VNIVERSIDAD NACIONAL
AVTÓNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ARAGÓN

SISTEMA DE INFORMACIÓN GEOGRAFICO PETROLERO

T E S I S

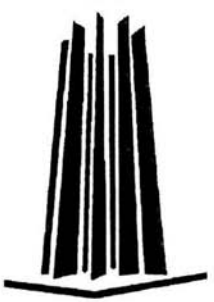
QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

LUIS EDUARDO PALOMINO MORENO

DIRECTOR DE TESIS:
ING. CESAR FRANCISCO GERMAN ROSAS



MÉXICO

2004

TITULO SISTEMA DE INFORMACION GEOGRAFICO PETROLERO

AGRADECIMIENTOS

INTRODUCCION

CAPITULO I CONCEPTO Y EVOLUCIÓN DE LOS SISTEMAS GEOGRAFICOS.

1.1 Concepto de Sistemas de Información Geográfica (SIG)
1.2 Historia de los (SIG).
1.3 Tipos de Sistemas de Información Geográfica.
1.4 Utilidad de los Sistemas de Información Geográfica

CAPITULO II SISTEMAS DE INFORMACION GEOGRAFICA EN PEMEX

2.1 Introducción.

2.2 SICORI (Sistema Corporativo de Información Geográfica)

2.3 Creación y Evolución (SICORI).

2.4 Servicios del SICORI y sus productos.

2.5 Proyectos y servicios desarrollados

2.6 Conclusión.

CAPITULO III NECESIDADES DE UN SIG CORPORATIVO PETROLERO

3.1 Identificación de necesidades
3.2 Alternativas
3.3 Estudio de Viabilidad
3.4 Funciones del sistema

CAPITULO IV ESTRUCTURA DEL SISTEMA (SIGP)

4.1 Estructura de las tablas
4.2 Caso de Uso
4.2 Diagrama de clases en la capa del cliente.
4.3 Diagramas de clases en el servidor

CAPITULO V IMPLEMENTACIÓN DEL SISTEMA

5.1 Implementación de Herramientas Lógica de Negocio
5.2 Módulos de privilegios
5.3 Módulos de Visualización.
5.4 Modulo de Herramientas.
5.5 Configuraciones iniciales del sistema.

CONCLUCIÓN FINAL

A mis Padres:

En la vida se nos da pocas oportunidades para salir adelante y contar con unos seres que nos induzcan y enseñen que no debemos darnos por vencidos para lograr nuestras metas e ideales.
Dios me ha dado la suerte de tenerlos y la oportunidad de contar con ustedes y compartir mis fracasos, triunfos y tristezas y alegrías.
Infinitamente les agradezco todo el apoyo que me han brindado para subir este escalón que será el inicio de mi profesión.

Introducción.

Hoy en día y desde diversas organizaciones se invierten grandes sumas de dinero en el desarrollo de bases de datos georeferenciadas y en Sistemas de Información Geográfica (SIG). Es previsible además que durante los próximos años se inviertan miles de millones más. Todo ello está sucediendo en un corto período de tiempo, ya que hace pocos años el SIG era una herramienta muy especializada sólo al alcance de pocas organizaciones y una curiosidad para el público en general. Se pueden dar dos explicaciones a estos fenómenos:

- La primera reside en el abaratamiento de los costos de los equipos informáticos, que cada día los hace más accesibles, para un mayor número de usuarios.

- La segunda y de mayor importancia es que la geografía (y los datos que sirven para cuantificarla) forma ya parte de nuestro mundo cotidiano; la mayoría de las decisiones que tomamos diariamente están en relación con o influenciadas por un hecho geográfico. Los camiones de bomberos, por ejemplo, se envían a su destino a través de la ruta más corta posible, las aportaciones económicas de los gobiernos a las comunidades se basan frecuentemente en la distribución geográfica de la población, o las enfermedades se estudian gracias a la identificación de las áreas en donde se producen y de la velocidad a la que se expanden.

CAPÍTULO I

CONCEPTO Y EVOLUCIÓN DE LOS SISTEMAS GEOGRAFICOS.

1.1 Concepto de Sistemas de Información Geográfica (SIG)

El término SIG se establece de la palabra en inglés Geographic Information System (SIG).

Existen varias definiciones para explicar lo que es un SIG una definición precisa y concreta podría ser:
- Un conjunto de equipos informáticos, de programas de datos geográficos y técnicos organizados para recoger, almacenar, actualizar, manipular, analizar y presentar eficientemente todas las formas de información georeferenciada así como presentar información alfanumérica o descriptivos que se asocien a esos mapas para formar a una base de datos integrada con este concepto de SIG.

Otras definiciones de SIG:

- Un sistema para capturar, almacenar, comprobar, integrar, manipular, analizar y visualizar datos que están espacialmente referenciados a la tierra. (Chorley, 1987).

- Sistemas automatizados para la captura, almacenamiento, composición, análisis y visualización de datos espaciales. (Clarke, 1990).

- Un sistema de hardware, software y procedimientos diseñados para soportar la captura, gestión, manipulación, análisis, modelado y visualización de datos espacialmente-referenciados para resolver problemas complejos de planeamiento y gestión. (lectura NCGIA por David Cowen, 1989).

Desde un punto de vista práctico un Sistema de Información Geográfica es un sistema informático capaz de realizar una gestión completa de datos geográficos referenciados. Por referenciados se entiende que estos datos geográficos o mapas tienen unas coordenadas geográficas reales asociadas, las cuales nos permiten manejar y hacer análisis con datos reales como longitudes, perímetros o áreas. Todos estos datos alfanuméricos asociados a los mapas más los que queramos añadirle los gestiona una base de datos integrada con el GIS.

Por ejemplo hoy en México existen varios sistemas de información geográfica, tenemos el caso de INEGI el cual maneja su información geográfica(base de datos espaciales) por medio de un (SIG).

1.2 Historia de los (SIG).

El desarrollo de los Sistemas de Información Geográfica comienza en los años 60´s. Sin embargo hasta principios de los años 80´s se mantuvieron como campo de investigación.

Se traza una visión retrospectiva y prospectiva sobre la tecnología (SIG) donde considera la existencia de tres generaciones de sistemas de información geográfica.

La primera generación se caracteriza por sistemas herederos de la tradición de la Cartografía, con soporte de bancos de datos limitado y cuyo paradigma típico de trabajo es el mapa (llamado de "cubierta" o de "plano de información"). Desarrollados a partir del inicio de la década del 80, y a partir de 1985, para sistemas PC/DOS, esta clase de sistemas es utilizada principalmente en proyectos aislados, sin la preocupación de crear archivos digitales de datos. Esta generación también puede ser caracterizada como sistemas orientados a proyecto ("project-oriented SIG").

La segunda generación de SIGs ("banco de datos geográfico") llegó al mercado a inicios de la década del 90's y se caracteriza por ser concebida para uso en ambientes cliente-servidor, acoplado a gerenciadores de bancos de datos relacionales y con paquetes adicionales para procesamiento de imágenes. Desarrollada en ambientes multiplataforma (UNIX, OS/2, Windows) con interfaces basadas en ventanas, esta generación también puede ser vista como sistemas para soporte a las instituciones ("enterprise-oriented SIG").

La tercera generación, inicia con el crecimiento de los bancos de datos espaciales y la necesidad de ser compartidos con otras instituciones requiere el recurso de tecnologías como bancos de datos distribuidos y federativos. Estos sistemas deberán seguir los requisitos de interoperabilidad, de manera que permitan el acceso a informaciones espaciales por diferentes SIGs.

La tercera generación de SIG puede ser vista aún como el desarrollo de sistemas orientados para el intercambio de informaciones entre una institución y los demás componentes de la sociedad ("society-oriented SIG") [http://revista.robotiker.com/revista_articulos/].

La próxima figura ilustra la evolución de la tecnología SIG fig 1.1

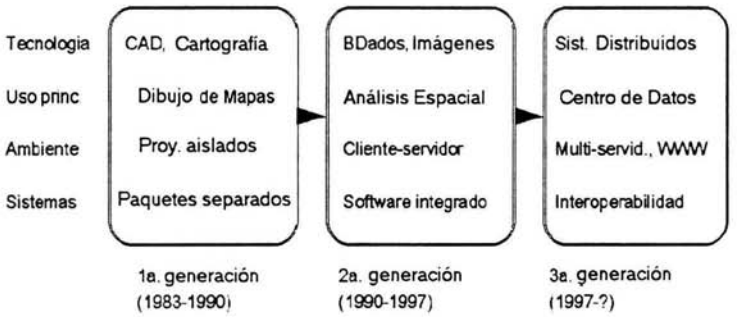


Fig 1.1 Evolución de la tecnología SIG.

La primera generación:

La primera generación de SIG se caracteriza por sistemas con operaciones gráficas y de análisis espacial sobre archivos. Su conexión con gerenciadores de bancos de datos es parcial (parte de las informaciones descriptivas se encuentran en el sistema de archivos) o es inexistente.

Más adecuados para la realización de proyectos de análisis espacial sobre regiones de pequeño y mediano porte, estos sistemas enfatizan el aspecto del mapeo. El sistema permite la entrada de datos sin definición previa del esquema conceptual, asemejándose así a ambientes que poseen la capacidad de representar proyecciones cartográficas y de asociar atributos a objetos espaciales(CAD). Por fuerza de su concepción, tales ambientes no poseen soporte adecuado para construir grandes bases de datos espaciales.

La segunda generación: Banco de Dados Geográficos

La segunda generación de sistemas se caracteriza por sistemas concebidos para operar como un banco de datos geográfico, entendiendo como un banco de datos no convencional aquél donde los datos tratados poseen, además de atributos descriptivos, una representación geométrica en el espacio geográfico.

La tercera generación: Bibliotecas Geográficas Digitales

Una biblioteca geográfica digital (o un "centro de datos geográfico") es un banco de datos geográfico compartido por un conjunto de instituciones. Esta biblioteca debe ser accesible remotamente y almacenar, además de los datos geográficos, descripciones sobre los datos ("metadatos") y documentos multimedia asociados (texto, graficos, audio y video).

Este nuevo paradigma es motivado por el refinamiento de nuestra percepción de los problemas ecológicos, urbanos y ambientales, por el interés en entender, de forma cada vez más detallada, procesos de cambios locales y globales y por la necesidad de compartir datos entre instituciones y la sociedad.[http://www.urbanisme.equipement.gouv]

1.3 Tipos de Sistemas de Información Geográfica.

Se pueden distinguir tres tipos de programas que aunque puedan denominarse conjuntamente SIG tienen diferencias fundamentales en su ámbito de aplicación. En primer

lugar distinguiremos un SIG propiamente dicho, como gran sistema informático que gestiona completamente una base de datos geográficos. Por otro lado delimitaremos las aplicaciones que se han dado en llamar Desktop Mapping (DM)- sistemas de análisis y visualización integrados entre las aplicaciones Desktop de ordenador personal. Finalmente distinguiremos los sistemas de Diseño asistido por Ordenador (CAD) y sistemas afines.

Funcionalidad del GIS:

- Construir datos geográficos: Mediante datos geométricos existentes en CAD, o capturándolos por digitalización, vectorización de imágenes, GPS, etc., el sistema permite depurarlos y estructurarlos topológicamente, asociándolos con bases de datos alfanuméricas. De esta forma se obtienen datos espaciales listos para su uso en el análisis.

- Modelado cartográfico: Creación de nuevos mapas a partir de mapas existentes: Combinado atributos del terreno como pendiente, vegetación, tipo de suelo, etc. Mediante un modelo matemático se pueden crear nuevas variables, como un índice de erosionabilidad, de riesgo de incendios, etc.

- El SIG nos permite analizar los mapas estructurados en combinación con bases de datos asociadas. Se pueden interrogar para seleccionar los datos de interés, ver los resultados interactivamente eligiendo la simbología en función de los atributos asociados y producir cartografía de calidad.

- También se pueden preparar aplicaciones a medida, como un plan de control de incendios, de evaluación de impactos ambientales, un modelo que notifique la evolución de un incendio o de una inundación, aplicaciones verticales como un sistema de gestión municipal o una aplicación para una empresa eléctrica, etc. Interfaz de un SIG Fig1.2

Interfaz de un SIG fig1.2

Limitaciones del SIG :

Un SIG es un gran sistema informático cuya implantación en una organización es siempre gradual y costosa. Se requiere siempre la adecuación del sistema al trabajo requerido, mediante programación (frecuentemente realizada por el suministrador del SIG) y recopilación de los datos necesarios (suministrados por otras organizaciones o introducidos por el cliente).

- Desktop Mapping.

Recientemente han venido apareciendo aplicaciones sencillas de visualización y análisis de datos con componente espacial para sistemas microordenadores con un coste de magnitud inferior a un SIG . La denominación habitual de estas aplicaciones en es Desktop Mapping (DM) "Cartografía de escritorio". En primer lugar, estas herramientas permiten el uso de

datos espaciales (posiblemente creados y estructurados con un SIG) por parte de usuarios que no son expertos en programación, cartografía, geodesia, etc. de forma análoga a como éstos mismos usuarios utilizan procesadores de textos, hojas de cálculo, bases de datos sin conocimientos de tipografía, manutención, teoría de computación, etc. Fundamentalmente los DM permiten ver y analizar la estructura espacial de los datos. Por otro lado, estas aplicaciones sirven también de vehículo para la creación de aplicaciones concretas que trabajen con datos espaciales una aplicación que gestione un inventario o una aplicación vertical para banca.

De la misma manera se pueden incorporar datos no gestionados directamente por el DM, como sonido, imagen de vídeo, fotografías, etc. Este tipo de aplicación tiene un mercado potencial mucho mas amplio que un GIS tradicional, por las mismas razones que lo tienen los procesadores de textos, hojas de cálculo, bases de datos, etc. Permiten crear un modelo geográfico del funcionamiento de un negocio u organización.

Actualmente el DM se usa en: departamentos de mercadotecnia, ventas, distribución y reparto, telecomunicaciones, propiedad inmobiliaria, planeamiento, seguros, servicios de urgencia (bomberos, policía), salud, administración local, etc. En cuanto al uso de un DM como complemento a un GIS, hay que distinguir que algunas herramientas DM trabajan directamente con las bases de datos de un GIS determinado, mientras que otras requieren un formato propio. En el segundo caso, si se quiere utilizar el DM como visualizador de un SIG determinado se debe establecer primero un mecanismo que adapte los datos del SIG a los requerimientos y estructuración del DM. [http://revista.robotiker.com/revista_articulos/].

Funcionalidad del DM:

- Una de las características principales es la Homogeneidad en el tratamiento de los datos: Combinar datos alfanuméricos con cualquier objeto espacial. Los datos espaciales deben poder integrarse sin importar el sistema de referencia geodésico ni la proyección usada por los diferentes archivos. Todo tipo de objeto espacial (puntual, lineal, extenso) debe ser tratado de forma homogénea, sin distinción por parte de las operaciones disponibles de distintos tipos de conjuntos de datos.

- Proporcionar facilidades flexibles de geocodificación, esto es, asignar localización espacial a las bases de datos que el usuario pueda tener.

- Tener la capacidad de incorporar datos en los formatos más populares como por ejemplo datos de tipo SVG (Scale Vector Grapchis) , son datos vectoriales que por medio de un visualizador como Adobe se pueden visuzalizar.

