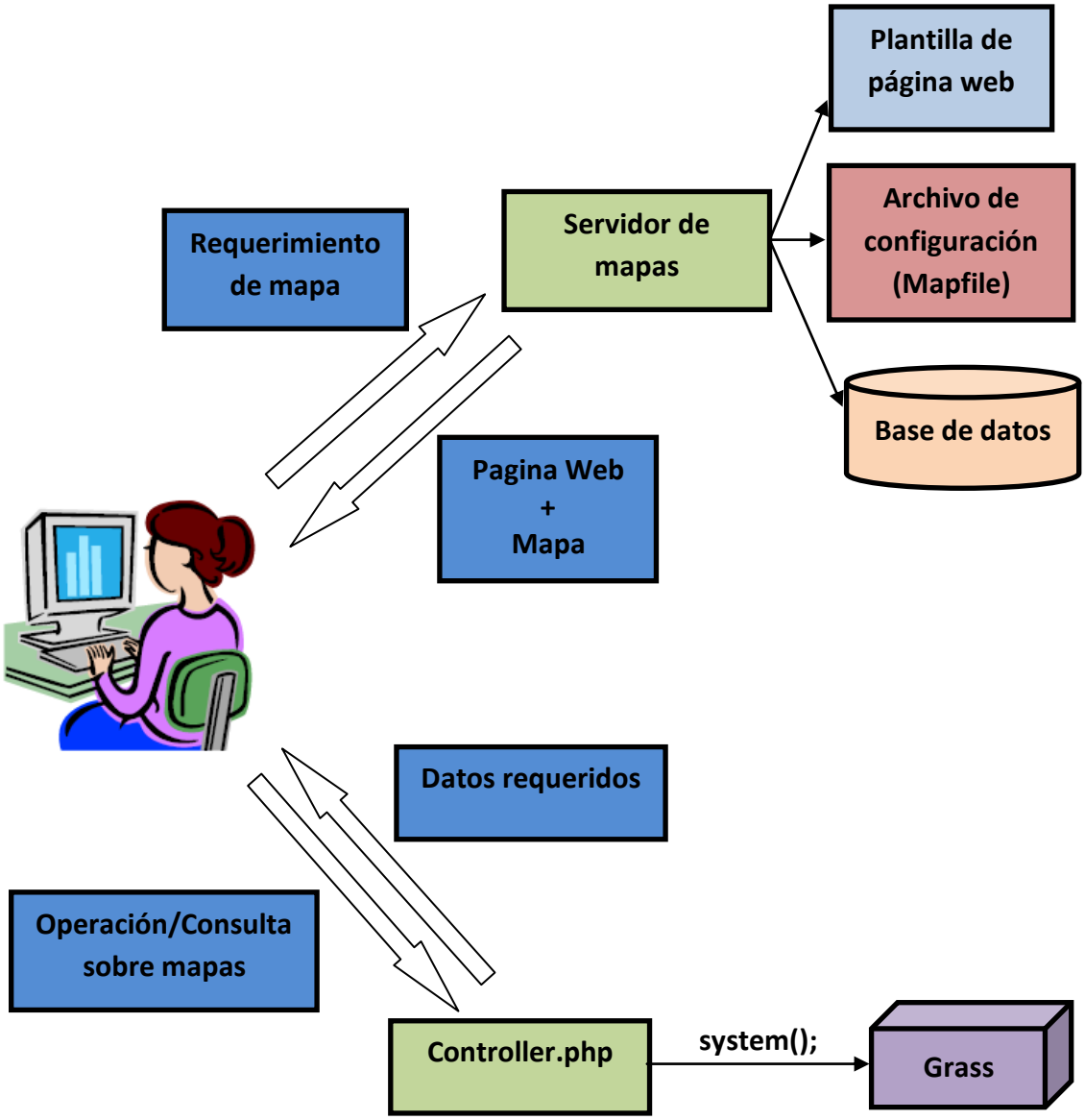
Recursos mineros:

El diseño de estos SIG facilitan el manejo de un gran volumen de información generada en varios años de explotación intensiva de un banco minero, suministrando funciones para la realización de análisis de elementos puntuales (sondeos o puntos topográficos), lineales (perfiles, tendido de electricidad), superficies (áreas de explotación) y volúmenes (capas geológicas). Facilitan herramientas de modelación de las capas o formaciones geológicas.

3. Diseño e implementación del sistema.

3.1. Diseño del sistema.

El esquema siguiente define la arquitectura del sistema implementado:



Donde:

- **Requerimiento de mapa:** incluye todas aquellas operaciones que dan como resultado una imagen de un mapa. Dichas operaciones engloba, evidentemente, a la operación de visualización de capas. Pero además se encuentran dentro de este grupo las operaciones de zoom y de desplazamiento (pan) sobre el mapa, puesto que el resultado de su aplicación también es una nueva imagen del mapa.

- **Plantilla de página web:** es una página web en html con la salvedad de que las rutas de mapas y operaciones no están instanciadas si no que tienen etiquetas entre corchetes que Mapserver leerá e interpretará para sustituirlas por las instancias necesarias.

- **Archivo de configuración (Mapfile):** Es un archivo con extensión .map. Se trata del fichero que leerá Mapserver para instanciar la mencionada plantilla. En él se detallan las capas que compondrán la web, se define la leyenda, la escala, la imagen del mapa de referencia,… Las capas se definen siguiendo una estructura de árbol como la que se muestra en el esquema.

- **Base de datos:** Define los mapas a usar. Para este ejemplo se empleará la base de datos **spearfish**. Además de la mencionada base de datos también se usarán archivos Geotiff.

- **Operación/Consulta sobre mapas:** incluye El resto de operaciones no incluidas en el apartado “Requerimiento de mapa”. Se invoca a alguna de estas funciones cuando lo que pretendemos es trabajar sobre los mapas existentes ya sea creando nuevas rutas o consultando los datos de las rutas existentes. En este caso no requeriremos que el sistema devuelva ninguna imagen asociada a un mapa.

- **Controller.php:** es el archivo que se encarga de realizar las llamadas al sistema necesarias para realizar las operaciones requeridas con grass.

- **Grass:** software encargado de realizar las operaciones de creación y consulta sobre mapas.

Si bien aquí se han explicado los elementos más importantes de forma sucinta, posteriormente se desarrollarán un poco más en profundidad los conceptos de Mapserver, Grass, Mapfile, el módulo controlador así como la plantilla HTML por tratarse estos de los pilares fundamentales de la aplicación.

A la vista del esquema y una vez aclarados sus componentes, estamos en condiciones de explicar el diseño del sistema.

Se podría decir que el servidor consta de dos núcleos. Por un lado está el programa CGI Mapserver. Éste programa se encarga de gestionar las operaciones de visualización sobre los mapas, (visualización de diferentes capas, realización de zoom, desplazamientos sobre el mapa,…).

La segunda parte activa del servidor es un script PHP (controller.php). Dicho script es el encargado de gestionar las peticiones (mediante llamadas al sistema) referentes a la creación y consulta de datos sobre los mapas. Es decir se recurrirá a este módulo para todas las operaciones que no tienen que ver explícitamente con la visualización de mapas. Si bien en algunas de las operaciones que ejecuta este módulo, como puede ser por ejemplo la selección de rutas, también se acaban visualizando mapas, hay que decir que no es el fin principal de esta función, si no antes bien, se persigue el aislamiento de las rutas seleccionadas para poder observar con mayor detalle sus características.

Si bien es cierto que Mapserver permite hacer consultas sobre los mapas, la decisión de montar una base de datos sobre Grass y que sea éste el encargado de instanciarla y consultar dichos datos se debe a la enorme potencia que esta alternativa posee, haciéndola mucho más versátil y personalizable que las posibilidades que a priori ofrecía Mapserver.

A partir de lo explicado se puede deducir como se implementarán las operaciones que proveerá el servidor.

En primer lugar la visualización de las diferentes capas del mapa se hará en base a peticiones al servidor Mapserver, el cual se encargará de instanciar el template html de la página principal indicándole la ruta con la imagen que ha creada tras la interpretación del mapa seleccionado a través de la mencionada petición.

Del mismo modo operan el resto de operaciones que sustentan su funcionamiento en Mapserver, siendo estas “Zoom In”, “Zoom out” y el desplazamiento sobre el mapa.

La creación de rutas en cambio no hace uso de Mapserver para nada. En su lugar, y como se explicará a continuación, se envía la petición al script PHP para que sea éste último el que mediante llamadas al sistema ejecute scripts de Grass que se encargarán en última instancia de la creación de mapas.

Mapserver no permite creación de nuevos mapas en el servidor, su tarea solo es de visualización, por lo que para incluir la nueva funcionalidad resulta imprescindible poder contactar en el servidor con un software que permita el tratamiento y la creación de mapas. Además, como ya ha sido comentado, esta filosofía de ejecución también se seguirá en la

consulta de datos sobre los mapas debido al mayor número de posibilidades y a la gran flexibilidad que presenta.

El mecanismo de exclusión mutua implementado es muy simple. Se basa en los autovalores de MySQL de forma que cuando se solicita la creación de una nueva ruta se comprueba el valor máximo de categoría existente en dicha base de datos e incrementa dicho valor en una nueva unidad. Este valor será usado como valor de categoría para la nueva ruta. Todos los mapas auxiliares se crearán nombrados con este valor.

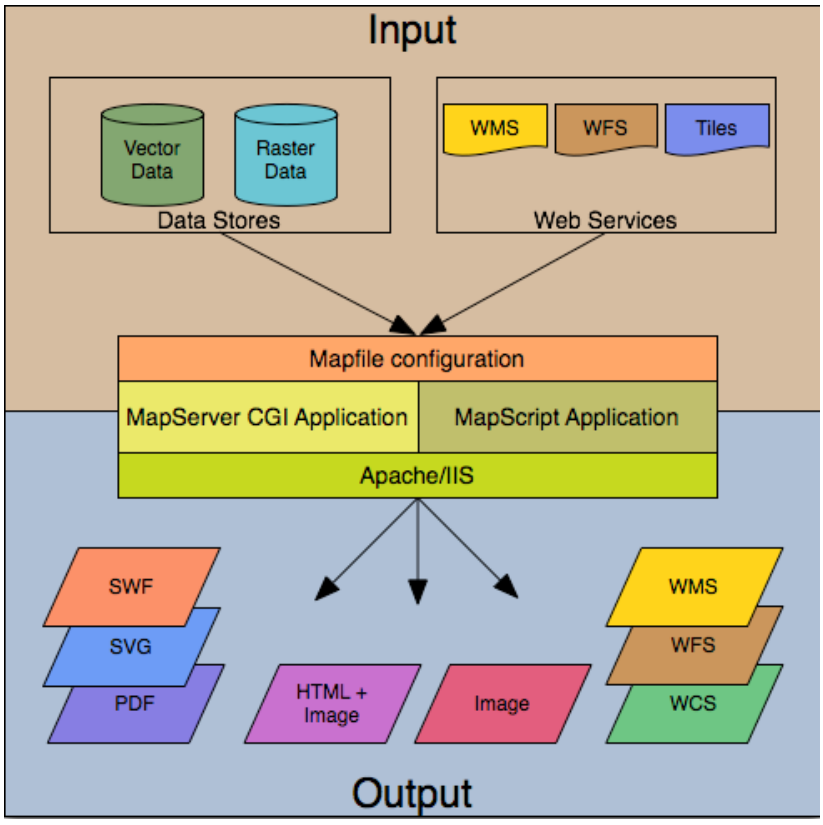
Estos autovalores se modifican de forma atómica, lo que garantiza la exclusión mutua entre los procesos. Este hecho además provoca la no interferencia en el proceso de creación de mapas al llamarse todos de forma diferente (el nombre será el valor de la mencionada variable).

3.1.1. Mapserver.

Mapserver es una aplicación CGI que permite dotar a las páginas de cierto dinamismo en cuanto que permite al usuario seleccionar de forma dinámica los mapas que quiere visualizar.

El funcionamiento es de lo más simple. Desde el cliente se le indica a Mapserver el mapa que se quiere visualizar. Este consultará en el conjunto de mapas de que dispone y apoyándose en el Mapfile, creará una imagen del mapa que será finalmente presentada al usuario. El archivo Mapfile se presentará en el apartado siguiente.

A continuación se expone un esquema de la "anatomía interna de Mapserver".



En este esquema se puede observar que Mapserver no solo puede leer los mapas del conjunto que tiene almacenado en el servidor. Además podrá consultar servicios web definidos al efecto los cuales permitirán la lectura de los ficheros solicitados.

El núcleo de la aplicación servidora es el que se encuentra en el centro de la imagen y en él se puede apreciar su funcionamiento en capas. En primer lugar la configuración del Mapserver es definida por el archivo Mapfile. Una vez definida la forma en la que el mapa se

mostrará la aplicación creará la imagen del mapa y Mapserver se encargará de instanciar el template html que será presentado al usuario a través de, como se puede ver en la imagen, un servidor Apache.

La aplicación CGI puede verse complementada por una aplicación Mapscript la cual puede estar sustentada en varios lenguajes. El más usual será PHP. Dicha aplicación permite gestionar las capas del Mapfile de forma dinámica y será comentada en el siguiente apartado.

Se puede observar también que la salida puede adoptar diferentes formatos. Por un lado está la mencionada instanciación de la web (HTML + Image). Sin embargo puede que de la salida solo nos interese la imagen. En este sentido hay que decir que mapserver siempre la almacena en el servidor, en la carpeta definida al efecto en el archivo Mapfile, por lo que podemos disponer de la imagen no solo para su visualización en la web. Así mismo es posible publicar los resultados en un servicio web o simplemente exportar a otros formatos como php.

Hay que decir que Mapserver toma las entradas que le proporciona el cliente al enviar el formulario al servidor a través de los métodos POST y GET. Existen multitud de variables del CGI. Aquí solo se mostrarán las más importantes. Si el lector desea ampliar dicha información puede encontrarla en los enlaces especificados en la bibliografía [1] y [3] entre otros.

Estas variables pueden ser referenciadas dentro del template a través del "name" de campos hidden si el valor será constante o del "name" de cualquier otro mecanismo html que permita instanciar su valor antes de enviarlo al servidor.

Si lo que se persigue es que Mapserver instancie dicha variable, esta deberá ir en el campo "value" o donde corresponda en el elemento html de la forma [*nombre*] donde *nombre* se refiere al nombre de la variable en cuestión. Mapserver interpretará dicha variable sistutiyendola por el valor instanciado. Las posibles etiquetas de sustitución más importantes que emplea Mapserver se comentarán en el apartado 2.3.5 correspondiente al archivo plantilla.

A continuación se explican algunas de las variables del CGI más importantes:

- **IMG:** Nombre asociado con la imagen del mapa que se verá y sobre la que se está trabajando.

- **IMGEXT [minx] [miny] [maxx] [maxy]:** La extensión espacial de la imagen actual, que corresponde a la imagen que el usuario ve actualmente en su explorador de internet.

- **IMGSIZE [columnas] [filas]:** EL tamaño (en pixeles) de la imagen actual.

- **IMGXY [x] [y]:** Coordenadas (in pixeles) de un click del mouse sobre la imagen.

- **LAYER [name]:** El nombre de un layer (capa) que aparece en el Mapfile. Enviando a mapserv el nombre de un layer, se establece como capa actuva para visualizarse (STATUS ON).

- **LAYERS [nombre nombre ...]:** Los nombres de los layers que serán activados para su visualización. Estos nombres deben estar separados por espacios. Permite visualizar varias capas a la vez a modo de "transparencias".

- **MODE [valor]:** Modo de operación. Algunos modos son:

    - **BROWSE:** Único modo válido en nuestro sistema. Corresponde a la interfaz donde se crean los mapas interactivos. El motivo de que sea el único modo válido se explicará más avanzado el texto en el apartado 7.4.

    - **QUERY:** ejecuta una búsqueda espacial de datos en el mapa.

- **REF:** Nombre asociado con la imagen del mapa de referencia.

- **ZOOMDIR [1|0|-1]:** Dirección del zoom. Un valor 0 indica que no se aplica zoom sobre la imagen. Por el contrario un valor 1 indica la aplicación de ZoomIn. Así el valor -1 corresponderá, como no puede ser de otra forma, con la aplicación de la operación ZoomOut.

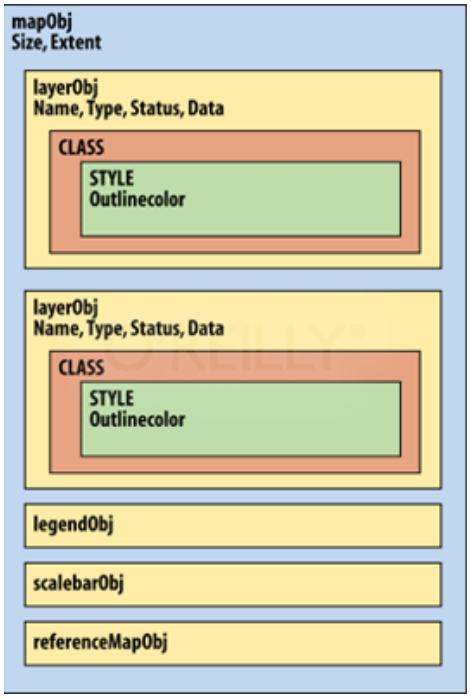
- **ZOOMSIZE [número]:** Valor absoluto del zoom. Usado con ZOOMDIR.

3.1.2. Mapfile.

En el fichero Mapfile se definen los datos a ser usados por nuestra aplicación. Esto es, en este fichero se definen la forma en la que Mapserver deberá presentar los datos al usuario. Además se le especifica dónde se encuentra la fuente de dichos datos para cada capa considerada, además de diversos parámetros de configuración que se desean que se carguen al principio.

Por tal motivo, podríamos decir que el corazón del Mapserver se configura a través de un archivo de texto, el Mapfile, el cual tiene extensión .map.

Bajo la perspectiva del programador se puede presentar como una jerarquía de objetos en la que se diferencia una parte principal de la que cuelgan diversas estructuras dependientes de esta. A continuación se muestra un esquema en el que queda más clara la estructura comentada.



Donde la etiqueta *"mapObj"* se refiere al Map Object. El Map Object no está explícitamente definido dentro del Mapfile si no que es este objeto el Mapfile propiamente dicho. Comienza a partir de la etiqueta **MAP** y finaliza con la cadena **END.** Es el padre de la jerarquía de objetos comentada previamente y define las características globales de la aplicación. Su ámbito es global.

De entre las etiquetas que definen las características más importantes del sistema podemos destacar:

- **NAME:** especifica el nombre que será usado para identificar la salida del sistema. Esta identificación será de la siguiente forma: un número identificativo generado a partir de la concatenación de la hora del sistema junto con el identificador de proceso. Este número además se concatenará con el valor de la etiqueta NAME objeto de estudio lo cual generará un identificador único. Podemos estar seguros de que el identificativo es único porque si bien, aunque altamente improbable, podría darse el caso de que dos usuarios enviasen una petición al servidor y esta se reciba justo a la vez el identificador de proceso será distinto para los dos clientes. El modo de ejecución de Mapserver es concurrente ocupándose de la gestión de cada petición procesos hijos, de ahí a que cada identificador de proceso sea diferente. Aún en el caso de que Mapserver operase de forma iterativa en el servidor, la unicidad de los nombres de los mapas también quedaría garantizada por el hecho de que en este caso la hora del sistema sería distinta para las dos peticiones.

- **STATUS:** Establece si el mapa está activo o no. Aunque a primera vista puede parecer que esta funcionalidad carezca de sentido hay que tener en cuenta que es posible que solo exista interés en generar la escala gráfica y la leyenda y no el mapa.

- **SIZE:** Definida como SIZE [int x] [int y]. Indica el ancho y el alto de la imagen de salida.

- **EXTENT:** Definida como EXTENT [minx] [miny] [maxx] [maxy]. Especifica la extensión del mapa en coordenadas terrestres. Estas coordenadas corresponden a la esquina inferior izquierda (minx,miny) y la esquina superior derecha (maxx,maxy). El ca´lculo de la escala requiere que los valores en coma flotante de las coordenadas representen las unidades especificadas en la cadena UNITS. La aparición de errores en la definición de la extensión puede dar lugar a mapas en blanco o erróneos. La medida debe ser definida obligatoriamente y debe estar en el mismo sistema de coordenadas que el objeto PROJECTION (si éste existe). En nuestro caso no aparecerá.

- **LAYER:** Indica el comienzo del objeto LAYER.

- **LEGEND:** Indica el comienzo del objeto LEGEND.
- **REFERENCE:** Indica el comienzo del objeto REFERENCE.
- **SCALEBAR:** Indica el comienzo del objeto SCALEBAR.
- **WEB:** Este objeto especifica la interfaz web, incluyendo paths, URLs, template files (puede haber más de uno) y demás detalles que afectan a la forma de responder de la aplicación frente a las peticiones del cliente. De entre los elementos más importantes que se pueden especificar en el objeto WEB podemos encontrar:
  - **IMAGEPATH:** Especifica el path del directorio donde Mapserver almacenará las imágenes que irá creando en su funcionamiento. Puesto que cada vez que el usuario acceda al sistema y trabaje con él Mapserver generará nuevas imágenes ignorando las creadas anteriormente (aún en el caso de que sean iguales) es recomendable que el administrador del sistema limpie de manera periódica ese directorio a fin de evitar la sobrecarga de recursos en cuanto a capacidad de almacenamiento se refiere.
  - **IMAGEURL:** Se refiere al mismo directorio que en el caso anterior pero en esta ocasión lo que se especificará no será la ruta absoluta al directorio si no su URL. Si bien el caso anterior se especificaba la ruta para que el programa servidor supiera dónde acceder dentro del servidor, en este caso es necesario especificar este parámetro con objeto de que el navegador conozca la dirección dentro del servidor.
  - **MINSACALE:** Especifica el valor mínimo de escala con el que los mapas serán devueltos por parte del servidor. Peticiones de menor escala no serán tenidas en cuenta devolviendo en este caso el servidor el mapa con este valor de escala.
  - **MAXSCALE:** Especifica el valor máximo de escala con el que los mapas serán devueltos por parte del servidor. Peticiones de mayor escala no serán tenidas en cuenta devolviendo en este caso el servidor el mapa con este valor de escala.
  - **TEMPLATE:** Especifica la plantilla HTML principal del proyecto.

De los mencionados además debemos hacer hincapié en LAYER, LEGEND, REFERENCE y SCALEBAR, los cuales merecen un tratamiento a parte, al igual que el objeto MAP, debido a su importancia y serán comentados a continuación. El primero de los comentados será el objeto LAYER. Dicho objeto comienza con la etiqueta LAYER y finaliza con la etiqueta END y determina qué datos especiales serán renderizados y la forma en que estos se clasificarán. Es necesario tener cuidado para que las características definidas para cada capa no entren en conflicto con las características globales definidas en el objeto MAP. De las características más importantes que Mapserver permite definir dentro del objeto LAYER se encuentran:

- **NAME:** Especifica el nombre de la capa. Este valor se usa en conexión con el valor del formulario *layer* especificado en el archivo de plantilla. Mediante él el CGI activará y desactivará la capa de forma interactiva respondiendo así a peticiones por parte del cliente.

- **TYPE:** especifica el tipo de capa, el cuál determina como debe ser presentada la capa. Entre los tipos permitidos podemos encontrar, *raster, line, polygon,…*

- **STATUS:** Especifica si la capa se mostrará o no. Puede presentar tres valores diferentes, siendo estos los siguientes:

    - **DEFAULT:** Indica que la capa siempre estará visible.

    - **ON:** Indica que la capa se mostrará.

    - **OFF:** Indica que la capa no se mostrará.

- **DATA:** Especifica el path hacia el mapa que será interpretado en esta capa y que posteriormente será mostrado. Para cambiar el conjunto de mapas objeto de estudio se deberá tener cuidado a la hora de actualizar si fuera necesario este valor.

- **TEMPLATE:** En caso de usar el modo QUERY que proporciona Mapserver, esta etiqueta del objeto LAYER especifica el path completo del archivo usado para mostrar los resultados de la consulta sobre la capa en cuestión. En este caso no aparecerán puesto que, como ya se ha mencionado, no se usa este modo de ejecución de la aplicación CGI.

- **CLASS:** Indica el comienzo del objeto CLASS. Este objeto determina la apariencia y las propiedades el etiquetado de las clases que aparecen en el mapa. Cada capa debe

especificar una o más clases. De entre las etiquetas más importantes que podemos encontrar en el objeto CLASS están:

- **NAME:** El nombre de la clase se usa en la leyenda del mapa. Si no se especifica ningún nombre la clase correspondiente del mapa se dibujará pero no aparecerá en la leyenda.

- **COLOR:** Especifica el color con el que se dibujará la clase en cuestión. Los valores serán especificados en RGB mediante el empleo de tres valores enteros en el rango [0-255].

- **LABEL:** Indica el comienzo del objeto LABEL. Este objeto define una cadena de texto o un símbolo que representará una característica del mapa representado. Entre los objetos dependientes de éste podemos destacar:

  - **POSITION:** Indica la posición de la etiqueta respecto del punto representado. Un posible valor de esta característica interesante para comentar corresponde a *auto*. Con este valor Mapserver situará la etiqueta en un punto donde no colisione con etiquetas previamente dibujadas. Si no es posible encontrar dicho lugar la etiqueta no se dibujará a menos que se especifique lo contrario en el objeto FORCE dependiente también de LABEL.

  - **SIZE:** Indica el tamaño del texto.

  - **COLOR:** Indica el color de la etiqueta.

A continuación se presentará el objeto LEGEND. Este objeto, como es fácil de prever, se encargará de la gestión de las características que presentará la leyenda del mapa, esto es, la apariencia de la misma en la web así como su posición. Como se ha comentado anteriormente, las clases en las que se ha especificado un nombre se incorporarán a la leyenda no ocurriendo lo mismo con aquellas clases sin nombre especificado. Una característica distintiva de este objeto es que el tamaño de la leyenda no se conoce hasta que no se dibuja puesto que no se dispone a priori del número de capas que se van a dibujar hasta que el usuario no las notifique explícitamente. De entre las características más importantes que puede presentar el objeto LEGEND podemos destacar:

- **KEYSIZE:** Indica el tamaño, en píxeles, de las cajas que contendrán los símbolos de la leyenda.

- **LABEL:** Mismo caso que la etiqueta LABEL comentada anteriormente.