- Ser capaz de representar distintos conjuntos de datos de forma superpuesta como "capas" en un mapa.

- Debe contar con un gestor de bases de datos. En realidad un DM es una extensión de una base de datos.

- Permitir la representación temática de los datos en la forma más flexible posible. Una función básica de estas herramientas es la determinación de la simbología en función de cualquier expresión de los atributos y/o la geometría.

- El sistema debe permitir diferentes vistas de los mismos datos: en forma de mapas que los incorporen como una capa, en forma tabular, gráficos, composición cartográfica, ...

- El sistema debe proveer facilidades para la preparación de documentos integrados (lo cual puede requerir la cooperación con programas de proceso de textos y/o autoedición) generando salidas gráficas con la calidad de presentación adecuada.

- Idealmente debe funcionar en las plataformas mas populares: DOS, Windows, Mac, UNIX NT, OS/2, Linux.

- Es necesaria una buena disponibilidad de mapas y datos del área y temas de interés del usuario.

- Debe contar con un buen lenguaje de desarrollo.

- También es importante un lenguaje de consulta de datos que combine operaciones espaciales con las tradicionales. (Mejor si esta integrado con el lenguaje de desarrollo y es accesible interactivamente).

- Asimismo utilidades de análisis geográfico

- Una utilidad muy interesante es la posibilidad de registrar y superponer imágenes estructuras de tipo raster tipo de grafico que permite ver sombras en la imagen a los datos vectoriales. Así es posible la digitalización en pantalla y el enriquecimiento del grafismo.

Limitaciones del DM.

- No es apto para la creación de nuevos mapas por digitalizacion, escaneado - vectorización, uso de GPS, o incorporación de datos geométricos no estructurados, pues habitualmente no se cuenta con la capacidad de depuración de los datos, creación de topología y transformación para su correcta localización espacial.

- Tampoco se podrán crear nuevos temas combinando datos existentes por análisis de superposición. No se cuenta con funciones avanzadas de manipulación de la topología ni de modelado cartográfico.

- No se pueden gestionar datos tridimensionales. Esto excluye la creación de vistas perspectivas, análisis de visibilidad, etc. (Aunque se pueden crear aplicaciones que implementen alguna funcionalidad de este tipo, como crear curvas de nivel, aplicar modelos de iluminación, ...)

- Tampoco se pueden esperar de un DM las capacidades mas avanzadas de los modernos SIG, por ejemplo existen empresas como son Arcinfo que tienen productos ya terminados que son para levantamientos de datos es decir añadir datos a la información espacial de una manera sencilla, control de concurrencia dinámico, variabilidad temporal de los datos, etc.)

CAD Mapping Systems.

Algunos vendedores de sistemas CAD (Computer Asisted Design - programas de delineación y diseño) han pensado que uno de estos sistemas puede evolucionar fácilmente y convertirse en un SIG. Los sistemas que aparecen de esta forma se denominan CAD Mapping Systems (CMS), y habitualmente el vendedor los llama SIG. Normalmente estos sistemas son el resultado de enlazar dos sistemas existentes en el mercado, un CAD y un RDBMS (Relational DataBase Management System - Sistema de gestión de bases de datos relacional). De esta forma los datos alfanuméricos contenidos en las bases de datos se asocian con elementos gráficos de un archivo - dibujo de CAD. Lo cierto es que un CAD presenta unas características que dificultan el desarrollo de un SIG. Un SIG no es la suma de CAD y RDBMS, mas bien el RDBMS se halla en el corazón del sistema, y las capacidades gráficas, sutilmente diferentes de las del CAD se hallan íntimamente ligadas al RDBMS.**[www.rosenblueth.mx/fundacion/ Numero01/art01_numero01.htm]**

1.4 Utilidad de los Sistemas de Información Geográfica

Los SIG son útiles en los negocios, el gobierno y en la investigación. Por la parte de los negocios se usan para tareas como el estudio de factibilidad del establecimiento de nuevas sucursales según factores como demografía, vías de acceso, niveles socioeconómicos de la población y existencia de competidores, para redefinir los territorios de ventas, en la búsqueda de menores gastos de transportación, de almacenamiento y en menores tiempos en operaciones, que resulta en que se puede dedicar mayor atención para hacer más clientes.

También se usan para elaborar mapas con fines de marketing, turismo, difusión, educación y seguramente otras áreas, porque difícilmente habrá tareas de administración, planeación y estudio que no sean más fáciles o que no den mejores resultados con un SIG

En cuanto al gobierno los GIS son una importante herramienta para la comprensión y preservación de nuestro medio ambiente. Son utilizados en esfuerzos para controlar la contaminación, proteger especies de extinción e identificar y comprender los hábitats de animales.

Sirven además para la gestión del catastro, el desarrollo urbano, el registro público, instalaciones eléctricas, de agua, gas y comunicaciones. Seguimiento de transportes en tiempo real. Búsqueda de las mejores rutas de transportación y distribución además de que están siendo utilizados para combatir el crimen. Respuesta a emergencias, seguridad pública, inventarios arqueológicos, identificación de áreas de riesgo, reservas ecológicas y recursos naturales.

Podemos citar SIG a modo de muestra, clasificadas en diferentes apartados:

Redes de distribución y transporte.

- Análisis de accesibilidad en el plan director de infraestructuras.- Los GIS constituyen una herramienta muy apropiada para el cálculo de los niveles de accesibilidad y la cartografía de los resultados finales.

Planificación urbana.

SIG en estudios de urbanismo y medio ambiente.- Redacción y Desarrollo de Planes Generales y Normas Subsidiarias, Redacción de Planes Parciales, Proyectos de Urbanización, Proyectos de Compensación y Reparcelaciones, Evaluaciones de Impacto Ambiental, Planes Especiales, Catálogos, son tareas que se han encomendado a los GIS en los equipos de Urbanismo y Medio ambiente.

Ciencias de la tierra.

El Consejo de Recursos Minerales es el organismo de información geológico-minera, cuenta con una importante experiencia como generador de información de ciencias de la tierra en el país; su experiencia se basa en más de cincuenta años de trabajo en diferentes especialidades y escalas dentro del territorio nacional.

El Consejo de Recursos Minerales(COREMI) cuenta con SIG quien ha compilado la información minera, geológica, de yacimientos minerales, producción minera, sistemas de minado, beneficio de minerales, así como el potencial y perspectivas geológico-mineras. **http://www.coremisgm.gob.mx/inicio.html**

Grandes bases cartográficas.

SIG en el INEGI.- Este proyecto hace referencia las funcionalidades que el GIS proporciona al Instituto Nacional de Estadística (INEGI). Desarrollado por la Unidad de Cartografía Digital, utiliza información geográfica digitalizada producida por otros Organismos de la Administración a la que se añade la información geográfica específicamente estadística: las Secciones Censales, permitiéndose su conexión con la información alfanumérica de las bases de datos estadísticas.

Secretaría de Estado de Hacienda del Ministerio de Economía y Hacienda organizó un sistema de gestión y difusión de los datos catastrales, que además de facilitar el mantenimiento del inventario referenciado espacialmente de bienes inmuebles y su valoración, permite desarrollar una gestión integrada que facilita la gestión tributaria y la toma de decisiones de las distintas Administraciones Públicas, basadas en el análisis y elaboración de la información recogida en el Inventario Catastral.

Medio ambiente.

La Secretaría de Medio Ambiente, Recursos Naturales y Pesca (SEMARNAP) tiene un Sistema de Información Geográfica tiene como fin de atender los problemas de desarrollo, fomentando criterios y normas que permitan el manejo y protección de los recursos y presenta la oportunidad de atender de forma integral los problemas de desarrollo, y protección de los recursos y del medio ambiente. **[http://www.uaaan.mx/proders/proders_2.html]**

CAPITULO *II*

SISTEMAS DE INFORMACION GEOGRAFICA EN PEMEX

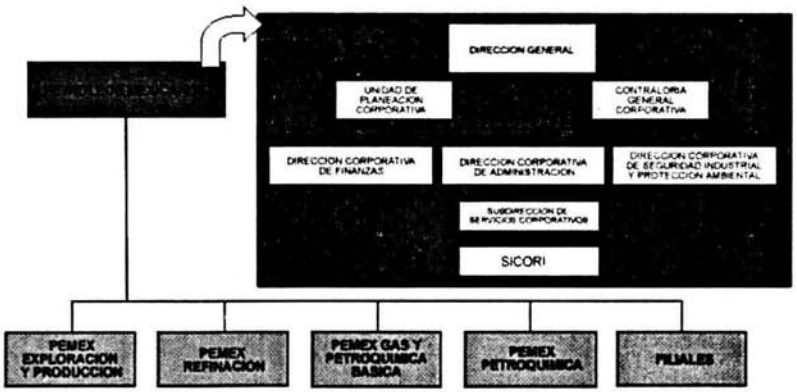
2.1 Introducción

Hasta antes de los 90 Se usaban los programas el software de cad(Computer Asistent Designer) en algunas áreas de ingeniería siendo los más comunes como autocad de autodesk y el micro station de bentley que principalmente se usaban para elaborar planos conocidos como plon -plan de instalaciones petroleras en el área de exploración además del software de CAD se comenzó a usar un software de tipo de GIS con el que se produjo cartografía escala 1 a millón, utilizando para este propósito el GEOSTART y fue hasta 1992 cuando empezó a operar formalmente el SICORI(Sistema Corporativo de información Geografica).

Con la tarea principal de escánear georeferenciar y vectorizar la cartografía nacional de INEGI para integrarla en un sistema de información geográfica, al mismo tiempo en la exploración se comenzó con la captura de todos los pozos a nivel nacional para utilizarlos en trabajos de simulación apoyados por la compañía Schulumberger para lo que se utilizó el software finder así mismo la región sur empezó a utilizar software de información geográfica para posicionar sus instalaciones actualmente son pocas compañías particulares la que participan para dar servicios a PEMEX y prácticamente todos los servicios provienen de SICORI.

2.2 *SICORI (Sistema Corporativo de Información Geográfica)*

La Unidad Corporativa de Sistemas de Información Geográfica -SICORI MR- es la dependencia de PEMEX que a nivel corporativo, proporciona servicios de información geográfica relacionados con la industria petrolera y su entorno físico, socio-político, económico y ecológico; especialmente en aspectos relacionados a la logística y seguridad de las instalaciones. Lo anterior para apoyar la toma de decisiones a niveles estratégicos tácticos y operativos.



PRESENTACION INSTITUCIONAL

Fig 2.1

PRESENTACION INSTITUCIONAL
ORGANIZACIÓN SICORI

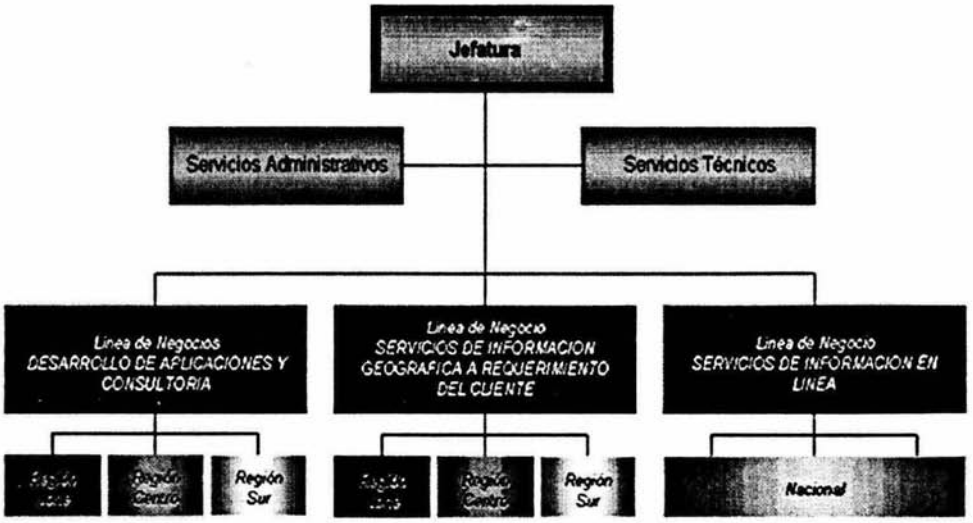


Fig 2.2 Organización SICORI

Estructura tecnológica

Región Sur
Villahermosa:
16 est. de trabajo
2 serv. impresión

Región Centro:
Distrito Federal:
12 est. de trabajo

Región Norte:
Distrito Federal:
9 est. de trabajo

Infraestructura Central de soporte
1 serv. base de datos Oracle (cluster)
1 servidor de archivos
1 servidor de dominio
4 servidores de impresión
1 servidor de Web

Sistema de Información Geográfica basado en el modelo georeferencial (archivos y base de datos), sistemas de información administrativa Institucionales (SAP) y locales.

2.3 Creación y Evolución (SICORI).

Proyecto sistema corporativo de información geográfica(sicori adscripción a la unidad de planeación, dca .

Con el propósito de apoyar el proceso de cambio estructural que se estaba gestando en la empresa, mediante el uso de la tecnología de Sistemas de Información Geográfica, el 25 de abril de 1991, el H. Consejo de Administración de PEMEX aprobó el desarrollo del "Sistema Corporativo de Información -SICORI-".
El 20 de febrero de 1992 se inició el Proyecto Sistema de Información Geográfica de Petróleos Mexicanos con la autorización de su integración y adscripción formal al área de Planeación Estratégica de la entonces Subdirección Técnica Administrativa
Módulos

Petroquimica y refinación (1993).

1993 se determina descentralizar módulos del SICORI, para depender de los actuales Organismos Subsidiarios, coordinados técnicamente por la unidad central, con el propósito de atender individualmente sus requerimientos particulares, por lo que se crean los Módulos de Refinación y Petroquímica.

UNIDAD CORPORATIVA DE SISTEMAS DE INFORMACIÓN GEOGRAFICA
SICORI – UCSIG

ADSCRIPCIÓN A LA SUBDIRECCIÓN DE SERVICIOS CORPORATIVOS, DCA
Módulos Villahermosa y Coatzacoalcos

1994-2001

El 11 de julio de 1994, se cambia la adscripción de SICORI a la Subdirección de Servicios Corporativos, dependiente de la Dirección Corporativa de Administración, su denominación oficial a partir de esa fecha es la de Unidad Corporativa de Sistemas de Información Geográfica (UCSIG).

Evolución Administrativa

En 2001 se realiza una reestructuración interna y se determina crear siete líneas de negocio, una de Desarrollo de Aplicaciones y otra de Productos a Requerimiento del Cliente en cada Región (Norte, Centro y Sur) y una de Servicios de Información en Línea a nivel nacional, con dos áreas centrales de soporte técnico y administrativo para apoyar la operación regional de las siete líneas de Negocio

Evolución tecnológica

ESTRUCTURA CENTRALIZADA (1992).

Plataforma tecnológica Intergraph, totalmente propietaria en hardware, sistemas operativos y software de aplicación,

Hardware: 1 servidor de base de datos (Oracle)
6 estaciones de trabajo (Intergraph)
Sis. Oper.: Versión UNIX de Intergraph (clix)
Software: GIS de Intergraph
Base de datos Oracle
Software cartográfico Intergraph
Red: Ethernet
Seguridad: Base de datos
Acceso lógico
Sistemas: Propietarios

ESTRUCTURA DISTRIBUIDA (2001)

Plataforma tecnológica Windows, sistemas abiertos, proveedores Intergraph, Data General, HP, Digital.