Otro objeto destacable del archivo Mapfile se refiere al dibujo que indicará la escala del mapa. Elemento de suma importancia a la hora de interpretar correctamente los datos mostrados en el mismo. Este objeto se denotará por la etiqueta SCALEBAR y sus parámetros más importantes serán:

- **LABEL:** Igual que en apartados precedentes.

- **STYLE:** Especifica el estilo de la escala. Solo soporta dos valores, 0 y 1.

- **COLOR:** Especifica el color de fondo de la escala. Se denota en RGB mediante tres valores en el rango [0-255].

- **UNITS:** Indica las unidades en las que se expresa la escala. Si estas son diferentes a las unidades especificadas para el mapa, la conversión se realizará automáticamente.

- **INTERVALS:** Indica el número de intervalos mostrados en el dibujo de la escala.

El último de los objetos comentados será el referente al mapa de referencia el cual se notará por la etiqueta REFERENCE. Y, como viene siendo habitual, especificará las características del objeto en cuestión, en este caso el mapa de referencia. Los mapas de referencia muestran el contexto sobre el que se está trabajando, dibujando una imagen genérica de la zona en cuestión sobre la que se enmarca la zona concreta sobre la que se está trabajando. Mapserver ofrece la posibilidad de hacer este mapa interactivo proporcionándole los mismos controles que al mapa convencional. Esta posibilidad será aprovechada también en nuestro sistema. De entre las características más destacables de este objeto podemos encontrar:

- **EXTENT:** Indica la extensión espacial del mapa de referencia.

- **MINBOXSIZE:** Indica el tamaño mínimo, en píxeles, de la caja que señala la región de trabajo. Si el tamaño se hace inferior al aquí especificado se empleará el símbolo especificado en MARKER en lugar de la mencionada caja.

- **MAXBOXSIZE:** Indica el tamaño mínimo, en píxeles, de la caja que señala la región de trabajo.

- **COLOR:** Indica el color de fondo de la caja que marca la zona activa. El valor se especifica en RGB a través de tres enteros en el rango [0-255]. El valor -1 -1 -1 indica que dicho color no aparecerá (transparente).

- **OUTLINECOLOR:** Color del marco de la caja que indica la región activa.

- **MARKER:** Indica el símbolo que se utiliza cuando la caja a la que se ha hecho alusión se hace demasiado pequeña.

Si el lector está interesado ampliar en el conocimiento de las variables existentes en archivo Mapfile se le recomienda la consulta de la referencia bibliográfica [1].

El archivo Mapfile presenta no obstante una importante desventaja y esta es la rigidez impuesta por el hecho de que dicho archivo es estático. Es decir, el número de capas definido es estático y fijo lo que impedirá la creación dinámica de nuevas capas.

El PHP Mapscript rompe ésta rigidez del .map (pues carga las capas configuradas en él al inicializar) y se podrá modificar, cambiar e incluso agregar más capas según se necesite. Ésto significa que si tenemos una capa de color verde inactiva, podemos modificarla a rojo y activarla. Por supuesto las posibilidades de operación de Mapserver no se verán reducidas por el PHP Mapscript, antes bien complementadas.

En el caso que aquí nos ocupa se ha optado por implementar la versión estática basada en el archivo Mapfile obviando, por tanto, la implementación de la aplicación Mapscript. La razón por la que se ha tomado esta decisión es que esta segunda opción complicaba bastante tanto el diseño como la implementación de la aplicación y una vez hecho un análisis en profundidad de los requisitos del sistema (los cuales se presentarán en el apartado 4) es fácil darse cuenta de que en realidad la implementación de la mencionada opción no es absolutamente necesaria para el desarrollo de la misma. Por lo tanto puesto que con la opción estándar de funcionamiento del CGI basada en el archivo Mapfile el funcionamiento de la aplicación no se verá mermado se ha optado por implementar la opción de menor coste en cuánto al tiempo y a recursos se refiere.

Además hay que tener en cuenta que si en un futuro nuevos requisitos funcionales del sistema requiriesen la implementación del Mapscript este podría hacerse sin mayor esfuerzo que el que supone la implementación de los métodos correspondientes no siendo necesaria la modificación del archivo Mapfile. Como es evidente será necesario dotar al servidor así como a la aplicación Mapserver de la capacidad de trabajo con el lenguaje sobre el que se desee implementar la aplicación Mapscript. El archivo template también habrá que modificarlo de la forma en la que se ha indicado anteriormente.

3.1.3. Grass.

En sus inicios, en 1982, fue desarrollado por el Cuerpo de Ingenieros del Laboratorio de Investigación de Ingeniería de la Construcción del Ejército de los Estados Unidos (USA-CERL) como herramienta para la supervisión y gestión medioambiental de los territorios bajo administración del Departamento de Defensa al no encontrar ningún GIS en el mercado que satisficiese estas necesidades. En 1991 se pone a disposición pública a través de Internet. Su popularidad se incrementa en universidades, empresas y agencias gubernamentales. En 1997, ante el anuncio de USA-CERL GRASS de que dejaría de dar soporte al programa, la Universidad de Baylor se hace cargo de su desarrollo. A partir de esta fecha aumenta su aceptación dentro del mundo académico. El 26 de octubre de 1999 con la versión 5.0 se libera el código del programa bajo licencia GNU GPL. GRASS era uno de los primeros ocho proyectos de la Fundación OSGeo. No fue hasta el pasado 2008 cuando oficialmente se dio por finalizada la fase de incubación.

Actualmente Grass se encuentra en la versión 6.7, aunque dicha versión aún no es estable. En el momento de comenzar el desarrollo de la aplicación la última estable era la versión 6.4. Es por ello por lo que será la empleada en el sistema. La sustitución de Grass por versiones posteriores en un futuro será fácil sin más que instalar el nuevo software en la misma ubicación prestando especial atención a los permisos asignados. Otro aspecto a tener en cuenta por parte del administrador del sistema será la necesidad de comprobar que las operaciones usadas en el sistema deberán llamarse de la misma manera, debiendo proveer este los mecanismos de traducción en tanto que este requisito no se cumpliera.

Esta herramienta permite el análisis de mapas tanto 2D como 3D. Así mismo permite el tratamiento de mapas tanto vectoriales como raster almacenados en distintos formatos. Si el lector desea ampliar más información respecto de los formatos de mapas soportados por Grass se le recomienda que consulte la bibliografía, en concreto las referencias [4] y [5].

Además del análisis geoespacial de los mapas, Grass permite la creación y edición de mapas.

Para los fines mencionados Grass posee más de 350 programas diferentes que dotan al conjunto de gran potencia en el cálculo y análisis de mapas. Además el hecho de que sea una herramienta de código libre hace que esté en continuo cambio y evolución permitiendo que las prestaciones ofrecidas aumenten a muy buen ritmo, lo que unido a la gran repercusión que este campo tiene en la actualidad, dota a esta herramienta de gran relevancia en el panorama de los GIS actuales. Es por estas razones principalmente por las que se ha optado por el uso de esta herramienta para el análisis y la creación de mapas en el sistema aquí descrito.

Como se ha comentado en alguna ocasión anterior, cada uno de los scripts de Grass es independiente del resto encontrándose estos integrados dentro de una interfaz de administración común.

Este sistema aprovechará esta posibilidad, llamando a los scripts necesarios de forma aislada sin necesidad de ejecutar todo el programa completo. Para ello se deberán instanciar en el módulo controlador una serie de variables globales usadas por Grass y las cuales serían instanciadas en su arranque en el sistema.

En este sistema se aprovechará toda la funcionalidad principal de Grass, esto es, visualización de mapas, creación de mapas, análisis de mapas y gestión de la base de datos.

Anteriormente se ha dicho que para la visualización de los mapas se empleará la herramienta Mapserver, ¿cómo es posible entonces que Grass se emplee también para la visualización? La respuesta a esta pregunta es bien sencilla si se atiende a una de las decisiones tomadas en el apartado del diseño de las operaciones y está estrechamente relacionada con el proceso de creación de mapas. Como se comentará en el apartado correspondiente a las operaciones, en concreto a la operación de creación de mapas. En este caso, una vez creado el mapa, se procederá a la creación de 2 imágenes. La primera correspondiente a la ruta creada sobre el mapa de elevación del terreno y el de carreteras y la segunda corresponderá al perfil de la ruta. Esta segunda imagen no es explícitamente creada por Grass, si no por el programa gnuplot. Lo que hace Grass en este caso es proveer a dicho programa de los datos necesarios para su creación extraídos estos del mapa en cuestión. El hecho de que la primera imagen se cree con Grass es simplemente por comodidad y rapidez en la ejecución, evitando así tener que enviar una petición entre programas (del módulo controlador al CGI Mapserver) una vez creado el mapa.

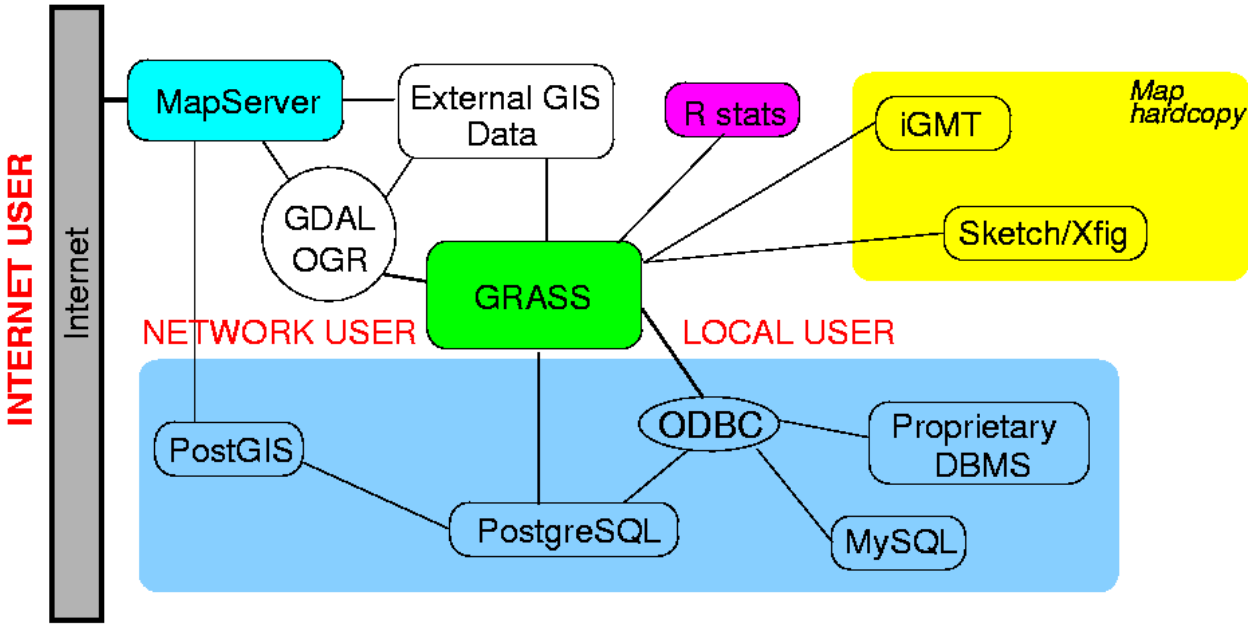
Cada vez que un mapa se crea Grass deberá actualizar la base de datos que usa el sistema y que depende de él (implementada a través del driver dbf) [7]. Para actualizar dicha base de datos previamente es necesario extraer los datos de interés del mapa en cuestión mediante un análisis del mismo.

La razón para usar la base de datos que provee Grass en lugar de una externa que proporcione una mayor potencia en el almacenamiento de datos y por consiguiente mayor número de posibilidades se debe nuevamente a una estrategia de diseño. Si bien es cierto que una base de datos implementada sobre un servicio MySQL o sobre PostgreSQL, dotaría al sistema de mayor capacidad de almacenamiento, también es cierto que a la luz de los requisitos del sistema esta no se hace necesaria, lo cual provocaría un esfuerzo y una complicación extra a la hora de instalar el sistema así como un mayor consumo de recursos mientras éste se encuentra en funcionamiento. Esfuerzo y consumo que, como se ha dicho, no

serán necesarios por cuánto que nos será suficiente con usar la base de datos que gestiona Grass para satisfacer las necesidades de este sistema en cuestión.

Puesto que la inserción de datos en la base de datos se hace en base a la utilización de comandos de Grass, resulta evidente entonces que las consultas sobre la misma también se harán en base al uso de estos comandos. Si bien no corresponde a este apartado comentarlos. Esto se hará en el apartado correspondiente a las operaciones.

A continuación se muestra un esquema que puede servir a modo de ejemplo del uso de Grass en el sistema.



En él se puede observar cómo trabajan conjuntamente Grass y Mapserver a través de la librería GDAL-OGR. En este sistema se hará uso de la librería GDAL-GRASS que permite a Mapserver leer mapas raster en el formato definido por Grass.

También se puede ver como la salida del sistema se puede redirigir hacia una base de datos o bien hacia el propio sistema proveyendo a este de datos necesarios para su funcionamiento, en este caso mapas.

3.1.4. Módulo controlador.

Esta es una de las grandes novedades de este sistema. El módulo controlador es un script en PHP que se ejecutará en el servidor llevando a cabo funciones específicas solicitadas por el cliente.

No se trata de una aplicación CGI como en el caso de Mapserver aunque su efecto puede ser el mismo, dotar a una página de cierto dinamismo. Este script se encargará de llevar a cabo todas las operaciones que para las que el sistema necesite Grass, esto es, para la creación de mapas y las consultas de datos sobre los mismos.