Hardware: 11 servidores (base de datos, impresión, archivos y Web)
36 estaciones de trabajo (intergraph)
Sis. Oper.: Windows NT/ Unix (clix)
Software: GIS de Intergraph
Base de datos Oracle
Software de apoyo Microsoft
Redes: Institucionales ATM/ E1/ Ethernet 100 Mbs
Seguridad: Base de datos, dominios, control de acceso físico y lógico
Sistemas: Abiertos, cliente-servidor

2.4 Servicios del SICORI y sus productos.

Existen tres líneas de negocio donde se atienden servicios y productos de información geográfica esto en respuesta a la demanda de Petróleos Mexicanos sus Organismos subsidiarios.Desarrollo de Aplicaciones y Consultoría.

1. Servicios de información geográfica a requerimiento del cliente.
2. Desarrollo de Aplicaciones y Consultoria.
3. Servicios de Información en línea. Modelo fig 2.3.

Fig 2.3 Las Lineas de negocio SICORI

Productos y Servicios

EL SISTEMA CORPORATIVO DE INFORMACIÓN GEOGRÁFICA DE PEMEX

SICORI pone a disposición de los clientes de la Industria Petrolera, su Sistema de Información Geográfica (SIG), constituido por un sistema informático que emplea la mejor tecnología en equipo de cómputo, software e información y es capaz de realizar una gestión completa de datos geográficamente referenciados -mapas cuyos datos tienen asociadas coordenadas geográficas reales-; esta condición hace posible manipular, gráfica y analíticamente, la información de Petróleos Mexicanos y su entorno físico, económico y sociopolítico.

Construcción e integración de datos geográficos: diseños en CAD, incorporación de información gráfica en papel u otras fuentes, asociación con bases de datos alfanuméricas.
- Edición cartográfica: creación de nuevos mapas a partir de mapas existentes, combinando atributos del terreno como pendiente, vegetación, tipo de suelo, etc. Permite producir cartografía de calidad.

Fig 2.8 Integrando Datos

- Análisis espacial: permite analizar los mapas estructurados en combinación con bases de datos asociadas. Se puede interrogar para seleccionar los datos de interés, ver los resultados interactivamente y desarrollar aplicaciones a la medida. Puede responder en tiempo real en formatos gráficos o alfanuméricos a las siguientes preguntas: ¿Qué elementos o datos se localizan en un punto o área determinada (ubicación)?, ¿Dónde se encuentran ciertos elementos que independientemente o combinados reunen ciertas condiciones (localización)?, ¿Qué tendencias o diferencias existen entre áreas o lapsos de tiempo distintos (tendencias)?, ¿Que patrones de distribución espacial existen en determinada área y donde se localizan (distribuciones)?, ¿Que resultados se obtienen si ocurre un hecho determinado (modelación empleando leyes científicas)?

Acervo de información de SICORI

SICORI ha integrado desde 1992 un gran acervo de información, necesaria para la generación de servicios, entre la que destaca la siguiente: Cartografía de nuestro País en distintas escalas a partir de la información de INEGI, red carretera nacional , imágenes de satélite y fotografías aéreas de las áreas de actividad petrolera, planos de conjuntos de instalaciones (plot plants), censos económicos y de población del País, mapas urbanos, comicios electorales e instalaciones e inmuebles de Petróleos Mexicanos y Organismos Subsidiarios (plantas de procesamiento, derechos de vía, pozos petroleros, ductos, centros administrativos, etc.).[http://www.sicori.dca.pemex.com/]

Fig 2.9 Cartografia de nuestro pais en diferentes escalas

2.5 Proyectos y Servicios Desarrollados

En más de nueve años de operación en la Industria Petrolera, SICORI ha adquirido gran experiencia en la integración, procesamiento y generación de información geográfica, particularmente en lo que se refiere a cartografía petrolera. Ha desarrollado especialistas en procesamiento e interpretación de imágenes de satélite y de fotografía aérea; en el uso de herramientas computacionales de análisis de rutas, modelado de terrenos, etc.; Cuenta con especialistas en el desarrollo de sistemas de información, así como en el diseño y administración de bases de datos y redes digitales de comunicación.

Servicios relevantes

- Edición y generación de mapas y espaciomapas
- Procesamiento de imágenes y modelado de terrenos
- Actualización de cartografía a partir de imágenes y fotografías
- Desarrollo de sistemas de información administrativa
- Desarrollo de sistemas de información geográfica
- Edición de planos de ingeniería y asesoría en el desarrollo de sistemas

En las siguientes figuras se muestra algunos de los servicios que proporciona SICORI .

Fig 2.4 Espaciomapa

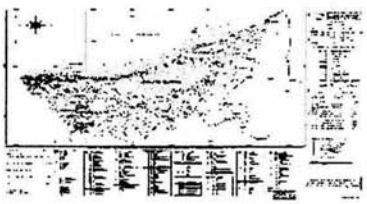


Fig 2.5 Plano cartográfico petrolero

Fig 2.6 Imagen de Satelite

Fig 2.7 Modelo Digital de terreno

Proyectos en desarrollo

Sistema Institucional de Información Geográfica de Ductos e Instalaciones.
Sistema de Información Geográfica para Instalaciones de Telecomunicaciones.
Sistema Institucional de Información Geográfica para el Catastro Petrolero.
Sistema de Información Geográfica para Ejecutivos de la Subdirección de Servicios Corporativos.
Sistema de Información Geográfica para la Reserva de la Biosfera Pantanos de Centla.
Sistema de Información Geográfica para el SIASPA.

Dentro de los organismos subsidiarios de PEMEX se encuentran varios proyectos:

Sistemas integrales diseñados, desarrollados e implantados por SICORI.

A lo largo de 10 años, el SICORI ha dirigido sus esfuerzos a satisfacer las necesidades de sus clientes en el ámbito de la información geográfica.
Una de las líneas de negocio más importantes es la denominada ***Desarrollo de Aplicaciones y Consultoría*** y está dedicada a proporcionar servicios integrales, a la medida de las necesidades de las áreas técnicas y administrativas de Petróleos Mexicanos.
Entre los sistemas están los siguientes:

- SIIGEDI "Sistema Institucional de Información Geográfica de Ductos e Instalaciones."

Beneficio.- Que los usuarios -las áreas de diseño, operación y mantenimiento de ductos de los Organismos Subsidiarios -,cuenten con información integrada, geográfica y alfanumérica para la administración de los Derechos de Vía y Ductos de Petróleos Mexicanos, así como con una herramienta para la planeación y toma de decisiones.

- SIIGCAP "Sistema Integral de Información Geográfica para el Catastro Petrolero."

Beneficio.- Que los usuarios -las áreas de administración patrimonial de PEMEX, Organismos Subsidiarios y SENER-, cuenten con un catastro petrolero actualizado, que cumpla con la normatividad vigente en materia de hidrocarburos, facilitando así la administración del patrimonio de Petróleos Mexicanos y a la vez satisfacer las necesidades de consulta y análisis de información de los usuarios del sistema.

- SIGARTT "Sistema de Análisis de Rutas para Transporte por Carretera de Productos Refinados".

Beneficio.- Que el usuario -Gerencia de Transporte Terrestre de PEMEX-Refinación-, cuente con toda la información carretera, localidades, puntos de embarque y entrega y los puntos y montos de peaje, para un adecuado análisis de rutas para hacer más eficiente y eficaz la distribución y entrega de los productos petrolíferos.

- Sistema desarrollado para la GERENCIA DE INGENIERÍA DE TELECOMUNICACIONES

Beneficio.- Que los usuarios -la GIT-, puedan realizar análisis espacial de las áreas de cobertura de comunicación *truncking* y de otras compañías de telefonía a nivel nacional.

- Atlas de Pasivos Ambientales

Beneficio.- Los usuarios -la DCSIPA-, podrán tener acceso vía Intranet y de manera eficiente y sencilla la información de instalaciones petroleras y analizar el entorno físico-espacial de los pasivos ambientales para la mejor toma de decisiones

- SIINFA "Sistema Integral de Información Ambiental Golfo de México."

Beneficio.- Que los usuarios del sistema -Región Marina Noreste de PEP- puedan planear, prevenir y evaluar los riesgos ambientales asociados con la operación marina de PEMEX-Exploración y Producción en el Golfo de México

- SIGSREST "Sistema de Información Geográfica de Sitios Restaurados en la Región Sur de PEP."

Beneficio.- Que los usuarios -la GSIPA de PEP-, cuenten con información histórica integrada de las acciones de restauración de sitios contaminados en la Región Sur de PEP.

- "Sistema de Información Geográfica de Manifestaciones de Hidrocarburos en la Región Sur de PEP."
- Beneficio.- Que los usuarios -la Gerencia de Planeación de la Región Sur de PEP-, cuenten con información para determinar oportunidades exploratorias a través de estudios de manifestaciones superficiales de hidrocarburos.

Actividades que realizan las tres líneas de negocio del (SICORI).

1.- Desarrollo de Aplicaciones y Consultoría.

PEMEX-Corporativo

Dentro de sus actividades que se encuentra el desarrollo de aplicaciones para el Sistema de Información Geográfica de Catastro Petrolero (SIIGCAP) donde se generó el archivo de asignaciones petroleras; Para el Sistema de Información Geográfica ejecutiva de la Subdirección de Servicios Corporativos (SIGESSC).

PEMEX -Refinación

Dentro de sus actividades están las Instalaciones marinas del Golfo de México; localización del poliducto Minatitlán México y ubicación de las tomas clandestinas.

Otra de sus actividades es el desarrollo de una interfaz gráfica y la generación de la información gráfica de protección civil; para el Sistema de Información Geográfica Atlas Ambiental de Petróleos Mexicanos .Donde se migrara toda la información que se tiene capturada a la fecha, para la consulta posterior en la Intranet de PEMEX .

Y se continúa con el procesamiento de información gráfica de los Organismos Subsidiarios.

2.- Servicios de información geográfica a requerimiento del cliente.

Dentro de sus funciones se encuentra el Sistema Institucional de Información Geográfica de Ductos e Instalaciones, también se levanto y proceso información de ductos sectoriales Monterrey y Torreón. A demás de que implanto el sistema de la Región Centro, Golfo y Sureste.

Esta línea de negocio se encuentra trabajando en el cálculo de distancias de ductos entre las terminales de almacenamiento y Distribución, las Refinerías; poliducto Cadereyta-Saltillo –Zacatecas con perfil topográfico.

PEMEX-Petroquímica

Complejos Petroquímicos Escolín, Cosoleacaque, Cangrejera, Tula e Independencia; trayectoria de los ductos de paraxileno, benceno e hidrógeno de la Petroquímica Cangrejera a la Refinería de Minatitlán.

PEMEX-Gas y Petroquímica Básica .

Ubicación de la zona de riesgo de la Terminal Terrestre de Salina Cruz, Oaxaca. Complejo procesador de Gas en Ciudad PEMEX, Tabasco; análisis de riesgo en la Terminal de Gas Licuado e Puebla.

PEMEX- Exploración y Producción

Estudios de riesgos de las baterías de separación del campo Cinco presidentes I y II; baterías de separación Otates , Ogarrio II Cinco presidentes I Bellota 114; apoyo en el proyecto de construcción del oleoducto de 20 Paredon y área de mezclado y distribución El misterio I; ubicación de polígonos de los proyectos Pidiregas en la República Mexicana ; Activo Luna con imagen de satélite; instalaciones estratégicas en la Región Sur localización de los pozos exploratorios del Activo de Producción Cinco presidentes con los derechos de vía, oleoductos y gasoductos; Bateriítas Cunduacán y Paredón con sus instalaciones petroleras con un radio de influencia de 10 a 15km, respectivamente Campos Hormiguero y Cobo.

3. *Servicios de Información en línea.*

Esta línea de negocio da servicio a todas las dependencias de PEMEX a nivel nacional, dentro de sus actividades esta la elaboración de un Sistema de Información Geográfica Petrolera el cual permite acceder a una gran base de datos geográfica cual permite elaborar mapas dinámicos a la medida y las necesidades de los clientes , en cuanto a información y cobertura .

Otra de sus actividades es la generación de Productos de Catalogo.

Donde se encuentra amplia variedad de mapas y productos de información geográfica, seleccionados de manera ágil y sencilla a partir de un catalogo estructurado en temas y categorías que facilitan su localización y selección.

Dentro de sus actividades también está la generación de mapotecas y bibliotecas digitales, con el fin de poder acceder al acervo de información geográfica petrolera para poder subir y almacenar todo tipo de imágenes , planos y mapas digitales.

También se encuentra dentro de sus actividades el proceso de comunicación con sus clientes en donde se entiende la importancia de la mercadotecnia como una herramienta vital de toda organización moderna .sus funciones principales esta:

Comunicaciones personalizadas estructuradas y periódicas para difusión promoción y venta , mantenimiento investigación de necesidades y evaluación de servicio .

2.6 Conclusión.

Unidad Corporativa de Sistemas de Información Geográfica es quien proporciona servicios de información geográfica a todo PEMEX tanto en lo socio-político , ecológico, con ello permite tener mejores decisiones en lo estratégico, táctico y operativo .

La innovación es entendida como fundamental para el sector público y en la empresa paraestatal en el México de hoy, por lo que, debe ser vista como una herramienta de mejoramiento en PEMEX para romper con viejos paradigmas y reinventar nuestros procesos enfocándolos a las mejores practicas en el mundo entero.

CAPITULO III

NECESIDADES DE UN SIG CORPORATIVO PETROLERO

3.1 Identificación de necesidades

Diagnostico del problema

Actualmente PEMEX cuenta ya con un SIG esta condición hace posible acceder a una gran base de datos geográficos manipular grafica y analíticamente la información de Petróleos Mexicanos (PEMEX) en donde para acceder al sistema los clientes necesitan registrase vía Web en SICORI y estar conectados a la Intranet de PEMEX .