El proceso seguido es el siguiente: el usuario envía el formulario al servidor. Previamente se comprueba en el cliente mediante Javascript el tipo de operación del cual se trata a fin de modificar convenientemente el *"action"* del formulario, esto es, dejando su valor por defecto y que la petición sea enviada al CGI Mapserver o bien modificarla para que esta se redirija al módulo objeto de estudio.

Suponiendo que la petición ha sido redirigida correctamente en el módulo controlador se recogerán las variables enviadas a través del método GET. Lo primero que se hará será comprobar la operación que se pretende realizar para, una vez determinada, proceder al desarrollo de la misma.

Este desarrollo se realizará a través de llamadas al sistema mediante el comando *"system()".*

Para que los comandos Grass surtan efecto anteriormente es necesario "simular" el arranque de Grass en el sistema, esto es, Grass no estará ejecutándose permanentemente en el servidor, ello permitirá valernos solo de los comandos necesarios liberando así al servidor del elevado consumo de recursos que supondría tener una instancia completa de Grass ejecutándose para cada usuario.

Para llevar a cabo esta opción nos valdremos del hecho de que la funcionalidad de Grass se compone de programas independientes gestionados por la autoridad principal de Grass. Lo que se ha planteado en este proyecto es erigir al módulo *"Controller.php"* como dicha autoridad. Para ello bastará, en primer lugar, con dotar a la carpeta donde se encuentra Grass de los permisos necesarios para poder ser ejecutado por un servidor web. En segundo lugar será necesaria la instanciación de una serie de variables globales que Grass usa habitualmente en su configuración. Estas variables se refieren normalmente a rutas de acceso a distintos recursos del sistema usados por el programa.

Este módulo se emplea siguiendo la misma filosofía de diseño que se ha mantenido en la implementación del sistema y no es otro que el Modelo-Vista-Controlador. En este caso el modelo de datos corresponde al conjunto de mapas sobre el que se sustenta la aplicación así como la base de datos definida al efecto y relacionada con dicho conjunto de mapas y de la cual se habló en el apartado 2.3.3. Así mismo la vista corresponderá tanto al archivo de plantilla del que se hablará en el apartado siguiente como a la vista de impresión que se le plantea al usuario en la realización de determinadas consultas. De dicha vista se hablará en apartados posteriores, cuando se trate el tema de las operaciones. Así mismo el núcleo controlador se encargará de gestionar las operaciones llevadas a cabo en el servidor. En este caso dicho núcleo, como ya se comentó, es doble. Por un lado se encuentra el programa CGI Mapserver y por otro el módulo controlador objeto de estudio.

3.1.5. Template HTML.

El archivo template no es más que una página web escrita, como no podía ser de otro modo en HTML.

La particularidad del mencionado archivo es que los valores de algunos de sus elementos se encuentran sin instanciar como por ejemplo la ruta donde se encuentra la imagen del mapa que se muestra.

Tal y como se expone en la referencia [2] dos son los usos principales de las plantillas, y estos son:

- Definir el aspecto de la interfaz de una aplicación CGI de Mapserver.
- Para presentar los datos de una consulta.

Como se ha mencionado las plantillas suelen ser archivos HTML aunque también se permite trabajar con URLs, esto es, http://www.somewhere.com/ [atributo] / info.html.

La instanciación de la plantilla requiere la petición de la misma al servidor. Para hacer esta operación transparente al usuario y que al entrara en la web esta la vea ya instanciada lo que se hace es realizar esta petición al servidor en la página de inicio (index.html) la cual redireccionará directamente la petición al servidor el cual instanciará el archivo plantilla que será finalmente mostrado al usuario.

El resto del funcionamiento de la página es exactamente igual que una página web pues no se puede olvidar el hecho de que no es más que eso al fin y al cabo. De este modo las plantillas HTML pueden incluir cualquier cosa relacionada con la programación web, como por ejemplo, JavaScript.

Las cadenas de sustitución se denotan de la forma [*cadena*] de forma que Mapserver interpretará dichas estructuras como etiquetas que deberá sustituir o, siguiendo con el mismo lenguaje empleado, instanciar.

De entre las cadenas más comunes de sustitución podemos encontrar:

- **[img]:** Ruta de acceso (en relación al documento raíz) de la nueva imagen. En una interfaz de mapa de plantilla, [img] es sustituido por la ruta a la imagen del mapa. En una plantilla de resultados de la consulta, se sustituye por la ruta a la imagen querymap.

- **[ref]:** Ruta de acceso (en relación al documento raíz) de la imagen de referencia.
- **[legend]:** Ruta de acceso (en relación al documento raíz) de la imagen nueva leyenda.
- **[scalebar]:** Ruta de acceso (en relación al documento raíz) de la imagen nueva barra de escala.
- **[mapx], [mapy]:** Coordenadas X e Y del click del ratón sobre el mapa.

Para ampliar más información sobre las etiquetas disponibles el lector debería consultar las referencias [1] y [2].

3.2. Implementación del sistema.

El servidor del que se hará uso será apache. Para facilitar su instalación se ha utilizado el software Xampp el cual ya contiene agregados MySQL, PHP, Perl,…

Además, como se ha mencionado anteriormente, se utilizará el software Mapserver para la visualización de mapas en el cliente y GRASS para trabajar con ellos en la edición. Para que Mapserver sea capaz de tratar con los mapas que usa GRASS hay que tener instalada la librería GDAL-Grass junto con Mapserver.El hecho de que Grass esté compuesto de "scripts" independientes programados en C hace posible poder invocar de forma aislada cada uno de ellos mediante llamadas al sistema en php, implementadas mediante la función "system()". Para que dichas llamadas surtan el efecto deseado es necesario cargar previamente una serie de variables de entorno que Grass requiere normalmente al arrancar. Estas variables se cargarán en el módulo controller el cual hará uso de Grass.

La estructura de directorios del sistema es la que sigue: Al mismo nivel que la raíz del servidor (htdocs) se crea una carpeta llamada map-script la cual contendrá los archivos mapserver.map (archivo de configuración) y marker.sym que contiene información de la proyección,… En la carpeta cgi-bin se sitúan los programas de Mapserver que se ejecutarán en el servidor. En la carpeta htdocs además estará el fichero de base de datos (spearfish) y una carpeta tmp donde se guardarán las imágenes generadas por el servidor. Además se dispone de una carpeta llamada profile en la cual se guardan los perfiles de rutas creadas por los usuarios. Por último en la carpeta images_Rutes se guardan las imágenes de las rutas creadas. Por último cabe mencionar que existe un directorio donde se almacenarán las coordenadas de las rutas una vez creadas. Estas coordenadas y las imágenes asociadas a las rutas se indexan de acuerdo al valor de la categoría de la ruta.

4. Instalación y configuración del servidor.

Como se ha mencionado anteriormente el servidor utilizado será Apache. Para facilitar tanto su instalación como su configuración con mysql y el lenguaje PHP se ha optado por instalar una distribución de Xampp, la cual integra perfectamente dichos elementos.

El siguiente paso será la instalación del programa GRASS en la máquina que actuará como servidor. Es importante mencionar que una vez instalado el programa será necesario dar los permisos necesarios para permitir su ejecución desde fuera. Para una primera versión se habilitarán todos los permisos.

Antes de proceder a instalar el software de Mapserver hay que instalar la biblioteca GDAL-GRASS la cual nos permitirá trabajar con Mapserver haciendo uso de mapas almacenados en el formato con el que trabaja GRASS.

En esta circunstancia ya estamos en disposición de instalar el Mapserver cuyo ejecutable deberá ser almacenado en la carpeta “cgi-bin” del servidor.

Una vez instalado todo el software y creadas las carpetas necesarias los permisos serán:

- Carpeta **/opt:** estarán todos los permisos habilitados siendo el propietario **nobody** y el grupo **root.**
- Carpeta **/grass:** nuevamente estarán todos los permisos habilitados siendo el propietario **nobody** y el grupo **root.**
- Carpeta **cgi-bin:** habilitados todos los permisos siendo el propietario **nobody** y el grupo **root.**
- Carpeta **map-script:** habilitados todos los permisos siendo el propietario **nobody** y el grupo **root.**
- Carpeta **htdocs:** los permisos habilitados por defecto siendo el propietario **nobody** y el grupo **root.**
- Carpeta **Proyecto:** los permisos habilitados por defecto siendo el propietario **nobody** y el grupo **root.**
- Archivo **gisrc:** habilitados todos los permisos siendo el propietario **nobody** y el grupo **root.**

- Archivo **.grassrc5:** habilitados todos los permisos siendo el propietario **nobody** y el grupo **root.**

- Carpeta **spearfish:** habilitados todos los permisos siendo el propietario **nobody** y el grupo **www.**

- Carpeta **tmp:** habilitados todos los permisos siendo el propietario **nobody** y el grupo **www.**

Todos estos cambios serán aplicados de forma recursiva a todas las carpetas que cuelgan de las aquí mencionadas.

5. Análisis de requisitos.

Basándonos en el diseño de la arquitectura comentado en el apartado 2 a continuación se procederá a mostrar los requisitos tanto funcionales como no funcionales de la aplicación en cuestión.

5.1. Requisitos no funcionales.

- El sistema será usado por cualquier persona que desee su consulta y disponga de un navegador web y de una conexión a internet por lo que su interfaz deberá ser intuitiva.
- El sistema se basará en el uso de herramientas de software libre.
- El propio sistema será finalmente una herramienta de software libre.
- El sistema debe permitir de forma cómoda la sustitución del conjunto de mapas sobre el que se trabajará.
- El sistema deberá ser extensible permitiendo de forma cómoda la incorporación de nuevas capas.
- Será el administrador de sistemas el encargado de la incorporación de nuevas capas así como de la sustitución del conjunto de mapas.
- El servidor deberá cubrir los requisitos de almacenamiento con capacidad para almacenar el conjunto de mapas y previendo el aumento del espacio en disco debido al aumento del número de rutas editadas por el usuario.
- Será necesario el empleo de un procesador de al menos 2 GHz para que así el proceso de creación de mapas se realice en el menor tiempo posible.
- El sistema deberá implementarse sobre el sistema operativo Linux.
- El funcionamiento del sistema deberá estar optimizado con el estándar Javascript no programándose servicios para otras plataformas.
- El sistema deberá estar optimizado para su funcionamiento en el navegador Mozilla Firefox.

- El sistema deberá trabajar rápidamente y el tiempo de respuesta deberá ser el mínimo posible.

- El uso del software por parte del cliente final no necesitará de ninguna instalación ni preparación previa.

- La indexación de cada ruta se hará por su número de categoría.

- La base de datos empleada será MySQL.

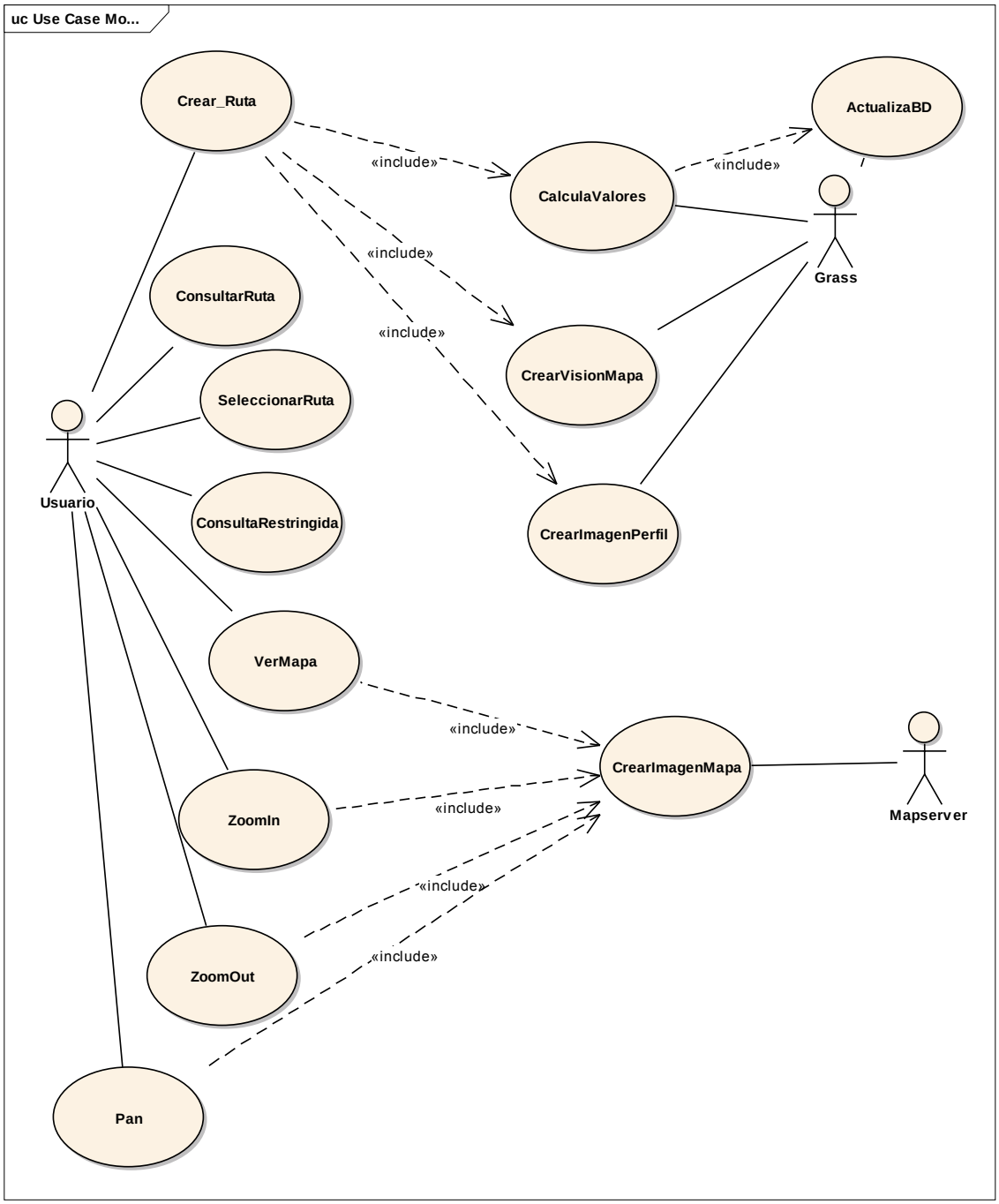
- A la base de datos se accederá a través del driver dbf de Grass.

- La información deberá ser mostrada al usuario de forma clara.

5.2. Requisitos funcionales.

- El sistema debe permitir que los usuarios creen nuevas rutas.
- Las rutas creadas por el usuario deberá almacenarse en el sistema.
- Por cada ruta deberá extraerse la información de longitud, altura media y pendiente máxima del terreno. Esta información se almacenará en la base de datos del sistema.
- Para cada ruta deberán generarse dos imágenes que se almacenarán en el servidor. Una del perfil de ruta y otra que consistirá en la visión de la ruta en el entorno de elevación en el que se encuentra, acompañada a su vez del mapa de carreteras de la zona.
- El sistema debe permitir que los usuarios consulten datos sobre las rutas ya existentes.
- El sistema debe permitir que los usuarios seleccionen un conjunto de rutas de las existentes.
- El sistema debe permitir que los usuarios puedan imprimir las rutas seleccionadas.
- El sistema debe permitir que los usuarios puedan buscar rutas en base a los parámetros conocidos de dichas rutas, esto es: nombre de la ruta, altura media de la ruta, pendiente máxima del terreno y longitud de la mencionada ruta.
- El sistema debe permitir aumentar o disminuir el zoom sobre la imagen.
- El sistema debe permitir el desplazamiento sobre la imagen cambiando así la zona visible del mapa.
- El sistema debe permitir el restablecimiento de la situación original cuando el cliente lo desee.

6. Diagrama de casos de uso.



7. Descripción de los casos de uso.

Nombre: Crear_Ruta.

Resumen: El usuario podrá dibujar la ruta que desee sobre el mapa que elija.

Dependencias: Incluye al caso de uso "CalculaValores", al caso de uso "CrearVisionMapa" y al caso de uso "CrearImagenPerfil".

Actores: Usuario (Iniciador), Grass.

Precondiciones: El mapa de rutas debe existir previamente.

Postcondiciones: La ruta queda almacenada en la base de datos. La imágen del mapa así como la del perfil de la ruta queda almacenada en el servidor

Curso Normal:

10. El Usuario selecciona la operación "Draw Rutes" de la página web.

20. El usuario introduce el nombre de la ruta.

30. El usuario dibuja la ruta sobre el mapa y pulsa el botón "Save Map".

4**0.** Caso de uso CalculaValores.

50. Caso de uso CrearVisionMapa.

60. Caso de uso CrearImagenPerfil.

Observaciones: Si el usuario no introdujera ningún nombre para la ruta a esta se le asignará por defecto el nombre "Ruta sin nombre". No obstante deberá pulsar el botón "submit" para poder seguir con la operación.

Requisitos no Funcionales Específicos: El sistema deberá trabajar rápidamente y el tiempo de respuesta deberá ser el mínimo posible.

Nombre: ConsultarRuta.

Resumen: El usuario podrá consultar la ruta que desee en el mapa.

Dependencias: Ninguna

Actores: Usuario.

Precondiciones: Debe existir la ruta previamente.

Postcondiciones: Ninguna.

Curso Normal:

10. El usuario marca la operación correspondiente “Query rutes”.

20. El usuario marca dos puntos sobre el mapa que delimitarán un cuadrilátero en el que se encuadra la ruta a consultar.

30. Se muestra la información de la ruta elegida. Además se mostrará también la imagen del perfil de la ruta.

Cursos Alternativos:

a) No se selecciona ninguna ruta

30. El sistema devuelve un mensaje de error.

40. El usuario deberá volver a la página de inicio.

b) Se selecciona más de una ruta

30. El sistema devuelve un mensaje de error.

40. El usuario deberá volver a la página de inicio.

Observaciones: Ninguna

Requisitos no Funcionales Específicos: La información debe ser mostrada al usuario de forma clara.

Nombre: SeleccionarRuta.

Resumen: El usuario podrá seleccionar un conjunto de rutas que el sistema deberá procesar y mostrar al usuario para que este pueda ver la información desde otra perspectiva e incluso imprimir esta información.

Dependencias: Ninguna.

Actores: Usuario.

Precondiciones: Debe existir la ruta previamente.

Postcondiciones: Ninguna.

Curso Normal:

10. El usuario seleccionará la operación adecuada, la correspondiente a “Select rutes”.

20. El usuario marca dos puntos sobre el mapa que delimitarán un cuadrilátero en el que se encuadra la o las ruta a seleccionar.

30. Se muestra la información de las rutas elegida en una nueva página. Además se mostrará, para cada ruta, también la imagen del perfil de la ruta. Junto con la imagen del mapa encuadrado en su entorno y el mapa de carreteras del lugar.

Cursos Alternativos:

a) No se selecciona ninguna ruta

30. El sistema devuelve un mensaje de error.

40. El usuario deberá volver a la página de inicio.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: La información debe ser mostrada al usuario de forma clara.

Nombre: ConsultaRestringida.

Resumen: El usuario podrá seleccionar un conjunto de rutas las cuales estarán caracterizadas por cumplir una serie de restricciones definidas por el usuario.

Dependencias: Ninguna.

Actores: Usuario.

Precondiciones: Deben existir las rutas previamente.

Postcondiciones: Ninguna.

Curso Normal:

10. El usuario seleccionará la operación adecuada, la correspondiente a "Restricted query rutes".

20. El usuario especificará los parámetros deseados sobre los que se basará la búsqueda de rutas.

30. Se muestra la información de las rutas que cumplen con las restricciones impuestas por el usuario en una nueva página. Además se mostrará también, para cada ruta, la imagen del perfil de la ruta. Junto con la imagen del mapa encuadrado en su entorno y el mapa de carreteras del lugar.

Cursos Alternativos:

a) Ninguna ruta encaja con las restricciones especificadas.

30. El sistema devuelve un mensaje de error.

40. El usuario deberá volver a la página de inicio.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: La información debe ser mostrada al usuario de forma clara.

Nombre: VerMapa.

Resumen: El usuario podrá seleccionar una capa o un conjunto de ellas para vsualizarlas conjuntamente en el visor.

Dependencias: Caso de uso “CrearImagenMapa”.

Actores: Usuario (Iniciador), Mapserver.

Precondiciones: Deben existir los mapas a los que se refieren las rutas previamente.

Postcondiciones: Ninguna.

Curso Normal:

10. El usuario seleccionará la capa o capas que desee visualizar. A continuación deberá pulsar el botón “Refresh”.

20. Caso de uso CrearImagenMapa.

30. Se muestra la imagen del mapa en el visor.

Cursos Alternativos:

Ninguno.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: Ninguno.

Nombre: ZoomIn.

Resumen: El usuario podrá ampliar la resolución de una zona concreta del mapa y centrar en ella su visualización.

Dependencias: Caso de uso "CrearImagenMapa".

Actores: Usuario (Iniciador), Mapserver.

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Curso Normal:

10. El usuario seleccionará la operación adecuada, la correspondiente a "Zoom In".

20. El usuario seleccionará una zona del mapa haciendo click sobre ella.

30. Caso de uso "CrearImagenMapa".

40. Se muestra una nueva imagen centrada en el punto donde se ha hecho click y cuya resolución ha aumentado al doble.

Cursos Alternativos:

Ninguno.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: Ninguno.

Nombre: ZoomOut.

Resumen: El usuario podrá reducir la resolución de una zona concreta del mapa y centrar en ella su visualización.

Dependencias: Caso de uso "CrearImagenMapa".

Actores: Usuario (Iniciador), Mapserver.

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Curso Normal:

10. El usuario seleccionará la operación adecuada, la correspondiente a "Zoom Out".

20. El usuario seleccionará una zona del mapa haciendo click sobre ella.

30. Caso de uso "CrearImagenMapa".

40. Se muestra una nueva imagen centrada en el punto donde se ha hecho click y cuya resolución se ha reducido a la mitad.

Cursos Alternativos:

Ninguno.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: Ninguno.

Nombre: Pan.

Resumen: El usuario podrá desplazar la zona donde se encuentra centrada la mapa.

Dependencias: Caso de uso “CrearImagenMapa”.

Actores: Usuario (Iniciador), Mapserver.

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Curso Normal:

10. El usuario seleccionará la operación adecuada, la correspondiente a “Pan”.

20. El usuario seleccionará una zona del mapa haciendo click sobre ella.

30. Caso de uso “CrearImagenMapa”.

40. Se muestra una nueva imagen centrada en el punto donde se ha hecho click.

Cursos Alternativos:

10. EL usuario hace click en cualquiera de las flechas que rodean al visor.

20. El centro de la imagen del mapa se desplaza en la dirección que indica la flecha que se ha pulsado.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: Ninguno.

Nombre: ActualizaBD.

Resumen: Al crear la ruta se deberán guardar los datos de dicha ruta en la base de datos dependiente de Grass.

Dependencias: Caso de uso "CalculaValores".

Actores: Usuario (Iniciador) y Grass..

Precondiciones: Los datos de la ruta deberán estar calculados previamente.

Postcondiciones: Ninguna.

Curso Normal:

10. El caso de uso comienza tras realizar el caso uso "Calcula Valores".

20. Grass calcula los datos de la ruta realizando un análisis del terreno sobre el que se encuentra la ruta.

30. Grass guarda dichos datos en la base de datos.

Cursos Alternativos:

Ninguno.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: La base de datos empleada será MySQL. A la base de datos se accederá a través del driver dbf de Grass.

Nombre: CalculaValores.

Resumen: Al crear la ruta se deberán calcular los datos de dicha ruta que posteriormente serán almacenados en la base de datos dependiente de Grass.

Dependencias: Caso de uso "Crear_Ruta" y caso de uso "ActualizaBD".

Actores: Usuario (Iniciador) y Grass..

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Curso Normal:

10. El caso de uso comienza tras realizar el caso uso "Crear_Ruta".

20. Grass realiza un análisis del terreno sobre el que se encuentra la ruta para extraer así los datos.

Cursos Alternativos:

Ninguno.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: El sistema deberá trabajar rápidamente y el tiempo de respuesta deberá ser el mínimo posible.

Nombre: CrearVisionMapa.

Resumen: Al crear la ruta se deberá crear una imagen de la ruta encuadrada en el entorno en el que se encuentra la ruta así como el mapa de carreteras de la zona.

Dependencias: Caso de uso "Crear_Ruta".

Actores: Usuario (Iniciador) y Grass..

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Curso Normal:

10. El caso de uso comienza tras realizar el caso uso "Crear_Ruta".

20. Grass crea el mapa de la ruta.

30. Grass realiza un análisis del terreno para calcular los datos de la ruta.

40. Grass crea el mapa del entorno de la ruta.

Cursos Alternativos:

Ninguno.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: El sistema deberá trabajar rápidamente y el tiempo de respuesta deberá ser el mínimo posible.

Nombre: CrearImagenMapa.

Resumen: Mapserver deberá interpretar los mapas y crear la imagen para instanciar posteriormente el template html.

Dependencias: Caso de uso "VerMapa", caso de uso "ZoomIn", caso de uso "ZoomOut" y caso de uso "Pan".

Actores: Usuario (Iniciador) y Mapserver.

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Curso Normal:

10. El caso de uso comienza tras realizar el caso uso "VerMapa" o el caso de uso "ZoomIn" o el caso de uso "ZoomOut" o el caso de uso "Pan".

20. Mapserver interpreta el mapa seleccionado.

30. Mapserver crea la imagen del mapa de acuerdo a los datos interpretados.

40. Mapserver instancia el archivo plantilla html.

Cursos Alternativos:

Ninguno.

Observaciones: Ninguna.

Requisitos no Funcionales Específicos: El sistema deberá trabajar rápidamente y el tiempo de respuesta deberá ser el mínimo posible.

8. Contratos.

Nombre	**Mostrar capa**
Responsabilidades	Muestra una imagen con la capa deseada
Tipo	Sistema
Referencias cruzadas	Caso de uso “VerMapa”. Caso de uso “CrearImagenMapa”.
Notas	
Excepciones	
Salida	Una imagen que contiene la capa o capas seleccionadas
Precondición	
Postcondición	- Se creó una nueva imagen en el servidor. - Dicha imagen es visible en la ventana de la aplicación.