El problema está que para que un cliente pueda ver el Sistema de Información Geográfica tiene que esperarse aproximadamente entre 5 y 10 minutos para poder trabajar con el sistema estó es debido a que la información geográfica (Información espacial) llega al cliente , pero estó es la Intranet de PEMEX , ahora si el sistema necesita ser requerido fuera de la Intranet de PEMEX el tiempo se incrementa demasiado esto debido a que se accede por vía MODEM y se requiere cargar demasiados archivos del lado del cliente a este sistema se le conoce como cliente pesado tiene la funcionalidad que se requiere pero entre más funciones tenga mas archivos necesitan ser cargados en el cliente esto cada vez que se accede al sistema .
La arquitectura que tiene el sistema actual es un modelo de tres capas pero podemos observar en la figura que la información geográfica (Información Espacial) viaja del servidor hacia el cliente

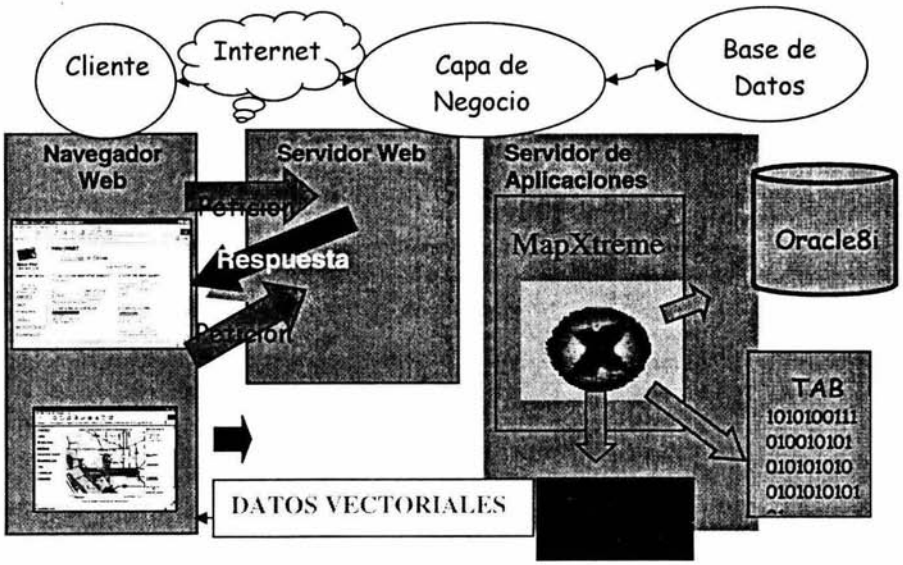


Fig 3.1
En este modelo de tres capas los datos vectoriales son enviados al cliente

Este sistema genera clientes mucho más pesados

- Menos prácticos para su "bajado" dinámico
- Plug-ins propietarios del browser
- Applets Java "Pesados" Java / Java Web Start
- Permite interfaces de usuario más sofisticadas
- Permite a los clientes interactuar con datos locales

En cuanto al a infraestructura tecnológica se cuenta con:

1 Servidor. Base de datos Oracle (cluster)
1 servidor de archivos
1 servidor de dominio
2 servidores de impresión
1 servidor de Web
5 estaciones de trabajo
También se cuenta con redes de alta velocidad, Sistema de Información Geográfica basado en el modelo georeferencial (archivos y base de datos), 1 licencia de MapXtreme Java 4.0, 1 licencia Jdeveloper, 2 licencias suite de oracle 8i, 1 licencia cartucho espacial, 1 licencia de mapinfo, software de apoyo Microsoft.

El reto es realizar un sistema que mejore al que este trabajando .

Requerimientos

Se requiere de un Sistema de Información Geográfica petrolera (SIGP).

En donde el sistema pueda crear mapas con características especificas que requiere el cliente es decir necesitamos construir un sistema de mapas interactivos donde se pueda tener acceso a la información petrolera tanto geográfica (Base datos espacial, imágenes Raster etc..), como alfanumérica(datos relacionados a elementos geográficos).

Funciones que debe tener el sistema .

Herramientas de visualización.

- Acercar mapa
- Alejar mapa
- Centrar mapa
- Visualización inicial.

Herramientas de análisis espacial.

- Herramienta de medición .
- Trazo de polígonos

Herramientas de información.

- Información de un elemento
- Búsqueda por elemento
- Latitud Longitud

Herramientas levantamiento de información cartográfica.

- Control para agregar capas

fig 3.2 Control de capas

Herramientas de control de información cartográfica.

- Control para niveles de capas

Representación grafica de los elementos.

- Presentar diferentes características de los elementos
- Herramientas de Etiquetado

Herramientas de impresión y almacenamiento de resultados.

El sistema debe responder de una manera rápida y eficaz,
Donde el cliente pueda crear sus propios mapas de acuerdo a sus requerimientos.

Con este sistema se pretende resolver los siguientes problemas:

Satisfacer las necesidades de los organismos subsidiarios de PEMEX en cuanto información geográfica se refiere.
Se proporcionara información de tipo:

Carreteras
Hidrológica
Localidades
Pozos
Ductos
Comunicaciones
Terracería
Petrolíferos
Otros

De tal forma que toda esta información la pueda manipular el cliente con las herramientas ya mencionadas. Esto para diferentes propósitos como pudieran ser:

Atmosféricos: Modelos de la circulación general empleados para previsiones del tiempo atmosférico o del clima a varias escalas espaciales (global, regional o local). Modelos de la difusión de contaminantes atmosféricos, originados en fuentes puntuales, lineales o poligonales.
Hidrológicos: Modelos de aguas superficiales: lluvias, simulación del flujo de corrientes, hidráulica de los flujos, modelos de aguas subterráneas: contaminación, modelos de transporte y flujos de variables saturadas.

Tierras/procesos superficie-subsuelo: Crecimiento de plantas, erosión o salinización del suelo, contaminación de suelos, etc.

Sistemas biológicos/ecológicos: Modelos terrestres de una o varias especies de vegetación y/o fauna. Modelos para lagos y ríos, modelos marinos: migración de la pesca, efecto de la pesca.
Modelos integrados: Combinan varios de los anteriores.
Igualmente modelos de sistemas de producción y uso de energía, propagación de ruidos, calidad visual del paisaje y otros similares también se integran en este apartado ambiental.

Económicos: Modelos de comercio internacional, desarrollo económico, input/output análisis, modelos de suelo urbano, mercado de viviendas. Modelos normativos basados en la teoría de la localización (minimización de costes de transporte).

Geográficos: Difusión espacial de innovaciones, migración de población (distancia y atracción/repulsión de lugares), dinámica demográfica de la población, interacción espacial y localización espacial, modelos de comportamiento espacial.

Sociológicos: Segregación espacial de grupos sociales, invasión de territorios por grupos de población, Espacios de acción.
Transportes: Modelos de viajes de personas y bienes, generación de flujos, elección de destinos, modo de transporte, modelos normativos para planificación del transporte.

Modelos espaciales y SIG
Discusión: "base de datos" frente a "caja de herramientas".
Los elementos normales de la caja de herramientas de un SIG poco útiles para realizar modelos matemáticos.

Beneficios de la implementación.
Ofrecer servicios de información geográfica a todos los Órganos que conforman la industria Petrolera Estatal.

3.2 Alternativas

Para la resolución de este problema se plantea:

Propuesta

Se pretende dar servicio a los trabajadores de PEMEX para poder acceder alas base de datos que se tiene mediante un sistema de Mapas Interactivos.

Con esto se pretende mayor cobertura del mercado petrolero, mayor facilidad de acceso a los servicios.
Consolidación y estabilización de la organización así como la consolidación de la diversificación de servicios.

Para llevar a cabo la aplicación del sistema se requieren bases para almacenar coordenadas espaciales relacionadas con información alfanumérica en donde ya se tenga capturada la información georeferencial de las diversas capas de información a presentar en el sistema donde se encuentren ya normalizadas de acuerdo a las necesidades de la aplicación ,también se debe contar con un motor de mapas que controle las bases así como de una presentación de la información hacia el cliente.

Recursos

Para la implementación del sistema se requiere

Bases de datos espaciales en Oracle, Archivos nativos (Tab). Imágenes Raster, Un servidor de aplicaciones que en este caso utilizaríamos Apache (el cual tiene la característica de actuar como un contenedor en la capa de la lógica) , por ser un estándar en muchas empresas y por su gran compatibilidad con diferentes sistemas además de la facilidad de integración de componentes, así como por la seguridad que ofrece hacia el exterior y por ser un sistema estable como un servidor robusto.

Servidor Web que se utilizara será Tomcat por la compatibilidad que ofrece con Apache, además del soporte que ofrece para toda la tecnología que rodea JAVA como son servlets, jsp, arquitecturas en diferentes capas , así como la facilidad que tiene para llevar un control de usuarios, una fácil configuración, y por el desempeño que llega alcanzar interactuando con Apache, a este Web Server se le conoce como el motor de servlets.

También necesitamos un motor de mapas que interactúe con las bases de datos georeferenciadas que se encuentran en diferentes gestores de bases de datos como son tablas en Oracle, archivos tab., archivos en GML(información Vectorial), así como la manipulación de imágenes de satélite (Raster), imágenes en diversos formatos. El cual se capaz de formar un mapa de acuerdo a los parámetros que se establezcan por el sistema además de que se capaz de poder enviar información al cliente en diferentes formatos y para se propone utilizar tecnología de MapXtreme Java 4.5 que permite realizar las necesidades del sistema además de que cuenta con un API de desarrollo para poder trabajar de acuerdo a las necesidades del cliente está tecnología permite la interacción más fácil con las bases de datos espaciales que se encuentran en PEMEX está soportado por mapinfo en generación de mapas esta tecnología es independiente de plataforma ya que esta creado con tecnología JAVA además de que nos permite trabajar con arquitectura MVC Modelo Vista Control esto permite el mantenimiento del sistema , tanto como su crecimiento. La siguiente figura muestra el modelo de tres capas utilizando un a arquitectura (MVC), en este modelo los datos vectoriales no viajan a través de la red .

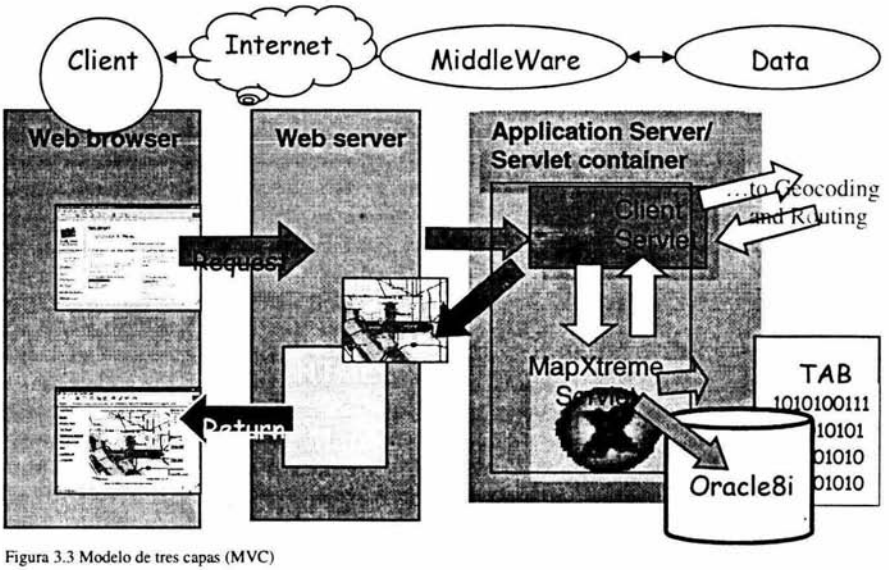


Figura 3.3 Modelo de tres capas (MVC)

MapXtreme proporciona una lista extensa de potentes funciones de gestión de mapas para resolver sus necesidades incluyendo (pero no limitándose a):

- Mapas temáticos que representan estudios comparativos dependiendo de valores asociados a los elementos del mapa.
- Soporte de sistemas de coordenadas y de proyección.
- Capas de anotación de usuario para el seguimiento GPS, edición y etiquetado dinámico.
- Acceso a una amplia variedad de fuentes de datos tales como:
  tablas de MapInfo, Oracle*8i.*
  Spatial
  DB2 SpatialWare Extender, tablas compatibles con JDBC.

- Gestor de definición de mapas, que permite manipular visualmente una colección de capas y estilos de mapas almacenados, como archivos XML o dentro de la base de datos.
- Representación sofisticada de mapas. Simbología de base vectorial escalable, junto a una amplia gama de símbolos, estilos de línea, patrones de relleno, transparencias y fuentes

Tiene varias tecnologías asociadas a su alrededor por mencionar algunas MapXtend, Sistemas de ruta optima, Sistemas de localización, las cuales permite transmitir información por diferentes protocolos de comunicación http, wml , etc..

Con esta herramienta tenemos tres alternativas

Cliente pesado.

Con esta Arquitectura es con la que se esta trabajando actualmente la información vectorial es enviada al cliente además de también son enviados varios beans del lado del cliente que permiten manipular la información vectorial con este modelo de 2 capas, el tiempo de procesamiento lo lleva el cliente y el sistema depende de la capacidad del equipo que este utilizando el cliente

Cliente medio

Con esta Arquitectura el trabajo de procesamiento lo lleva el servidor en este modelo la capa de negocio es manejada por servlets , Java Beans , y la interfaz de usuario es un Applet con el cual se puede diseñar una interfaz muy amigable con el cliente , en este tipo de arquitectura no se envía los datos vectoriales al cliente solo se envía una imagen la cual es elaborada por el motor de mapas (MapXtreme) y es enviada hacia el cliente.

Cliente ligero

Con esta Arquitectura toda la carga la lleva el servidor utilizando un modelo de tipo MVC(Modelo Vista Control) este tipo de modelo permite resolver a diferentes tipos de clientes (html., wml etc) al cliente soló se envía datos e imágenes la interfaz es construida con JavaScript y la vista que genera el servidor.

En la figura se pueden ver los diferentes tipos de cliente que se maneja.

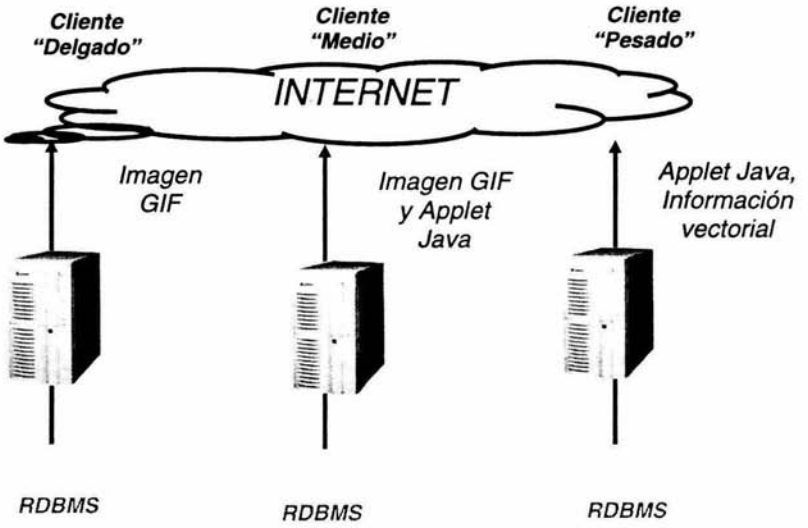


Fig 3.4 Tipos de cliente utilizando MapXtreme

Para la construcción del sistema utilizaremos 3 licencias de desarrollo Java Developer 9i por las herramientas que ofrece para la interacción con Oracle, por la fácil creación de componentes de negocios además de que permite interactuar de una manera fácil con

archivos en XML, también nos permitirá modelar en UML, y poder seguir la arquitectura de MVC, ofrece un ambiente de desarrollo gráfico para agregar herramientas creadas con java botones cajas de diálogo, paneles, etc..

Permite agregar beans creados con JAVA y una utilización fácil de los mismos además nos permite una fácil interacción con sistemas ajenos.

La información que se despliegue en el cliente podrá ser vista con cualquier navegador que tenga soporte para java.

Existen diversos caminos para la resolución del problema pero se opta por esta tecnología puesto que JAVA es la tecnología mas madura y por la compatibilidad con diferentes componentes.