Nombre	**Zoom In**
Responsabilidades	Muestra una imagen con la capa las capas seleccionadas tras haber ampliado el zoom sobre ellas. Dicha ampliación estará centrada sobre el punto que seleccione el usuario. Si este no selecciona ningún punto y hace click sobre “Refresh” el zoom se centrará, valga la redundancia, en el centro de la imagen.
Tipo	Sistema
Referencias cruzadas	Caso de uso “ZoomIn”. Caso de uso “CrearImagenMapa”.
Notas	
Excepciones	
Salida	Una imagen que contiene la capa o capas seleccionadas y sobre la que se ha aplicado un zoom positivo.
Precondición	
Postcondición	- Se creó una nueva imagen en el servidor. - Dicha imagen es visible en la ventana de la aplicación.

Nombre	**Zoom Out**
Responsabilidades	Muestra una imagen con la capa las capas seleccionadas tras haber reducido el zoom sobre ellas. Dicha reducción estará centrada sobre el punto que seleccione el usuario. Si este no selecciona ningún punto y hace click sobre "Refresh" el zoom se centrará, valga la redundancia, en el centro de la imagen.
Tipo	Sistema
Referencias cruzadas	Caso de uso "ZoomOut". Caso de uso "CrearImagenMapa".
Notas	
Excepciones	
Salida	Una imagen que contiene la capa o capas seleccionadas y sobre la que se ha aplicado un zoom negativo
Precondición	
Postcondición	- Se creó una nueva imagen en el servidor. - Dicha imagen es visible en la ventana de la aplicación.

Nombre	**Desplazamiento (Pan)**
Responsabilidades	Muestra una imagen con la capa las capas seleccionadas tras haberse Desplazado en el sentido indicado. La dirección del deslazamiento se puede indicar clickando sobre las flechas de dirección que rodean la imagen o bien clickando directamente sobre la imagen del mapa.
Tipo	Sistema
Referencias cruzadas	Caso de uso "Pan". Caso de uso "CrearImagenMapa".
Notas	
Excepciones	
Salida	Una imagen que contiene el fragmento de la capa o capas seleccionadas. Dicho fragmento corresponde a un desplazamiento sobre la imagen original en la dirección indicada.
Precondición	
Postcondición	- Se creó una nueva imagen en el servidor. - Dicha imagen es visible en la ventana de la aplicación.

Nombre	**Creación de rutas**
Responsabilidades	Crea una nueva ruta, calcula los datos de interés (altura media, distancia, longitud,…) y los almacena en la base de datos. Guarda en la carpeta correspondiente el fichero de coordenadas. Además genera la imagen correspondiente al perfil de la ruta. Por último crea la imagen de la ruta junto al modelo de elevación del terreno y la capa de las carreteras.
Tipo	Sistema
Referencias cruzadas	Caso de uso “calculaValores”. Caso de Uso “actualizaBD”. Caso de Uso “CrearVisionMapa”. Caso de Uso “CrearImagenPerfil”.
Notas	
Excepciones	
Salida	
Precondición	- El mapa de rutas debe existir previamente.
Postcondición	- El mapa de rutas queda actualizado quedando añadida la nueva ruta. - La imagen del perfil de la ruta queda almacenada en el sistema. - Se crea un archivo con las coordenadas de la ruta. - Se crea una nueva imagen del mapa de la ruta junto al modelo de elevación y la capa de carreteras.

Nombre	**Consulta de ruta**
Responsabilidades	Consulta los datos almacenados sobre la ruta seleccionada.
Tipo	Sistema
Referencias cruzadas	Caso de uso “consultarRuta”.
Notas	Solo se puede seleccionar una ruta a la vez
Excepciones	
Salida	Fila de la tabla correspondiente a la ruta seleccionada
Precondición	- El mapa de rutas debe estar seleccionado y visible previamente.
Postcondición	- La información asociada a la ruta así como el perfil de dicha ruta son visibles en la web.

Nombre	**Búsqueda de ruta**
Responsabilidades	Selecciona las rutas que cumplan con las restricciones impuestas por el usuario.
Tipo	Sistema
Referencias cruzadas	Caso de uso “ConsultaRestringida”
Notas	
Excepciones	
Salida	Datos de las rutas que cumplen con las restricciones impuestas.
Precondición	
Postcondición	- La información asociada a las rutas seleccionadas así como los perfiles asociados se mostrarán en una página con posibilidad de impresión.

Nombre	**Selección de rutas**
Responsabilidades	Realiza las consultas pertinentes sobre las rutas seleccionadas por el usuario.
Tipo	Sistema
Referencias cruzadas	Caso de uso “seleccionarRuta”.
Notas	
Excepciones	
Salida	Datos de las rutas que han sido seleccionadas por el usuario.
Precondición	- El mapa de rutas debe estar seleccionado y visible previamente.
Postcondición	- La información asociada a las rutas seleccionadas así como los perfiles asociados se mostrarán en una página con posibilidad de impresión.

9. Descripción de las operaciones.

9.1. Visualización de mapas.

Esta operación es quizás la más básica en cuánto al funcionamiento de cuántas se ofrecen en la herramienta.

En esta función se persigue que el usuario sea capaz de visualizar los distintos mapas presentes en la base de datos de la que disponemos. Para que sean visibles no basta con que estén almacenados en la carpeta de mapas. Además es necesario que la capa correspondiente al mapa esté presente en el Mapfile.

Antes de pasar a la descripción del funcionamiento hay que decir que en esta operación no solo implica la visualización de las capas seleccionadas. Además se podrá realizar "zoom in", "zoom out" y desplazar la imagen (pan), posibilidades estas que serán comentadas más adelante.

La visualización se sustenta sobre el archivo Mapfile. Dicho archivo es el que consulta Mapserver para así llevar a cabo la correcta instanciación del template que compone la web. Para ello se compone de una serie de capas definidas al efecto. Estas capas corresponderán a los mapas que se podrán visualizar. Están compuestas de la ruta hacia al mapa así como diversos parámetros de configuración que determinarán en última instancia la forma en la que el usuario verá el mapa en cuestión.

Las capas que presenta el archivo Mapfile para esta configuración en este sistema determinado son estáticas, es decir, el número de capas y por tanto el conjunto de mapas que se visualizarán es fijo. En algún momento de la vida del programa el administrador puede desear modificar el conjunto de mapas o bien añadir nuevas capas o incluso eliminar alguna existente. Para lo primero deberá, en primer lugar cambiar el conjunto de mapas en la base de datos si lo que pretende es la modificación completa del conjunto de mapas. El siguiente paso a seguir será la modificación de las capas existentes en Mapfile para, primero, señalar las nuevas rutas de los mapas y segundo modificar los parámetros de configuración de las capas si así lo desea.

Por otro lado, si el administrador desea modificar el número de capas (aumentarlo o reducirlo) en primer lugar deberá crear o, en su caso, eliminar la capa concreta del archivo Mapfile y en segundo lugar deberá efectuar los cambios oportunos en la interfaz web a fin de, si lo que se va a hacer es eliminar una capa, quitar la posibilidad de que el usuario seleccione dicha capa si dicha visualización depende del criterio del usuario, o bien eliminar el elemento

de la página que contenía dicha capa. Si por el contrario nuestra intención es crear una capa, la forma de proceder, evidentemente, será la contraria. Esto es, crear en primer lugar la capa correspondiente en el archivo Mapfile, y en segundo lugar crear la posibilidad de que el usuario seleccione dicha capa para su visualización si ésta depende de él o, en otro caso, crear el contenedor adecuado para albergar la imagen de dicha capa en la web.

Al acceder a la web el usuario verá el visor y, por defecto, el mapa de elevación del terreno. Esta capa es la denominada "capa por defecto" y será visible al acceder a la web y además permanecerá, a modo de referencia del terreno, visible al visualizar los mapas vectoriales (formato shape).

El mencionado mapa por defecto podrá cambiarse por el que desee el administrador del sistema sin más que modificar el archivo Mapfile y el valor "por defecto" (DEFAULT) a la capa correspondiente al mapa deseado. Por supuesto esta propiedad debe ser eliminada de la capa correspondiente al mapa que anteriormente se consideraba "por defecto".

Hay que decir que en Mapserver es posible visualizar más de una capa a la vez de modo que los sucesivos mapas si superponen el uno sobre el otro a modo de transparencias. Esta posibilidad ha sido aprovechada en nuestro sistema permitiendo al usuario, por ejemplo, consultar el mapa de carreteras a la vez que el mapa de rutas con el fin de poder visualizar de forma fácil la manera de llegar a la ruta en cuestión.

Los mapas con formato "shape" (MAPAS CON FORMATO SHAPE) siempre se visualizan sobre el mapa raster subyacente que en ese momento se encuentre como capa activa o, si ninguna se encuentra activa, sobre el mapa por defecto.

Junto con los mapas se visualizan también la leyenda de los mismos así como una escala con objeto de facilitar la comprensión. Por supuesto, como se explicó en el apartado correspondiente de la sección de diseño, para la visualización de los mencionados objetos también hay que crear la capa correspondiente en el Mapfile.

En este apartado además cabe comentar las operaciones "Zoom In", "Zoom Out" así como el desplazamiento sobre los mapas (Pan), ya que, si bien se trata de operaciones separadas, éstas están estrechamente relacionadas con la visualización de mapas, hasta el punto de que no se entienden sin ella. Es por este motivo por el que pasamos a comentarla a continuación.

Para hacer Zoom In sobre una zona del mapa, en primer lugar hay que marcar la opción correspondiente en el menú de la izquierda (ver manual de uso).

A continuación, se deberá pulsar sobre la zona del mapa deseada.

De esta forma al enviar el formulario al servidor Mapserver éste interpretará la operación a realizar al comprobar el valor de la etiqueta correspondiente, la cual indica la operación a realizar, así como la zona del mapa sobre la que se desea hacer zoom. Para ello Mapserver hace automáticamente la georreferenciación de las coordenadas pulsadas sobre el mapa.

El valor del aumento al aplicar zoom está preestablecido a un factor de 2 unidades. Para modificar este valor hay que cambiar el valor del campo "hidden" correspondiente que existe en el template html.

La operación Zoom Out es idéntica a la anteriormente comentada, sin embargo, el efecto producido es exactamente el contrario. En este caso, evidentemente, la opción que se deberá marcar del menú de la izquierda es la correspondiente a "Zoom Out". El valor del factor de reducción es el mismo que el de ampliación (valor 2) leyéndose este del mismo campo "hidden" siendo Mapserver el encargado de aplicarlo de la forma conveniente (a valor -2).

Como última función relacionada con este apartado está la de desplazamiento sobre el mapa. Esta es la función seleccionada por defecto así que se podrá llevar a cabo directamente con solo pulsar sobre el mapa. Esta acción desplazará el mapa en la dirección marcada por la línea imaginaria que une el centro de la imagen con el lugar en el que se ha pulsado el ratón. Otra posibilidad de llevar a cabo esta tarea será pulsando sobre las flechas de dirección que rodean al visor del mapa.

En este momento es oportuno hablar del mapa de referencia.

La mencionada imagen sirve para comprobar desde otra perspectiva la porción del mapa sobre la que se está trabajando. Si se hace click sobre esta imagen se puede comprobar que el efecto es el mismo que al pulsar sobre la misma región en el mapa principal.

Si en algún momento se desea volver a la situación de inicio habrá que pulsar el botón "Full Map".

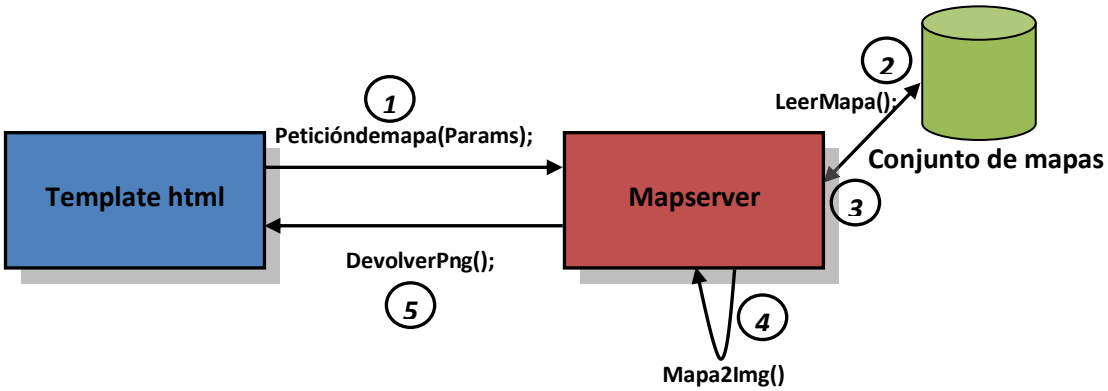
En definitiva, el fundamento que siguen este tipo de operaciones es el siguiente:

1. Desde el cliente se envía la petición al servidor.

2. El servidor ejecuta Mapserver con los parámetros recibidos del cliente.

3. Mapserver interpreta el mapa y los parámetros recibidos y crea una imagen ".png" que almacena en el servidor y devuelve al cliente la ruta de la imagen creada.

4. Así el cliente es instanciado permitiendo la visualización de la imagen deseada.

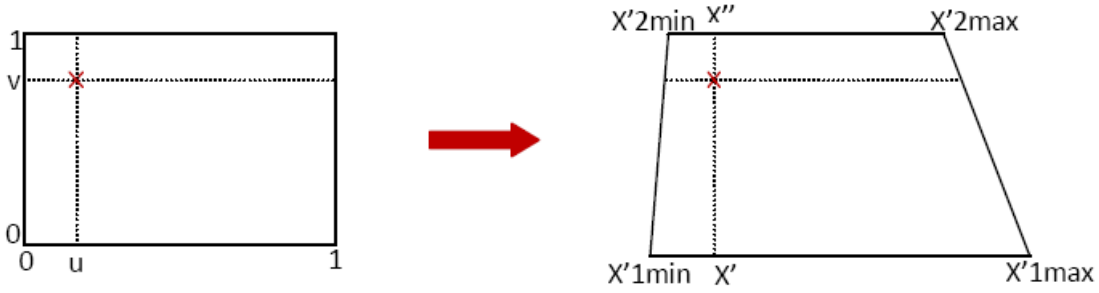


9.2. Creación de rutas.

Esta operación permitirá al usuario marcar sus propias rutas sobre el terreno. El cliente deberá marcar la opción de la mencionada operación y a continuación deberá marcar sobre el mapa las coordenadas de la ruta deseada. Dicho camino se irá formando uniendo los sucesivos puntos que el usuario irá marcando como se ha mencionado ya.

Como paso previo a la introducción de los puntos que marcarán la ruta el usuario deberá escribir el nombre que llevará esta ruta. Si el cliente no escribe nada el nombre guardado por defecto será "Ruta sin nombre". Si bien hay que decir que antes de enviar la petición de la creación de la ruta al servidor hay que pulsar el botón "Submit" del PopUp habilitado por el sistema para que el usuario introduzca el mencionado nombre de la ruta.

Una vez marcada la ruta deseada y justo antes de almacenarla habrá que georreferenciarla, esto es, pasar las coordenadas de la ruta dibujada (relativas a la página web) a coordenadas terrestres. Para ello se empleará el procedimiento que puede verse en la figura:



Es decir la correspondencia no puede ser automática puesto que en realidad la imagen del mapa sería un trapezoide, no un rectángulo. X'2min, X'2max, X'1min, X'1max son datos conocidos de antemano que corresponden a las esquinas del mapa. Para obtener los puntos x' y x'' se procede como sigue:

$$X' = u * (X'1max - X'1min) + X'1min;$$

$$X'' = u * (X'2max - X'2min) + X'2min;$$

Para la Y se procede análogamente.

El valor u se obtiene de la siguiente forma:

$$u = (X – Xmin)/(Xmax – Xmin);$$

Para obtener el valor de v se procede de la misma forma pero usando los valores del eje Y.

Los valores Xmax y Xmin son conocidos y corresponden a las coordenadas del mapa en la web. Los valores X e Y se corresponden con las coordenadas del click del ratón del usuario.

Finalmente obtendremos 4 valores: x', x'', y', y''. Para obtener los valores finales XY georreferenciados lo que hacemos es crear una recta entre x' y x'' e y' e y'' e intersecarlas. El valor de la intersección será el valor de las coordenadas buscadas.

A continuación se enviará la petición al servidor, pero en este caso no a Mapserver. Para esta operación se hará uso del script "controller.php", el cual mediante llamadas al sistema ejecutará los módulos apropiados de Grass para la creación del mapa.

Una vez creado el mapa y, de forma totalmente transparente para el usuario, el controlador ejecuta diferentes módulos de Grass los cuales tienen como objetivo el análisis del mapa de ruta creado con objeto de poder extraer los parámetros deseados que serán posteriormente almacenados en la base de datos. Dichos datos serán: altitud media, pendiente máxima del terreno y longitud de la ruta. Además se creará, de la misma forma comentada, el perfil del terreno que atraviesa la ruta.

Será también Grass, por supuesto mediante llamadas al sistema y también de forma transparente en la misma operación, la que almacene todos estos parámetros en la base de datos, los cuales estarán indexados por el número de categoría de la ruta creada, el cual será único para cada una de ellas.

Hay que decir que la imagen del perfil se almacenará en la carpeta "profile" del servidor con objeto de poder realizar posteriores consultas de forma rápida pues no será necesario volver a crear esta imagen.

Además las coordenadas de la ruta se guardarán en la carpeta "coordinates" pues pueden ser interesantes para posteriores operaciones futuras.

Por último en la carpeta images_Rutes se almacenará una imagen que contendrá el modelo de elevación del terreno, sobre este el mapa de carreteras así como la ruta marcada. El objetivo de la realización de esta imagen es análogo al del perfil, de forma que solo será creada una vez, permitiendo que las operaciones que requieran de esta imagen se ejecuten más rápidamente.

Tanto las dos imágenes mencionadas anteriormente como el archivo de coordenadas tendrán por nombre la categoría de la ruta seguida de la extensión correspondiente.

Para cada ruta el valor de la categoría se extrae de la siguiente manera; se consulta en la base de datos el valor de las categorías de los mapas hasta ahora almacenados y se incrementará en una unidad el valor del mayor resultado obtenido. Este será el nuevo valor de la categoría y su extracción será (como viene siendo habitual) transparente al usuario.

Por último hay que señalar que, para evitar interferencias en el trabajo de los diferentes clientes, para esta operación es necesario organizar el acceso a los recursos mediante un mecanismo de exclusión mutua.

Como se ha comentado en apartados anteriores, El mecanismo de exclusión mutua implementado se basa en los autovalores de MySQL de forma que cuando se solicita la creación de una nueva ruta se comprueba el valor máximo de categoría existente en dicha base de datos e incrementa dicho valor en una nueva unidad. Este valor será usado como valor de categoría para la nueva ruta. Todos los mapas auxiliares se crearán nombrados con este valor.

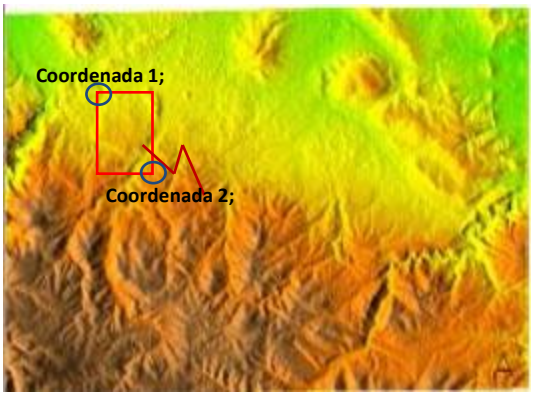
Estos autovalores se modifican de forma atómica, lo que garantiza la exclusión mutua entre los procesos. Este hecho además provoca la no interferencia en el proceso de creación de mapas al llamarse todos de forma diferente (el nombre será el valor de la mencionada variable).

9.3. Selección de mapas.

Esta función permite que el usuario pueda seleccionar (incluyendo en un cuadrilátero) las rutas de las que desea información. Como es fácil deducir se podrán seleccionar, en este caso, más de una ruta a la vez. No ocurrirá lo mismo con la consulta sobre los mapas que se expondrá en el siguiente apartado.

En este caso, al igual que ocurría con la operación de creación de rutas, no se hará uso del cgi Mapserver. Ahora, como entonces, el módulo donde se redirigirá el formulario de la página html será a "controller.php".

Para resolver este problema el procedimiento seguido es el siguiente: en primer lugar el usuario deberá marcar en el visor de los mapas 2 puntos que corresponderán a las coordenadas de las esquinas superior izquierda e inferior derecha respectivamente de un cuadrilátero de forma que el resultado es el mostrado en la figura siguiente.



Como se puede observar en el esquema no es necesario que la ruta o rutas queden encuadradas completamente dentro del cuadrilátero, bastando un simple solapamiento entre el mismo y la ruta para que ésta sea considerada para su selección.

Una vez marcado el ámbito de selección de rutas todas las operaciones que se realizan a continuación lo hacen de forma transparente para el usuario.

Tras marcar el usuario el segundo punto que definirá el mencionado cuadrilátero, este es georreferenciado usando el mismo procedimiento que el descrito en el apartado 7.3. Estas coordenadas junto con la operación seleccionada por el usuario se envían de forma automática al servidor. A continuación el módulo controlador analizará la operación recibida y procederá como sigue:

1. En primer lugar crea un nuevo mapa vectorial consistente en el cuadrilátero seleccionado.

2. Tras esto se realiza la operación de Grass “v.patch” con el mapa de rutas y el mapa creado anteriormente. Esta operación devolverá las rutas que se cruzan o están incluidas en el mapa creado en primer lugar (el cuadrilátero).

3. Grass extraerá los números de categoría de los mapas seleccionados y realizará las consultas oportunas a la base de datos para así extraer los datos almacenados sobre dichas rutas, las cuales, como es de esperar, están indexadas en la base de datos por el mencionado número de categoría.

4. Tras obtener toda la información deseada creará una ventana emergente la cual instanciará llamando a la función “CreateRutesView(params)” que recibirá como parámetro el resultado de la consulta. Esta función está incluida en el módulo rutes.php y lo que hará será compone una página web a través de comandos “echo”. Esta página web tendrá la forma y la función que se emplica a continuación.

La página tendrá el fin último de que el usuario pueda consultar de manera cómoda y completa todos los parámetros de interés de las rutas seleccionadas. Además esta página se podrá imprimir.

La información que aparecerá para cada ruta será:

1. Un mapa de elevación del terreno junto con la imagen de la ruta y el mapa de carreteras superpuesto para poder elegir una ruta adecuada para llegar al lugar. Este mapa se crear en el momento de guardar la ruta en la operación “Crear rutas” comentada en el apartado 7.3 por lo que solo habrá que consultarla ya que está indexada por su número de categoría.

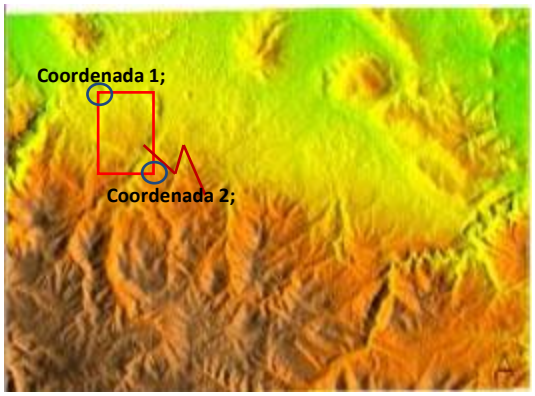
2. Una imagen del perfil del terreno por donde pasa la ruta. En él se esquematiza la variación de altura que sufre la ruta conforme aumenta la distancia recorrida. Esta imagen también se crea en el momento de guardar la ruta la ruta y, como no puede ser de otra forma, también se encuentra indexada por el número de categoría del mapa.

3. Toda la información asociada al mapa, esto es, nombre de la ruta, altitud media que presenta, máxima pendiente que nos podemos encontrar en el recorrido y longitud de la ruta.

9.4. Consulta de ruta.

El fundamento de esta operación es la consulta de una ruta en concreto. Se puede considerar el paso previo a la operación comentada en el apartado anterior de forma que el cliente podrá consultar de manera rápida las rutas existentes en el servidor para posteriormente seleccionar las que más le interesen evitando así que se estén continuamente abriendo nuevas ventanas con los datos de rutas que no serán de su interés.

El procedimiento seguido en este caso es análogo al tratado en el apartado anterior debiendo en este caso seleccionar la operación correspondiente.



La restricción impuesta en este caso será que solo se podrá seleccionar una ruta en cada caso produciéndose un mensaje de error en otro caso.

En esta operación tras seleccionar la ruta el servidor creará de nuevo un mapa consistente en el cuadrilátero creado y posteriormente seleccionará la ruta mediante v.patch. Es en este punto cuando el servidor comprueba que no se ha seleccionado más que una ruta.

Una vez constatado este hecho se procederá a realizar la consulta en base al número de categoría de la ruta seleccionada. Una vez obtenidos todos los datos se volverá a instanciar el cliente con la capa de las rutas activa para permitir nuevas operaciones con ellas sin obligar al usuario a volver a seleccionarla. Al mismo tiempo en la parte derecha de la pantalla aparecerá toda la información de interés asociada a la ruta seleccionada.

En este punto hay que decir que Mapserver ya proporciona funcionalidad para realizar consultas sobre los mapas. Para ello, habría que cambiar el valor de la etiqueta “mode” del cgi pasando esta de “browse” a “query”. Tras esto deberíamos disponer en la web de las instancias necesarias sobre las que trabajaría Mapserver.

La decisión de usar también Grass para la realización de las consultas en lugar de Mapserver se justifica gracias a la enorme flexibilidad que presentan este tipo de consultas frente a las de MapServer ya que al usar el primero una base de datos en lugar de la información contenida en el propio mapa, hace que el abanico de datos que asociar a las rutas sea mucho mayor. Además hecho de tener la base de datos separada del mapa permite su consulta en cualquier momento no requiriendo su solicitud por parte del cliente si no que en cambio se puede realizar la consulta en cualquier momento que lo requiera la operación que se está llevando a cabo de forma totalmente transparente al usuario.

9.5. Búsqueda restringida.

Con este apartado terminan las operaciones que se le ofrecen al usuario del sistema. Se trata nuevamente de una operación de consulta sobre los mapas.

En esta ocasión la selección de los mapas a consultar no se realizará en base a ningún ámbito físico definido por el usuario, entendiéndose por ámbito físico a la selección realizada en los dos apartados anteriores consistente en la definición de un cuadrilátero por parte del usuario el cual enmarcaba a las rutas escogidas. Lo que deberá hacer el usuario en lugar de ellos será definir una serie de variables sobre las que se realizará la consulta en la base de datos y que se definirán posteriormente. Naturalmente las rutas seleccionadas serán aquellas que cumplan con las restricciones definidas por el usuario.