Diferentes soluciones

Es el que propone la empresa de Autodesk

llamado Autodesk MapGuide esta tecnología incluye un nuevo formato de almacenamiento basado en XML, que amplia las capacidades de desarrollo de aplicaciones.

En este tipo de sistemas se utiliza Visual Studio Net el cual podrá permitir agregar mas funcionalidad al sistema de mapas interactivos con combinación de JavaScript del lado del cliente, esto debido a que el API que se maneja esta desarrollada con tecnología de Microsoft , el visualizador que se presenta en Internet el cual se puede descargar desde Internet , están desarrollados por componentes de Microsoft .

En el cliente se necesita descargar el visualizador gratuito Autodesk Map Guide Viewer es altamente interactivo y opera con su navegador: Disponible para Netscape e Internet Explorer para Windows, Completamente personalizadle a través de una sólida API Requiere descarga e instalación.

3.3 Estudio de Viabilidad.

Viabilidad Operativa

Restricciones de la puesta en marcha:

Al incrementar el número de usuarios que utilicen el sistema se requiere un servidor de aplicaciones más grande para dar soporte a todos los usuarios.

Funcionamiento y rendimientos requeridos

El sistema debe de ofrecer el servicio de poder elaborar mapas dinámicamente y de acuerdo a las necesidades del cliente, con herramientas para manipulación del mismo.

El sistema funciona en un modelo de tres capas en donde se encuentras las bases de datos, servidores y cliente.

Permite un desplegado de las capas geográficas, así como la realización de consultas a una serie de bases de datos.

Realizando cálculos y análisis espacial y permitir la consulta a dichos atributos relacionados.

Impresión de resultados así como el de guardar la información del mapa.

En cuanto al rendimiento depende la capacidad de procesamiento del servidor de aplicaciones, para realizar conexiones y manipulación de las bases, tanto la capacidad de recepción del cliente.

El sistema estará bajo estándares lo cual le permitirá crecer, en cuanto información geográfica y tecnológica además de que actuara como integrador de aplicaciones.

Así el sistema estará preparado para los futuros cambios tanto para la competencia sirviendo para aplicaciones futuras.

Viabilidad Legal

El sistema esta bajo normas de Aseguramiento de calidad en donde se encuentra la norma internacional de ISO-9000, ISO-9002

Propiedad intelectual (RDL).

Viabilidad económica

Para realizar el Sistema de Información Geográfica Petrolera existen costos relacionados con el sistema estos costos son aproximados.

Costos relacionados con el personal * Proyecto .

1 líder de proyecto	$ 240,000 .00
1 analista	180,000 .00
1 Manejador de base de Datos	120,000.00
3 Desarrolladores	120,000.00
1 Diseñador	120,000.00

Costo total del personal =$ 780,0000.00

Costo de acuerdo a capacitación:

Curso de sistemas de información en Línea	$100,000.00 * Desarrollador
Tecnologías MapXtreme, MapXtend	$100,000.00 * Desarrollador

Costos de Licencias Aproximados.

1 Licencia de MapXtreme 4.5	$ 160,0000.00
3 Licencia de Java Developer 9i	180,000.00
1 Licencia Suite de Oracle	250,000.00
1 Licencia de Oracle Spatial	250,000.00
1 Licencia Mapinfo	120,000.00
1 Licencia de Rational Rose	120,000.00
1 Licencia Apache	0.00
1 Licencia Tomcat	0.00

Costo Hardware:

1 servidor Base de datos (Cluster)	60,000.00
1 servidor de archivos	60,000.00
1 servidor de Aplicación	60,000.00
2 servidores de impresión (Impresora, Ploter)	100,000.00
1 servidor Web	40,000.00
3 estaciones de trabajo	40,000.00

Costo de reclutamiento:

La búsqueda de personal tendrá un costo de:	$ 12,000.00

Costo del espacio físico del proyecto	$ 140,000.00

Otros gastos:	
Conferencias	
Actualizaciones	
Sistemas Ajenos etc...	$150,000.00

El costo total del proyecto es un aproximado.	$ 2,722,000.00

Beneficios

No se realizaran gastos de instalación, el cliente no tiene la necesidad de instalar software especial para poder consultar el sistema, todo se ejecuta por la Web.

Dar soluciones a los organismos subsidiarios de PEMEX, en el manejo de información cartográfica.

Permitir el acceso a la información que se encuentra entorno nacional y regional donde realizan sus operaciones.

Permitir consultar una gran base de datos cartográfica integrada por SICORI a todos los trabajadores de PEMEX.

Sistema que en su etapa inicial tendrá información cartográfica relacionada:

- Suelo/vegetación
- Áreas Protegidas
- Carreteras
- Comunicaciones y transportes
- Conductos y Líneas de transmisión
- GEO-estadística(Estados y Municipios)
- Hidrografía
- Localidades

Esta información será presentada tomando en cuenta las prioridades e intereses de nuestros clientes.

Esto permitirá una mejor toma de decisiones en los diferentes organismos de PEMEX.

Permitirá la integración de grandes volúmenes de información que se encuentra dispersa por todo PEMEX para una representación de forma gráfica y el manejo de la misma de una forma sencilla mediante herramientas de gran capacidad analítica.

3.4 Funciones del sistema

Los siguientes modelos se concentran en lo que debe hacer el sistema no en como lo hace.

Funciones

Permitir el acceso al sistema

Presentación de información de acuerdo con los diferentes roles de usuario

Presentar un mapa inicial de acuerdo a las necesidades y prioridades del cliente

Agregar información cartográfica (Capas de Información)

Control de presentación de las diferentes capas de información

Consultar información, imágenes, videos, tablas, que se encuentren relacionados al punto geográfico.

Resaltar de forma gráfica el punto, polígono, línea, seleccionado.

Búsquedas por nombre de elemento, punto, línea o polígono en las diferentes capas de información del sistema mediante sistemas de búsqueda. Presentando la información encontrada así como poder seleccionarla permitiendo resaltar el elemento encontrado.

Función para que el cliente pueda almacenar su mapa, así como guardar una referencia del mapa en el servidor y después el cliente pueda recuperar su mapa.

Funciones para integrar tablas ajenas y tener un análisis georeferencial de las mismas.

Contraseñas usuarios deben de viajar en la red de forma cifrada.

Cuenta con capacidades de visualización.

Acercamiento: Permite ver mas a detalle el mapa

Alejamiento. Permite ver un mayor número de elementos en el mapa

Paneo: Permite mover el mapa el mapa ala posición señalada

Visualización de todo el mapa.

Retornar el mapa a sus características iniciales.

Cuenta con herramientas:

Medir distancias: Esta herramienta permite ver la distancia que existe de un punto a otro punto y permite ir sumando las distancias.

Longitud Latitud: Esta tiene la función de proporcionar la Longitud/Latitud del punto seleccionado

Región: Tiene la función seleccionar un área de trabajo y aplicar toda la funcionalidad restringiéndolo sobre esa área en particular.

Herramientas de impresión.

Debe tener la funcionalidad de poder etiquetar con diferente fuentes el elemento del mapa ya se por el nombre, fecha o cualquier otro campo que tenga la tabla.

Generación de mapas temáticos esta herramienta permite colorear un mapa de acuerdo a ciertas características que puede presentar la capa por el ejemplo el índice de población que existe en la república.

El sistema debe tener la funcionalidad de poder ir incorporando nuevas funcionalidades ay la posibilidad de trabajar en diferentes clientes Ejemplo de ello celular, Palm, Pocket PC, etc.

Además de que el sistema pretende dejar Componentes para una futura utilización.

Especificaciones

El sistema tendrá un rendimiento óptimo en la Intranet de PEMEX en donde el cliente es un applet, y donde toda la carga la lleva la capa de negocio que es la encargada de distribuir el trabajo en los diferentes servidores llámese motor de mapas, motor de servlets, etc. que realmente esta capa es la encargada de comunicarse con las distintas bases de datos que se encuentran relacionadas con el sistema así mismo también es la encargada de integrar diferentes sistemas y de dar una respuesta al cliente de una forma rápida .

Una de las especificaciones el cliente debe tener instalada la maquina virtual de java la mayoría de los Navegadores ya cuenta con una.

De lado del servidor se debe contar con un web server que tenga un motor para servlets en este caso utilizaremos Tomcat 4.1.2 bajo Apache 1.3.2.

Si el sistema fuera requerido en redes vía MODEM se manejaría un cliente más ligero utilizando tecnología JSP, donde el código realizado es reutilizable incorporándolo al sistema como java beans y etiquetas ya elaboradas y listas para su utilización.

CAPITULO IV

ESTRUCTURA DEL SISTEMA (SIGP)

4.1 Estructura de las tablas

En este capitulo se presentan las tablas con las que trabaja el sistema estas tablas no presentan un Entidad relación. Solo son utilizadas para el mismo control del sistema.

La información que se presenta sobre el mapa es la información georeferenciada esta información es proporcionada por el INEGI, así como la información que tiene PEMEX acerca de los pozos, ductos, petroliferos etc.... También se presenta información ajena siempre que este ligada a un punto geográfico.

Tablas del sistema

MI ARBOL	
PROD CLAVE ...	789
PROD NOMBRE ...	A
PROD DESCRIPCION	A
PROD PRECIO ...	789
PROD UNIDAD ...	A
PROD CVE METADATO ...	A
PROD PADRE ...	789
PROD STATUS ...	789
INEGI	A
QUERY	A
RENDER	A
SERVIDOR	789
GRAFICO	A
TABLA	A
DUENO	A
ZOOM MAX	789
ZOOM MIN	789

Tabla de configuración

Permite obtener información para formar el árbol que se presenta en el sistema .

Esta tabla le permite al sistema crear roles de usuario.

FOTOS	
ID FEATURE	789
NOM CAPA	A
NOM ARCHIVO	A
NOM FOTO	A
FECHA FOTO	31
AUTOR	A
DESCRIPCION	A

Tabla almacenamiento de Fotos

Esta tabla permite almacenar el nombre de las fotos las cuales son relacionadas con el ID para su presentación de las mismas

USER INF DISPONIBLE
DUENO
TABLA

Tabla USER_INF_DISPONIBLE

Esta tabla permite tener roles de usuarios.

SIL BDI
TAB SIL
SDO GID
TAB BDI
CVESE

Vista SIL_BDI

Esta Vista permite la relación con las base de datos de BDI Para presentación de información georeferenciada.

WEB USUARIOS
USUARIO
CONTRASEÑA
ID
PERFIL
NOMBRE
APELLIDO PATERNO
APELLIDO MATERNO
FECHA ALTA
CORREO ELECTRONICO
ESTATUS USUARIO
FICHA

Tabla usuarios

Esta tabla es donde se almacenan los usuarios del sistema

FICHA
NOMBRES
AP PATERNO
AP MATERNO
SEXO
FEC NAC
ORG CLAVE
EMP CLAVE
CEN CLAVE
DEPTO CLAVE
REGIMEN TIT
ESTATUS

Tabla V_SICORI

Esta tabla permite realizar la validación en el sistema

4.2 Diagrama caso de uso

La idea de los diagramas de caso de uso es involucrar a los usuarios en las etapas iniciales del análisis y diseño del sistema . Esto aumenta la probabilidad de que el sistema sea de mayor provecho para la gente a la que supuestamente ayudara, en lugar de ser un manojo de expresiones de computación incompresibles e inmanejables por los usuarios finales.

De acuerdo con el cliente y el analista se determino que existen dos tipos de actores:

Cliente
Administrador.

Funciones del cliente:

En el diagrama se puede observar como el cliente interactúa con el sistema. Primero el cliente realiza una petición al servidor posteriormente el sistema envía un mapa inicial con ciertas características establecidas por el administrador posteriormente el cliente puede realizar funciones como realizar búsquedas de acuerdo a un elemento geográfico, tiene acceso a utilizar herramientas como visualizar el mapa a nivel nacional, centrar el mapa , acercar el mapa , mover mapa, alejar mapa , obtención de la latitud , se le ofrece al cliente un control de capas, herramienta para etiquetar el mapa con los datos de la base , agregar capas , herramienta de impresión.

Funciones del administrador:

El administrador es el encargado de manejar la información geográfica. Una de sus funciones es controlar de donde se va obtener la información, existen distintos tipos de almacenaje de datos dentro de los cuales están

Base Espacial Oracle
Archivos Tipo Raster
Archivos Nativos (.tab)
Imágenes png
Imagenes gif

Otra de sus funciones es la de administrar la información inicial que se le presentara al cliente cuando este realice una petición es decir que tipo de mapa se presentara al cliente.

También es el encargado de establecer los roles de usuario es decir que tipo de información geográfica podrá agregar al mapa inicial .

Observemos como interactúan los actores en el diagrama de caso de uso.

Diagrama

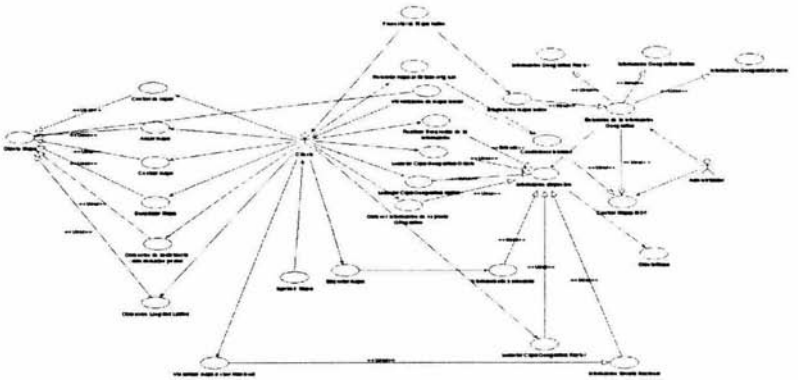


Fig 4.1 Caso de uso del sistema modelado con la herramienta Rational Rose

VER DIAGRAMA ORIGINAL EN LA SECCIÓN ANEXOS

4.2 Diagrama de clases en la capa del cliente.

Para poder representar los atributos, operaciones y relaciones con los que cuenta cada clase dentro del sistema se utiliza la metodología UML en donde para poder representar las clases utilizamos diagramas de clases, los cuales son elaborados por la herramienta Rational Rose.

En el siguiente diagrama podemos observar una panorámica general del sistema del lado del cliente, donde se muestran las relaciones con cada una de las clases.

Dentro del esquema podemos observar la relación que existen con las diferentes clases dentro de las cuales destacan:
Bitácora
BuscarPasivos
Consulta
DialogoBusquedas
DialogInfoPto
DialogoArbol
InfoPto
MCanvas
PanelCapas
PanelPrint
Parámetros
Estas clases son utilizadas por el applet entre otras como por ejemplo clases de swing, awt, sockets, etc..

Mapper Applet

La clase principal del applet es MaperApplet la cual es la encargada de comunicarse con las demás clases que se encuentran operando en la capa del cliente El siguiente diagrama muestra las variables que necesita el sistema y los métodos requeridos para su funcionamiento.
Diagrama de clase en la sección de anexos

MCanvas

Esta clase es la encargada de dibujar el mapa en el cliente y la clase encargada de dibujar sobre el mapa.
Diagrama de clase en la sección de anexos

Parámetros

Esta clase es la encargada de recibir los parámetros del applet, guardando estos datos en variables y proporcionando métodos para la recuperación y almacenaje de las mismas.
Diagrama de clase en la sección de anexos

Productos

Esta clase implementa una interfaz Serializable la cual permite visibilidad de la clase tanto del servidor como en el applet por lo que se pueden guardar valores por ambas capas. Una interfaz serializable se caracteriza por actualizar los valores de esa clase viéndose reflejados por ambas capas.
Diagrama de clase en la sección de anexos

Consulta
Esta clase es la encargada de realizar la comunicación con el servidor el cual regresa una imagen, y objetos que son los encargados de formar el árbol de capas posteriormente. En esta clase se utilizan clases de JAVA como Vector, Hashtable , para la manipulación de las capas. *Diagrama de clase en la sección de anexos*

Arbol

A partir de esta clase se generan objetos los cuales son los encargados de formar el arbol

Esta clase permite iniciar a formar el árbol que se presentara al cliente creando dos vectores conteniendo los índices y los nodos a utilizar, se generan métodos para obtener el padre , recorrer el árbol, crear el árbol , remover nodos, agregar nodos etc..
Diagrama de clase en la sección de anexos

Dialogo Arbol

La clase Dialogo Árbol es la encargada de generar un panel y pintar el árbol también es la encargada de escuchar los eventos del árbol y realizar ciertas operaciones como la de hacer visible al nodo , levantar capas, modificar las capas etc..
Diagrama de clase en la sección de anexos

Nodo

La clase Nodo es la encargada de guardar las características de los nodos del árbol que se obtuvieron de la tabla ARBOL para ello se generan métodos para obtener el valor del nodo así como mandar valores a los nodos como la imagen que va a contener el nombre etc..
Diagrama de clase en la sección de anexos

Producto
La clase Producto actúa como repositorio de los datos, que son obtenidos de la base de datos en este caso de la tabla MI_ARBOL.
Esta clase implementa la interfaz serializable con estó se logra que la clase tenga visibilidad del lado del cliente.
Diagrama de clase en la sección de anexos

Existen mas clases en la capa del cliente pero estas son las principales del lado del cliente las demás actúan como módulos que requieren datos de estas clases.

4.3 Diagramas de clases en el servidor

Existen varias clases en la capa de negocio las cuales son las encargadas de dar repuesta a las peticiones del cliente.

Se muestran diagramas de clases con la metodología UML la cual nos permite realizar una representación de los atributos y las operaciones con las que cuenta cada una de las clases entre ellas se encuentran MapperServlet, ServletProductos, ServletVista, ServletPaneo entre otras que permiten dar respuesta a las peticiones que realiza el cliente.

La clase principal del lado del servidor es MapperServlet la encargada de establecer la comunicación con el servidor de mapas (MapXtreme).
Diagrama de clase en la sección de anexos

MapperServlet

En la clase MapperServlet existen métodos para poder generar un mapa inicial, métodos para convertir la imagen de un formato vectorial a una imagen gif , otro de los métodos a destacar es el de remover las capas nos permite quitar capas al objeto mapa .

En el diagrama de clase podemos observar que utiliza clases de mapinfo el cual es el motor de mapas, también utiliza clases serializadas para guardar información en donde esta información es utilizada por ambas partes.

También esta clase utiliza de clases para guardar la sesión tal es el caso de la clase Puntos así como de la clase para guardar el mapa en una sesión.
Diagrama de clase en la sección de anexos

ServletProductos

Otra de las clases importantes es el ServletProductos esta clase es la encargada de comunicarse con la base de datos y obtener información para crear el árbol en el cliente, para lo cual se necesitara abrir un socket para enviar un vector con información de las capas
Diagrama de clase en la sección de anexos

En la capa de negocio existen clases que tiene que dar respuesta a las herramientas del sistema entre las herramientas del sistema podemos destacar:

Herramientas de medición, acercar el mapa , alejar el mapa , centrar el mapa al punto indicado ,volver el mapa a su estado inicial, herramienta que nos presenta la longitud latitud del punto seleccionado ,así también como herramientas de imprimir herramientas de búsqueda etc..

Para cada una de estas herramientas existe una clase que da respuesta a la petición del cliente empecemos por observar el diagrama de clase de ServletZoomIn .

ServletZoomIn .

La principal funcionalidad de esta clase es la recibir las coordenadas en píxeles y comunicarse con el servidor de mapas para realizar una conversión a coordenadas espaciales para ello se genera un método renderMap el cual va ser el encargado de realizar esta función así como la de enviar el mapa al cliente .
Diagrama de clase en la sección de anexos

ServletDistancia

Esta otra clase es similar a la anterior, ya que también recibe las coordenadas X, Y de dos puntos para su transformación a coordenadas espaciales con el fin de poder calcular la distancia entre dos puntos.
Diagrama de clase en la sección de anexos

ServletModificarCapas

Otra de las clase importantes es la clase ServletModificarCapas la cual debe tener la funcionalidad de agregar capas al sistema , para poder lograr se necesita de otra clase(AgregarCapas)que nos permita elegir entre que base de datos vamos a extraer la información esta puede ser de archivos, de Oracle o imágenes Raster.
Diagrama de clase en la sección de anexos

CAPITULO V

IMPLEMENTACIÓN DEL SISTEMA

Introducción

En este capituló se mostrara la implementación del sistema con JAVA , tanto en la parte del cliente como en la capa de negocio, así mismo como las conexiones a las diferentes bases de datos por medio de drivers como JDBC , así como la implementación de Apache y nuestro Web Server tomcat .

Para este sistema se maneja un cliente mediano es decir , que el cliente para poder utilizar nuestra aplicación necesitara de la maquina virtual de java (JVM).

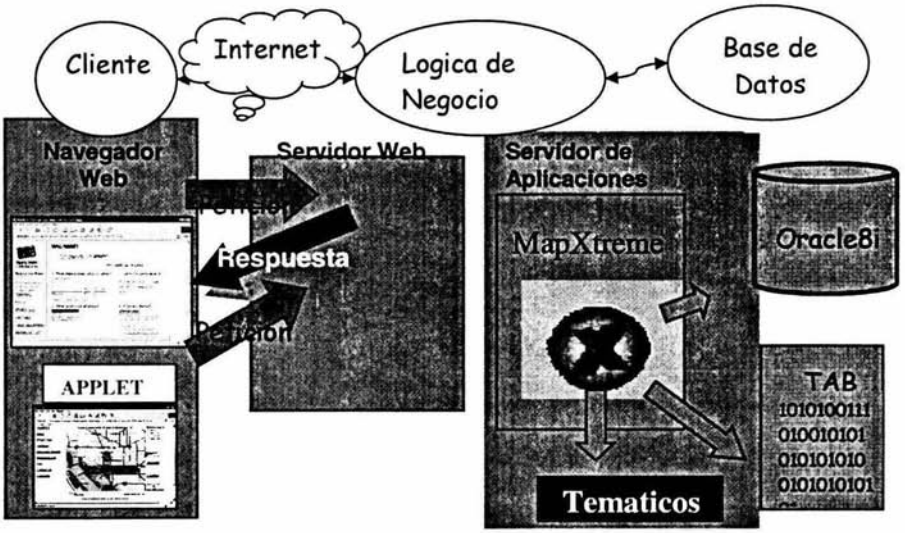


Fig 5.1 Modelo del cliente Mediano

5.1 Implementación de Herramientas Lógica de Negocio

Instalación de Apache y Tomcat

La configuración es sencilla, solo hay que configurar una librería que contiene el programa que sirve de puente entre ambos, mod_webapp.

Lo que tenemos que hacer es copiar estos dos ficheros al directorio:

libapr.dll
mod_webapp.so

APACHE_HOME\modules

APACHE_HOME\conf\httpd.conf buscamos la sección *LoadModule* y añadimos al final las siguientes lineas:
LoadModule webapp_module modules/mod_webapp.so AddModule mod_webapp.c .

Para ello añadimos al final del mismo fichero *httpd.conf*WebAppConnection conexion warp localhost:8008 WebAppDeploy examples conexion /examples

Deberemos arrancar primero Tomcat y después Apache.
Para reiniciar, primero detendremos Apache, después Tomcat, y arrancaremos como antes, primero Tomcat y después Apache.

Lo siguiente es instalar el motor de mapas que en este caso es MapXtreme nos permitirá realizar las consultas a las bases de datos espaciales y poder dar un redering al mapa así como la manipulación de las bases espaciales presentándole información al cliente en forma geográfica más adelante se explicara la funcionalidad de esta API. Sección anexos.

Se necesitan distintas librerías que son :

Drivers (JDBC). Permiten la comunicación con las bases de datos .
MapXtreme 4.0 o Superior. Motor de Mapas
JavaScript : Lenguaje de programación del lado del cliente.
ServletRuntime : Clases de JAVA que permite comunicación con el cliente .

Del lado del cliente será necesario instalar la máquina virtual de JAVA 1.3 o superior, para entender las clases de java en el cliente y objetos que son enviados desde el servidor.

5.2 Módulos de Privilegios

Modulo rol de usuario

Para poder implementar este modulo se requieren de dos tablas especificadas anteriormente

Tabla de Clientes y Productos

Para la asignación de un rol de usuario

Para poder realizar el envió del nombre de usuario y contraseña se abre un socket del lado del cliente para realizar el envió de la información cifrada la cual es procesada por el servidor accesando ala base de datos por medio de un driver que proporciona JAVA JDBC y obteniendo la información al a que puede acceder el usuario es enviada al cliente en un Vector .

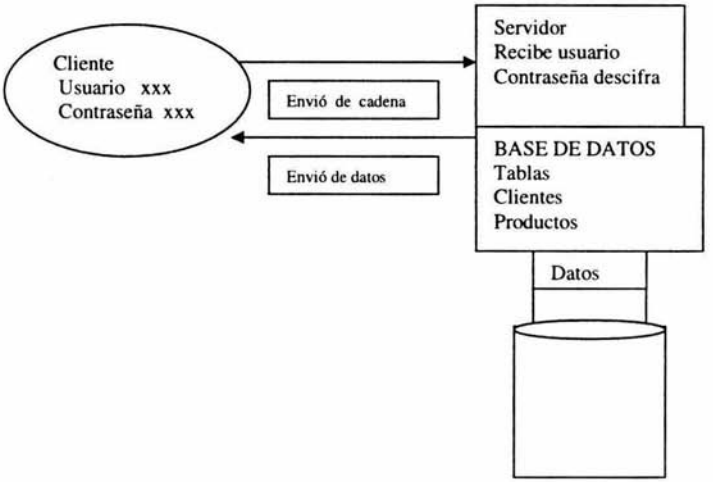


Fig 5.2 Diagrama que permite ver el flujo de la contraseña y usuario

El sistema del lado del cliente necesita realizar una conexión con el servidor para poder enviar el usuario y la contraseña de forma encriptada para poder enviar la información abrimos un socket.

Se reciben los parámetros y se utiliza la función descifrar () para obtener los datos reales y poder realizar la conexión.

Esta función permite conectarse con la base de datos:

public void iniciarConexion

Esta función permite realizar un query a la base de datos y regresarnos un vector con la información del usuario.

public void procesaConsulta(String query)

Posteriormente este objeto es enviado al cliente para su utilización

```
ObjectOutputStreaoos=newObjectOutputStream(newBufferedOutputStream (response.getOutputStream()));
oos.writeObject(this.nombres_Aux);
```

Modulo Árbol de Información

Para poder implementar el árbol se tiene que realizar una petición al servidor para que el servlet realice una conexión con la base de datos en la tabla productos y obtenga los campos para formar el árbol en donde se realiza una clase serializada para guardar los datos de la tabla y sea visible tanto como para los servlets como para las clases del cliente , posteriormente con estos datos se generan dos vectores uno que nos indica el índice donde que ubicación tienen los nodos y el otro vector es quien contiene los datos .

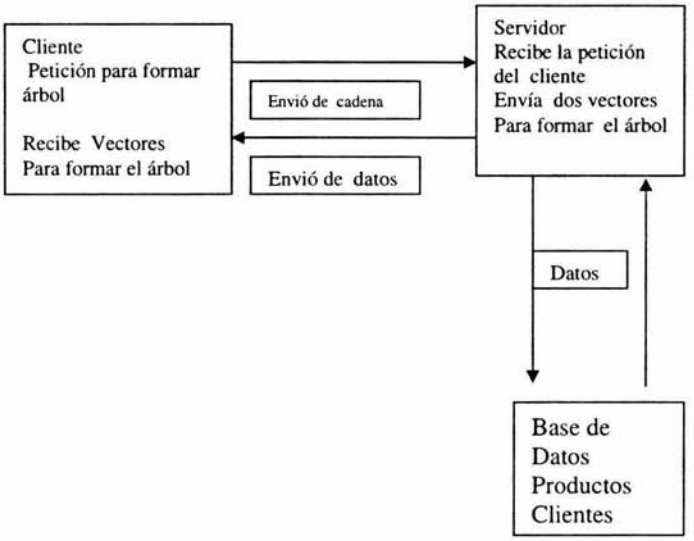


Fig 5.3 Comunicación entre servidor y cliente para formar el arbol de capas.

Esta función permite formar el árbol junto con otras clases. La clase Productos que actúa como repositorio de datos para formar el árbol ,

El servlet que obtiene los datos de la base de datos es Servlet Productos el método a destacar de esta clase es la de procesa consulta

5.3 Módulos de Visualización

Módulo de Mapa Inicial.

Este modulo es el encargado de presentarle información al cliente geográficamente en este caso se le presenta un mapa inicial el cual contiene a la república mexicana con información de los estados e industrias Petroleras.

Para poder presentar información el cliente requiere realizar una petición al servidor en donde el servidor realiza peticiones al servidor de mapas y este se encarga de realizar conexiones con bases de datos espaciales, realizar querys espaciales para la obtención de un mapa georeferenciado el cual es devuelto al cliente.

Dentro del applet existe un método llamado actualiza mapa el cual abre un socket con el servidor para poder enviar una url con parámetros y este poder recibir una session enviada por el servidor y una cadena en bytes para poder formar una imagen del lado del cliente .

En donde esta cadena es enviada a un canvas que es la encargada de su visualizarla.

En la capa de negocio se construye un Servlet el cual es el encargado de recibir la petición del cliente generando una sesión inicial y un mapa.

Para la generación del mapa se debe establecer una comunicación con el servidor de mapas que en este caso utilizamos MapXtreme que es el encargado de ir ala base de dato espacial y generar un mapa inicial el cual es devuelto al cliente.

Guardando en la sesión del cliente el mapa.

Módulo herramienta información

Esta herramienta es la que presenta información del punto seleccionado, además de presentar una imagen relacionada con el punto si contiene el punto, y presentar relaciones con bases de datos externas.

En este módulo el cliente selecciona la herramienta información en donde el applet obtiene el evento del botón , y posteriormente seleccionado un punto dentro del mapa el

applet realiza una consulta al servidor enviando al servidor la coordenadas de la imagen donde las coordenadas están dadas en píxeles para la realización se abre un socket entre cliente y servidor enviado por parámetros de la url las coordenadas X y Y
.