El proceso de búsqueda será el siguiente:

1. En primer lugar el usuario marcará la opción correspondiente a la mencionada operación, hecho que provocará que se le presenten al usuario una serie de opciones que deberá seleccionar. Estas opciones serán:

    a. Nombre de la ruta.

    b. Valores máximo y mínimo de la pendiente de la ruta.

    c. Valores máximo y mínimo de la altitud media deseada para la ruta.

    d. Valores máximo y mínimo de la longitud deseada para la ruta.

    Estos parámetros tienen un orden de importancia el cual se explicará posteriormente, y en base al cual se definirá la consulta. Además hay que decir que no es obligatorio rellenar todos los campos para la búsqueda.

2. Después de definir los parámetros estos se enviarán al servidor el cual los procesará y ejecutará la consulta correspondiente sobre la base de datos.

3. Una vez devueltos los resultados correspondientes a la búsqueda se volverá a llamar a la función “CreateRutesView(params)” mencionada en el apartado 7.3 para volver a componer la página lista para la impresión que albergará aquellas rutas que cumplan con los parámetros definidos por el usuario.

4. Si por el contrario no se encontrara ninguna ruta que cumpliera con las especificaciones requeridas en la búsqueda se informará al usuario del correspondiente incidente.

Anteriormente se ha expuesto que los parámetros de búsqueda siguen un cierto orden de importancia. Esto quiere decir que la especificación de unos tiene más peso en la búsqueda que otros. En este sentido hay que decir que si el usuario especificara un nombre de ruta la búsqueda se haría exclusivamente en base a este parámetro no tomándose en cuenta los demás. Esto es así porque si el usuario conoce el nombre de la ruta y así lo especifica se asume que quiere los parámetros asociados a ésta ruta en concreto y no a otras por lo que la especificación de los demás parámetros carece de importancia, es más, en caso de que el usuario introdujese mal estos valores podría llevar a que la búsqueda no produjese ningún resultado.

Si no se introdujese ningún nombre de ruta entonces se tendrán en cuenta el resto de las opciones. En este caso ninguna tendrá más peso que otra, tomándose en cuenta todos aquellos valores especificados por el usuario. Estos parámetros son, como ya se ha mencionado, valor máximo y mínimo de la altitud media de la ruta, máximo y mínimo de la máxima pendiente de la ruta y máximo y mínimo de la longitud de la ruta.

El hecho de que se dé la posibilidad de introducir valores máximos y mínimos es para definir así un rango de posibles valores ya que el tener que introducir valores únicos y exactos haría esta operación inviable debido a la altísima variabilidad que pueden presentar estos parámetros. Aun así no es obligatorio tener que definir ambos extremos. Si solo se da valor a uno de ellos significará que el intervalo queda abierto por la parte no definida.

Es fácil notar que mientras más de estos parámetros queden definidos más restringida será la búsqueda.

De lo expuesto en el punto 3 se puede deducir que el objetivo perseguido por esta operación es exactamente el mismo que el de la operación definida en el apartado 7.3, diferenciándose ambas operaciones únicamente en la forma de seleccionar las rutas que se visualizarán.

10. Manual de uso.

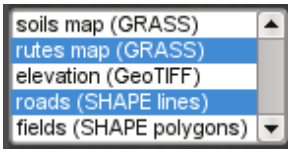
A continuación se expone una pequeña guía de uso de la aplicación con objeto de que el usuario pueda empezar a manejar la herramienta sin ningún tipo de problemas.

En primer lugar el usuario deberá acceder a la aplicación a través de la URL http://<Dirección_IP>/Proyecto. Habrá que sustituir <Drección_IP> por la Ip pública del servidor donde se encuentra instalada la aplicación.

Es entonces cuando el usuario verá la pantalla de inicio. Dicha pantalla ya estará instanciada puesto que como se mencionó en apartados anteriores en el archivo index.html se redirecciona enviando una petición al servidor para instanciar el archivo de plantilla.

Sin más empezaremos a detallar las operaciones que el usuario podrá llevar a cabo con la aplicación.

La más sencilla es la visualización de mapas. Para ello el usuario deberá seleccionar la capa deseada para dicha visualización. Es posible ver más de una capa a la vez. Para ello habrá que seleccionar las distintas capas manteniendo la tecla “ctrl” pulsada.

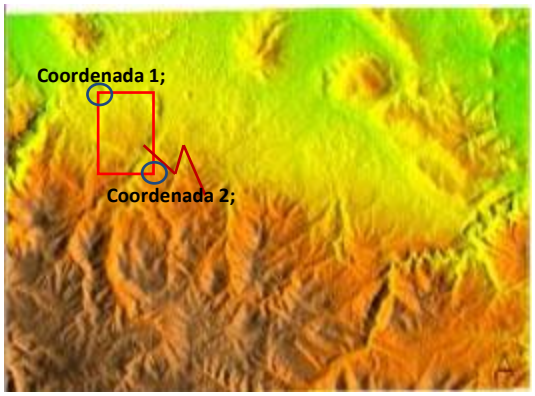


Por último se deberá pulsar el botón “Refresh” lo que provocará el envío de la petición al servidor Mapserver el cual instanciará de nuevo el archivo de plantilla.

Si el usuario desea tener una visión un poco más centrada de una zona del mapa o simplemente cambiar la zona de acción. Ello es posible realizarlo de tres formas diferentes. La primera de ellas es pulsando sobre las flechas de dirección que rodean al visor. Otra posibilidad es, manteniendo seleccionada la opción de “Pan” pulsar sobre la zona del mapa deseada. Esta acción provocará que el mapa se desplace en la dirección del vector que marcaría la línea imaginaria entre el centro del mapa y el punto donde el usuario ha hecho click. La última posibilidad es haciendo click sobre el mapa de referencia. El efecto producido es mismo que para la última opción comentada.

Además es posible aumentar o reducir la resolución del mapa centrada en una zona concreta. Para llevar a cabo esta posibilidad es necesario seleccionar las operaciones correspondientes “ZoomIn” o “ZoomOut” dependiendo del efecto deseado. Una vez hecho esto se deberá hacer click con el ratón sobre la zona del mapa deseada. Si bien es cierto que la forma de proceder en la operación de Pan y de Zoom In o Zoom Out es la misma, el servidor diferenciará entre ambas por la operación seleccionada en el momento de enviar la petición a éste.

Otra de las operaciones permitidas es la de consulta de los datos sobre el mapa. Para ello la primera acción a realizar es visualizar las rutas existentes en el servidor. Una vez hecho esto se seleccionará la operación “Query rutes”. Ello permitirá al usuario poder hacer click sobre el mapa sin que esta acción provoque el envío de una petición al servidor. En este caso el usuario deberá marcar dos puntos sobre el mapa que señalan la diagonal de un cuadrilátero dentro del que se enmarca la ruta a consultar. Solo se podrá seleccionar una ruta cada vez, mostrando el servidor un mensaje de error en otro caso.



Una vez marcado el segundo punto se enviará una nueva petición al servidor que provocará que este devuelva la consulta con los datos correspondientes a la ruta seleccionada.

Si el usuario desea ver la gráfica del perfil de la ruta de forma un poco más clara solo deberá pasar el ratón por encima de la imagen y esto provocará un efecto lupa sobre esta imagen.

Otra operación será la selección de mapas. El modo de proceder será idéntico a la operación anterior salvo que en este caso podrán encuadrarse en la selección más de una ruta a la vez y la operación seleccionada será “Select rutes”. En este caso el efecto no será la devolución de los datos de la ruta a la plantilla inicial. En este caso se abrirá una nueva ventana que mostrará los datos de cada ruta seleccionada. Además se proporcionará la imagen del perfil de cada ruta así como la imagen de la cada ruta encuadrada en el entorno en el que se encuentra junto con el mapa de carreteras de la zona con objeto de que el usuario pueda averiguar cómo llegar a la ruta en cuestión.

Para la realización de estas dos operaciones previamente es necesario que la capa de las rutas esté seleccionada.

La última de las operaciones de consulta permitida será la de búsqueda restringida de mapas. Se deberá seleccionar la operación “Restricted query rutes”. Al seleccionar la operación se abrirá una ventana emergente que permitirá al usuario seleccionar los parámetros en los que se basará la búsqueda.

En este momento hay que decir que si el usuario indica un nombre de ruta no se tendrán en cuenta el resto de parámetros. En caso contrario se tendrán en cuenta el resto de parámetros que el usuario indique. El resultado de esta operación será el mismo que el de la operación anterior.

La última de las operaciones permitidas será la de creación de rutas. Para ello el usuario deberá seleccionar la operación “Draw rutes”. Ello provocará que se pueda dibujar sobre el visor del mapa. La forma de dibujar será ir marcando sucesivos puntos de la ruta y el sistema irá uniendo dichos puntos generando así la ruta. Una vez se ha acabado de dibujar la ruta se deberá pulsar el botón “Save map” lo cual provocará que se envíe una petición al servidor el cual se encargará de generar la ruta marcada por el usuario.

11. Conclusiones.

La gran cantidad de aplicaciones existentes hoy en día y basadas en el uso de sistemas SIG pone de manifiesto el auge de estos sistemas en la actualidad y por lo tanto se muestra como un campo de investigación muy abierto y actual prestándose a continuos cambios y aportaciones.

Se ha comentado con anterioridad que una posible medida de la bondad de un sistema SIG es la calidad y vigencia de los datos sobre los que opera. Si bien esta definición se puede considerar suficientemente objetiva y aproximada, hemos de decir que no es del todo completa en cuanto que no tiene en cuenta qué operaciones nos permite realizar para tratar esos datos. Si bien es cierto que la mayoría de los sistemas SIG clientes que se instalan en el ordenador de usuario proveen de todas las operaciones necesarias es por esta razón por la que no se tiene en cuenta a la hora de evaluar un sistema SIG. Esta afirmación, sin embargo, no es del todo cierta cuando nos referimos a sistemas SIG implantados siguiendo una arquitectura cliente-servidor.

En estos últimos casos es habitual que estos sistemas se reduzcan a meros visores de información geográfica y los más completos además permiten la consulta interactiva de datos sobre dichos mapas.

En el sistema implementado se propone dar un paso más allá en la implementación de este tipo de aplicaciones cliente-servidor. A parte de de la funcionalidad básica mencionada (visualización y consultas sobre los mapas) además se provee al usuario de la capacidad de crear mapas contra el servidor. Esta novedad puede aportar mucho al campo de los sistemas SIG cliente-servidor en cuanto al hecho de que supone un paso más a la hora de reducir la distancia a la que se encuentra este tipo de sistemas de las herramientas cliente comentadas en primer lugar.

Si bien es cierto que muchas arquitecturas cliente-servidor sigue una filosofía parecida a la planteada en esta ocasión combinando las herramientas servidoras como Mapserver con herramientas SIG como Grass, esta interacción se refiere más bien a la posibilidad de que el servidor pueda interpretar correctamente el formato de los mapas con los que trabaja el programa cliente. Esta interacción será también indispensable en el sistema objeto de estudio por cuanto que los mapas serán creados por la herramienta cliente por lo tanto será necesario poder visualizarlos posteriormente.

El hecho de que esta herramienta sea también de código libre hace que su evolución se prevea también como continua. Es por ello por lo que nunca se ha considerado como una

aplicación final. Más bien se concibió con la idea de proporcionar una base de trabajo que sigue en la línea de la tendencia actual.

Además el hecho de que para las consultas se emplee el SIG cliente dotará de mayor grado de flexibilidad a la aplicación. Es decir al no estar limitada la base de datos por los datos almacenados en el mapa si no que, por el contrario, se permite realizar análisis del terreno por lo que las posibilidades de consulta se encuentran limitadas, por tanto, por las posibilidades de análisis que ofrecerá el sistema en cuestión.

Es por estas razones por las que además de la vigencia y la calidad de los datos sobre los que se trabaja también debe tenerse en cuenta las posibilidades que el sistema ofrece al usuario para trabajar con los mapas. Es en este sentido donde la herramienta propuesta podría obtener una evaluación más positiva puesto que no solo ofrece nuevas posibilidades con respecto a los sistemas actuales si no que, como se ha comentado, provee de la base para el posterior desarrollo de nuevas aplicaciones.

12. Bibliografía.

- [1]. Kropla Bill (2005) Beginning Mapserver. Open Source GIS Development.
- [2]. http://mapserver.org/documentation.html . [Documentación de la página oficial de Mapserver].
- [3]. http://old-mapserver.gis.umn.edu/doc40/cgi-reference_es.html.
- [4]. http://grass.itc.it/gdp/tutorials.php.
- [5]. M. Neteler, H. Mitasova, 2007. Open Source GIS: A GRASS GIS Approach. Third edition.
- [6]. http://grass.itc.it/grass64/manuals/html64_user/index.html.
- [7]. http://grass.osgeo.org/grass62/manuals/html62_user/grass-dbf.html.
- [8]. http://wiki.debian.org/es/Grass.
- [9]. http://es.wikipedia.org/wiki/ Sistema_de_Información_Geográfica.
- [10]. http://www.monografias.com/trabajos14/informageogra/informageogra.shtml.
- [11]. http://grass.osgeo.org/grass64/manuals.
- [12]. http://us3.php.net/.
- [13]. http://www.joseane.com/recursos/overlib.htm.
- [14]. http://www.walterzorn.com/jsgraphics/jsgraphics_e.htm.