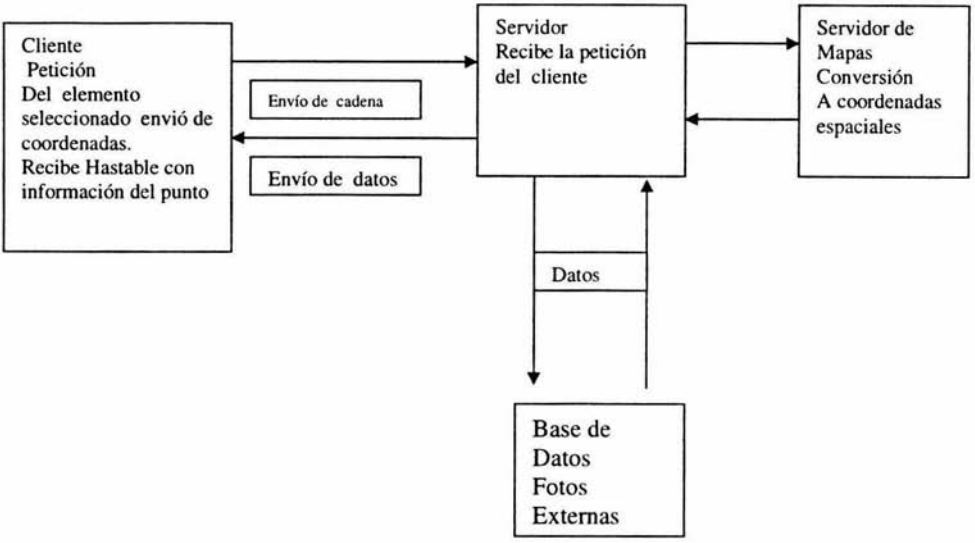


Fig 5.4 Diagrama que presenta el flujo de datos de la herramienta Información.

Esta la función que se encarga de enviar del enlace con el servidor enviando las coordenadas por la URL.

conec(String url) ;

Recibiendo un Hashtable con la información del punto posteriormente se envía la información a la clase Dialog_InfoPto que es la encargada de recorrer el Hastable y presentar el valor de la llave en la ventana posteriormente si selecciona la llave desde la ventana se le presenta a detalle la información del punto , si encuentra relación con alguna foto realiza una nueva conexión con el servidor enviando la foto relacionada al punto.

El servlet encargado de recibir las coordenadas 'X' y 'Y' es ServletInfoPto este se encarga de realizar un enlace con el motor de mapas para poder transformar las coordenadas a coordenadas espaciales donde posteriormente se obtiene la información de las capas y se realiza una búsqueda de acuerdo al punto seleccionado creando un Hastable en donde la llave es el nombre de la capa y la primer columna que en este caso es la columna de nombre presentado como llave y en donde el contenido es las columnas de la capa teniendo como información el resultado de la búsqueda .

El método anterior es el que se encarga de extraer la información de la capa , guardando la información en un Hashtable.

public Hashtable obtInfPunto(DoublePoint punto,MapJ myMap)

Que recibe como parámetros el punto del elemento seleccionado y el objeto Mapa

Este objeto de tipo Hashtable es el que es enviado al cliente, donde posteriormente existe un tratamiento para extraer la información y realizar un enlace con el servidor si es que existe una foto relacionada a la capa y al punto.

Si existe una foto relacionada al punto realiza una nueva petición enviado el ID al servidor , con el cual se realiza una búsqueda en la base datos de la tabla fotos , enviando la foto relacionada del punto al cliente y sus meta datos .

De igual forma se realiza con bases de datos ajenas al sistema, se abre la conexión con estas bases extrayendo su información, enviándola al cliente.

Módulo Búsqueda Sencilla.

En esta herramienta su principal función es la búsqueda de elementos de tipo geográfico para su representación geográficamente estos elementos pueden ser Polígonos, Líneas, Puntos , acercando el elemento a una escala visible dentro del mapa esto se hace seleccionando de una lista la capa de datos a consultar donde posteriormente en una caja de texto se escribe el elemento a buscar esta búsqueda regresa en forma de lista todos los elementos relacionados con la palabra, esto permite seleccionar un elemento donde después podemos visualizarlo geográficamente .

Para poder realizar este módulo existen dos tablas ya mencionadas con anterioridad que son la tabla de Productos de donde se obtiene el nombre de la capa donde posteriormente se realiza una consulta a la base de datos espacial para obtener todos los elementos relacionados con la palabra buscada y posteriormente utilizar el motor de mapas para poder ejecutar un consulta espacial regresándonos el feature(características de un punto) buscado se convierte a una imagen para poder ser enviado al cliente para su visualización .

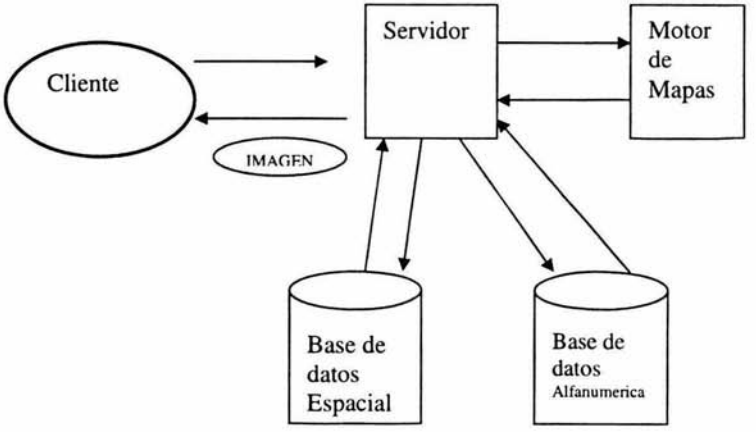


Fig 5.5 Representa el flujo de información que se realiza para poder obtener elemento buscado .

Módulo de Acercar mapa

En este módulo consiste principalmente en poder realizar funciones con el mapa que se le presenta al cliente que en este caso en particular es la herramienta de navegación de acercamiento esta herramienta esta construida tanto del lado del cliente como en el servidor.

En el cliente su principal función es la de realizar un recuadro dentro del área del mapa donde posteriormente se obtienen las coordenadas en píxeles de los dos puntos del recuadro que son los de las esquinas , las cuales son enviados al servidor por medio de la url . Posteriormente se espera recibir respuesta del servidor donde recibimos una cadena de bytes donde son enviados a la función encargada de convertirlos en imagen y presentarla al cliente .

En la capa de negocio su principal función es recibir las coordenadas enviadas en la url para posteriormente ser enviadas al motor de mapas para su transformación a coordenadas espaciales, posteriormente el servidor se encarga de obtener la sesión del mapa que en este caso obtenemos el objeto MapJ para poder trabajar con este objeto y aplicarle la nueva escala de visualización y la generación del mapa guardando en sesión el mapa .El mapa generado es convertido a una cadena de bytes para el envio al cliente .

Motor de mapas es el encargado de extraer los datos de las bases de datos y la manipulacion de la misma con la cual forma el mapa.

En el servidor se encarga de obtener las coordenadas convertirlas y generar el mapa para su envió.

Existen herramientas de navegación como son centrar el mapa en el punto seleccionado, mover el mapa, alejar el mapa , la mayoría de estas herramientas son muy similares en su funcionalidad , lo importante a destacar es la manipulación de la imagen en el cliente .

Su principal característica se vuelve la captura de los eventos del ratón como son moved, relassed , pressed etc.. y capturar sus coordenadas para su envío al servidor donde posteriormente son convertidas a coordenadas espaciales para la manipulación del motor de mapas se ocupa la misma función para el envío de la imagen guardando el objeto MapJ en la sesión cada vez que es modificado

Como se puede observar son herramientas de navegación muy similares con pequeños cambios lo cual permite heredar clases para la funcionalidad del sistema.

5.4 Módulo de Herramientas

En este módulo se realizaron diversas herramientas las cuales hacen del sistema un mejor modelo para el manejo de información cartográfica y cumplir con los requerimientos del cliente.

Herramienta de medición.

Esta herramienta su principal función es la de medir distancias de un punto dentro del mapa a otro punto e ir sumando las distancia entre los diferentes puntos presentando la información en kilómetros .

Las forma en que se realizo la herramienta es la de capturar las coordenadas de los puntos de la imagen que forma la recta enviando el punto inicial y final en pixeles al servidor estos puntos son dados por parte del usuario al trazar la recta automáticamente en el servidor son transformados a coordenadas espaciales que posteriormente son enviadas al motor de mapas el cual realiza una consulta espacial para la obtener la distancia la cual es enviada al cliente , donde la presenta en una ventana nueva .

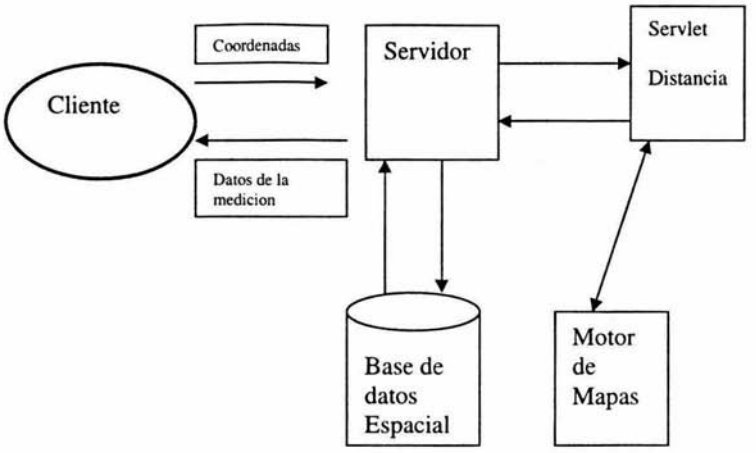


Fig 5.6 Diagrama que presenta el flujo e información para obtener la distancia entre dos puntos

Herramienta Etiquetar mapa.

Esta herramienta es una de las herramientas mas importantes para obtener información acerca del mapa temático que estamos consultando, con esta herramienta existe la posibilidad de poder etiquetar capas de información geográfica indicando la columna por la cual se quiere etiquetar Nombre, ID, Fcode, INEGI, etc... Tiene funciones como cambiar el tamaño de la letra, el color, además tiene función de la etiqueta ,siga la curva, polígono, recta .

Esta herramienta nos da una mayor ubicación en donde esta georeferenciado el mapa donde podemos agregar ciudades, industrias petroleras , vegetación , lo cual nos puede indicar en que lugares existen zonas de riesgo etc.

Esta herramienta se construyó tanto en el cliente como en la capa de negocio interactuando con las bases de datos, en el cliente se creó una caja de dialogó la cual es visible después de seleccionar la capa etiquetar esta capa se elige en el caja de dialogo de control de capas , posteriormente elegimos la columna por la cual se va etiquetar , en donde para poder presentar las columnas por la cuáles se quieren etiquetar se tuvo que realizar una petición al servidor el cual es el encargado de obtener las columnas de la capa y ser enviadas de vuelta al cliente, posteriormente de seleccionar se elige el tipo de letra y color de la misma , en una caja de verificación se pide que el etiquetado sea visible, y seleccionamos aceptar en este momento se realiza una petición al servidor para que no se devuelto un mapa con etiquetas, donde el servidor se encarga de interactuar con las clases que permiten etiquetar el mapa.

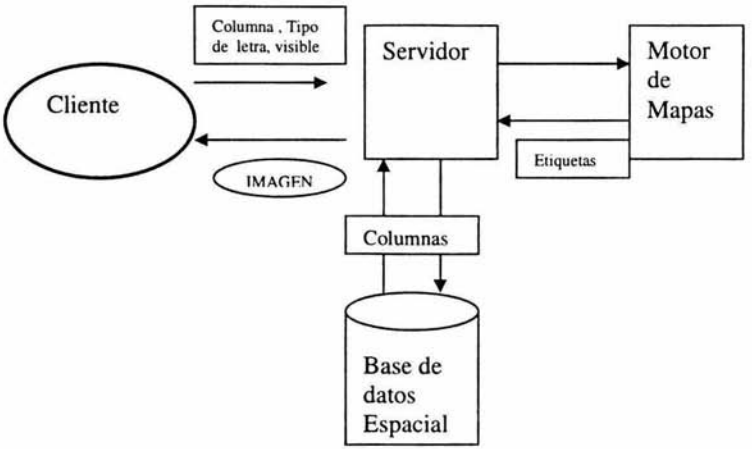


Fig 5.7 Flujo de información para generar etiquetas que representan las características del elemento.

Herramienta Impresión.

En esta herramienta su principal función es la de imprimir el mapa aquí toda la operación lo realiza la maquina del usuario esta herramienta fue construida dentro del applet dentro de la cual se implementaron clases ya hechas para imprimir, estas solo se importaron y se crearon objetos de estas clases en donde para utilizarlas se extrajo la imagen del canvas y se agregó a un panel dentro de la cual puede ser visualizada en este momento se puede mandar a imprimir en la maquina del cliente o se puede dar imprimir pantalla el cual se puede utilizar para poder visualizar la imagen con algún visualizador , y poder agregarle características especiales que el cliente deseara poner slogan , autor, fecha de creación, titulo, fecha de creación sobre el mapa.

Se pretende posteriormente implementar la impresión con formato vectorial el cual le permitiría a la imagen cambiar de escalas sin perder resolución , para lo cual se piensa utilizar el formato estándar por la open GIS (GML) para poder tener la imagen en un formato XML y después pasarla a un formato Scale Vector Graphics (SVG), utilizando plantillas donde posteriormente esta imagen podría ser mandada a imprimir en diferentes escalas, en determinados ploters de alta resolución .

Herramienta Agregar Capas.

Es una de las herramientas importantes del sistema debido a su funcionalidad el cual permite agregar mas capas de información esto mediante un control de forma de árbol que fue implementado en la primer etapa , seleccionado las capas de interés en las cuales esta organizadas en diferentes niveles de acuerdo ala información que la capa presente , por ejemplo Sociopolítica dentro de la cual tiene capas contenidas como Estados, Localidades, Municipios. La forma te traer la imagen es eligiendo el botón aceptar después haber seleccionado las capas que se decidieron consultar, este control tiene también la funcionalidad e presentar la información a diferentes acercamientos.

Como funciona

Depuse de haber implementado el árbol que se explicó en la primera parte de este capituló se seleccionan las capas a consultar indicando su selección con un gif de la mismas características de la capa en el cual en este momento el trabajo de operación lo lleva el cliente formando un vector con las capas seleccionadas indicando las características de la capa en donde existe una clase la cual tiene métodos que nos indica si ya fue seleccionada la capa con esta información o si todavía no ha sido levantada . este vector con las características de cada capa es enviada al servidor por medio de un socket el cual se abre en el applet recibiendo como respuesta una cadena de bytes .

En el servidor la clase encargada de recibir este vector es el Servlet ModificarCapas la cual se encarga de mandar el vector a la clase Agregar capas esta clase por su parte se encarga de obtener las características de la capa las cuales están contenidas dentro una clase Llamada Nodo la cual nos indica el tipo de información , de esta clase extraemos la información llámese desde una base de datos Oracle, Archivos tab , Imágenes de tipo Raster etc.. de esta clase también obtendremos si la capa ya fue levantada ahorrándonos el tiempo de volverla a cargar , en esta clase también se guarda el padre de la capa , el tipo de grafico, los índices del nodo, ya obteniendo las características de la capa la clase AgregarCapas se encarga de agregar las capas realizando la comunicación con el motor de mapas MapXtreme el encargado de interpretar bases de datos de datos espaciales para la extracción de información, si todavía no se levantara la capa o haciéndola visible si ya fue levantada con anterioridad toda esta información es posteriormente aplicada sobre el objeto MapJ , donde posteriormente el objeto mapa es guardada en la sesión del cliente para su futura utilización después la clase Modificar capas es la encargada de convertir el objeto mapa a una cadena de bytes entendida por el cliente , esta es envida al cliente en donde este se encarga de representarla gráficamente con la clase Canvas métodos propios de JAVA.

Este método es el encargado de levantar la capa .

public void levantaCapa()

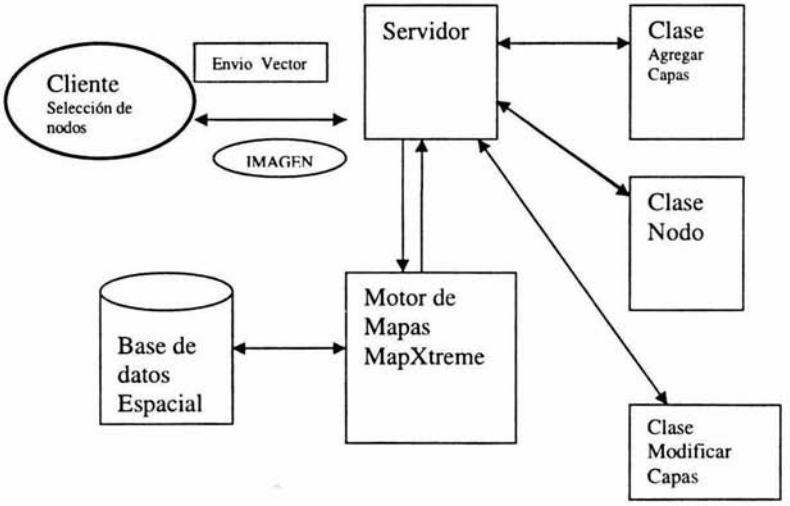


Fig 5.8 Flujo de información para levantar una capa

Herramienta control de capas.

La función de esta herramienta es la de organizar las capas en su forma de visualizarse es decir que posición va ocupar en el mapa.

Al cliente se le presenta un panel en el cual se le presentan las capas levantadas y el o orden en el que se le presentan posteriormente el usuario puede seleccionar que capas van hacer movidas y la posición que van ocupar posteriormente ya establecido el orden de las capas se selecciona aceptar .

Las capas se agregan a un vector indicando la posición que se encuentran guardando en una lista el índice del vector moviendo los índices, posteriormente este vector se envía al servidor para se recorrido y ver la posición que ocupan las capas , llamando al Motor de mapas para organizar las capas , guardando la información en el objeto mapa enviando la imagen de vuelta al cliente .

Ésta es la función de organizar el orden de las capas de acuerdo al vector enviado por el cliente en donde lleva el contenido de las capas.

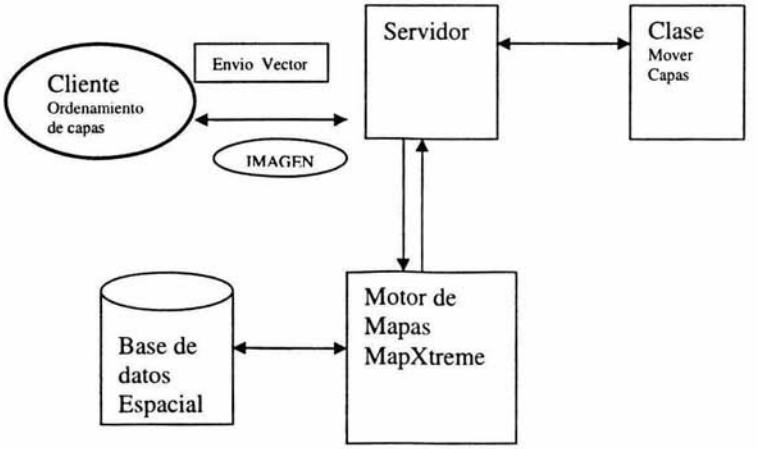


Fig 5.9 Diagrama que muestra el flujo de información para el control de las capas

Herramienta Latitud Longitud

Esta herramienta es la encargada de proporcionar información de la latitud y longitud del punto seleccionado dentro del mapa presentándola en cajas de texto dentro del applet.

Esta herramienta captura las coordenadas "X" y "Y" del punto seleccionado por medio del evento mouse pressed de JAVA las cuales estan dadas en píxeles , son enviadas al Web Server en donde se establece comunicación con MapXtreme para la conversión de las coordenadas a espaciales se obtiene su dato numérico y es enviado al cliente el cual es presentado por el applet.

Herramienta Mapa Escala.

Su principal función de esta herramienta es la de mostrar el mapa a un nivel mas alejado del real permitiendo al cliente poder visualizar mas fácilmente su ubicación dentro del mapa.

En la presentación de este mapa no se cargan capas de información solo tiene la capa principal que en este caso es la de Estados.

Para poder ver esta imagen dentro del applet se genera un nuevo objeto de la clase canvas generando un nuevo panel en el cual se va presentar a imagen para esto se realiza una petición al servidor para la generación de un nuevo objeto de tipo mapa cual no puede contener capas excepto la principal , no puede ser seleccionable tampoco puede ser editado este mapa tiene las mismas características que el objeto del mapa principal es decir también se le genera una sesión y se guarda en cada cambio que se le realiza al mapa principal , en donde el servidor realiza una petición al motor de servlets para poder presentar este mapa a un diferente nivel de acercamiento posteriormente se genera la imagen la cual es enviada al cliente en forma de bytes y reconstruida por la clase Canvas, la imagen es actualizada cada vez que se cambia el nivel de acercamiento repintado la imagen del lado del applet.

5.5 Configuraciones Iniciales del Sistema

Existen configuraciones en el sistema que nos permiten enviarle parámetros al sistema tanto al applet como a los servlets , en el lado del cliente se pasan parámetros por medio del html los cuales son recibidos por el applet para su correcto funcionamiento, dentro de estos parámetros le indicamos el usuario que se encuentra trabajando la ip del cliente, que nos sirven para generar bitácoras, también recibimos como parámetros la contraseña del cliente encriptada la cual nos sirve para ver el tipo de rol que tiene el usuario, así también el sistema cuenta con un parámetro en donde indicamos donde se encuentran ubicadas las imágenes, también indicamos las url que se necesita para establecer una conexión con el servidor, a la vez es enviado un parámetro en el cual indicamos la tabla la cual tiene información de todas las capas disponibles a cargar, además utilizamos un parámetro el cual nos permite cargar un mapa inicial .

A continuación se presenta una tabla en la cual se encuentran todos los parámetros recibidos por el applet.

La configuración de las rutas se establece en el html.

NOMBRE DE LA VARIABLE	**VALOR**	**DESCRIPCIÓN**
CODE	MapperApplet.class	Nombre de la clase principal
CODEBASE	http://141.21.1.30:8080/mapxtreme45/SIGL_PEMEX	Ubicación de la clase principal del applet
servleturl	http://141.21.1.30:8080/mapxtreme45/servlet/MapperServlet	Ubicación de la url del servlet principal
imURL	http://141.21.1.30:8080/mapxtreme45/SIGL_PEMEX/images/	Ubicación de las imágenes del árbol
MAYSCRIPT	true	Indica la activación de clases javaScript dentro del applet
nombre	pruebas	Indica el nombre del usuario
IP Cliente	141.21.1.30	IP del cliente
localHost	http://141.21.1.30:8080/mapxtreme45/	Ubicación de la carpeta de mapxtreme
U	erhrgz	Nombre del usuario encriptado
P	xqsw7472	Contraseña del usuario encriptada
mdf	C:\MapInfo\tomcat-4.0.1\mxj450\webapps\mapxtreme45\SIGL_PEMEX\mexico2.mdf"	Ubicación del mdf

tabla	MI_ARBOL	Indica la tabla de configuración del arbol
botonRegion	false	Indica si se requiere utilizar la herramienta region
imagenHerra	images/icons_herramientas/	Indica la ubicación de las Imágenes De herramientas
imagen	images/	Indica la ubicación de las Imágenes

En la capa de negocio también se establecen configuraciones iniciales dentro de los cuales están pasar parámetros que nos indican a que tipo de repositorio vamos a ir por la información espacial , en le caso de PEMEX existen varios repositorios tales como, archivos tab. Tablas en ORACLE, imágenes georeferenciadas como son imágenes Raster, sombreados en 3D, a lo que llamamos curvas de nivel , también indicamos donde se encuentra ubicado el servidor de mapas el cual no permite realizar querys a la base datos, también indicamos donde se encuentran la fotos almacenadas de los diferentes instalaciones de PEMEX .

Estas configuraciones se realizan en el Web Server dentro de un archivo llamado web.xml el cual tiene tanto IAS como tomcat , en donde estos parámetros son leídos por los servlets dentro del método INIT de cada servlet .

Rutas en el servidor

En el web.xml se configuran los parámetros iniciales para la ejecución de los servlets

MapperServlet:

NOMBRE DE LA VARIABLE	VALOR	DESCRIPCIÓN
filetoload	C:\MapInfo\tomcat-4.0.1\mxj450\webapps\mapxtreme45\SIGL_PEMEX\mexico2.mdf	Esta variable permite cargar el mdf inicial
mapxtremeurl	http://localhost:8080/mapxtreme45/servlet/mapxtreme	Indica la ubicación del motor de mapas

ServletInfoPto:

NOMBRE DE LA VARIABLE	VALOR	DESCRIPCIÓN
url	http://141.21.1.255:8080/mapxtreme45/servlet/	Indica la ubicación del repositorio de los servlets
urlFotos	http://141.21.1.55/ServLinea/Fotos/	Indica la ubicación de las fotos

Existen diversos repositorios para la ubicación de las bases de datos, archivos ,imágenes Raster.

Que igualmente se configuran en el archivo web.xml

Clase AgregarCapas

NOMBRE DE LA VARIABLE	VALOR	DESCRIPCIÓN
ubicacionTab	C:/tabs	Indica la ubicación de archivos tabs
ubicacionRaster	C:/raster	Indica la ubicación de las imágenes raster
ubicacionVilla	141.70.1.47	Indica la ip donde se encuentra la base de datos
ubicacionMexico	141.21.1.64	Indica la ip donde se encuentra la base de datos

Conclusión

Un SIG se puede usar para aplicaciones generales incluyendo publicación de información en INTERNET y están creciendo en la actualidad en la medida que la cultura de la información geográfica va permeando a la gente no especializada en el área que la mayoría de las empresas requiere un SIG este crecimiento se ha visto incrementado por la presencia de software libre que en algunos casos ya son un estándar caso (OPEN GIS) también ha ayudado a este crecimiento el hecho de que no se requieren equipos de computó especiales para poder operar (SIG) .

Ahora los SIG están actuando como integradores de aplicaciones y se están aprovechando las tecnologías como las comunicaciones, energía, petroleras, etc.. con el fin de georeferenciar sus aplicaciones, esto se logra llamando métodos de diferentes sistemas creados con diferentes herramientas para poder trabajar conjuntamente y crear un sistema más sólido utilizando tecnologías como (GML, XML ,CORBA, JAVA, SVG,WEB SERVICES).

ANEXO 1 MapXtreme Java

- Mejoras a MapXtremeServlet
- Protocolo XML MapInfo Enterprise
- Recursos Nombrados
- Enterprise Manager
- Librería para JSP
- Mejoras a *Renditions*
- Mejoras al Etiquetado
- Servlet Forwarding
- *Rendering* Progresivo
- DataBinding
- DataProvider API
- Utilerías Geográficas
- Soporte Para Imágenes Raster y Grids
- WBMP
- Opciones de Logging

MapXtreme Java 4.0 es una herramienta de desarrollo de nivel empresarial basada en la arquitectura de servlets especificada en Java 2 Enterprise Edition.

Requiere de una configuración web server/Servlet Container que soporte servlets (J2EE 1.2 Servlet 2.2).

Entre las novedades de esta versión está el soporte de JavaServer Pages

- Las aplicaciones y los datos son cuidados y manejados en forma centralizada
- Las aplicaciones son fácilmente accesadas y modificadas
- Multi plataformas
- Altamente escalable

MapXtreme Java 4.0 es una herramienta de desarrollo de nivel empresarial basada en la arquitectura de servlets especificada en Java 2 Enterprise Edition.

Requiere de una configuración web server/Servlet Container que soporte servlets (J2EE 1.2 Servlet 2.2).

Entre las novedades de esta versión está el soporte de JavaServer Pages

- Las aplicaciones y los datos son cuidados y manejados en forma centralizada
- Las aplicaciones son fácilmente accesadas y modificadas
- Multi plataformas
- Altamente escalable

MapXtremeServlet

- Engine de mapeo del lado del servidor, implementado como un servlet.
- Adopta las iniciativas de Java 2 Enterprise Edition
- El servlet container administra el balanceo de carga, el threading, la tolerancia a fallas
- MapXtreme Java 4 implementa el soporte a J2EE usando JSP y servlet forwarding.
  - Incluye una librería de tags JSP
  - El servlet forwarding ofrece una manera opcional más directa para hacer llegar una imagen al cliente

Protocolo XML MapInfo Enterprise

- Define como se deben realizar las solicitudes y como se realizan las respuestas hacia productos MapInfo, como MapXtreme Java, MapMarker J Server y Routing J Server

- Se incluyen los DTDs que especifican las reglas para los documentos XML. Estos DTDs se localizan en el archivo midtds40.jar en /mapxtremejava/server
- Formatos
    - TAB
    - Oracle 8i Spatial
    - Oracle 7 & 8 Con Spatialware
    - Informix Universal Server con SpatialWare
    - DB2 con SpatialWare
    - JDBC XY
    - ESRI Shape

NUEVO EN MAPXTREME JAVA 4

- GeoTIFF y MIGRID
- Data Binding
- API para DataProviders

NUEVO EN MAPXTREME JAVA 4

- Mejoras a los Renditions
    - Renditions de etiqueta por figura
    - Style Chooser Dialog
    - Renditions nombradas (JNDI)
    - Etiquetas escalables
- Mejoras al Etiquetado
    - Etiquetado temático
    - Etiquetado por figura
    - Nuevas propiedades
        - *Expresiones*
        - *Multilínea*

NUEVO EN MAPXTREME JAVA 4

- *Servlet Forwarding*
    - *IntraServletContainerRenderer.*
- *Rendering Compuesto*
    - *Permite especificar cuales capas deben redibujarse cuando el mapa se actualiza*
- *Rendering Progresivo*
    - *Envía imágenes parciales del mapa al cliente, seguidas de imágenes completas después de cierto periodo de tiempo*
- *Enterprise Manager*
- Genera imágenes
- Multi-threaded
- La ejecución de los Threads es "inteligente"
    - Maximiza el paralelismo (minimizando el bloqueo)
    - Minimiza el tiempo transcurrido por petición
- Sin estado
- Asíncrono
- Actualización progresiva de imagenes
- Provee una interfase al estilo de MapX
- Administra el estado
- Se comunica con el renderer y dataproviders
- Es delgado

REQUERIMIENTOS – SOFTWARE

- Un servidor Web
    - Netscape, Apache, Microsoft IIS
- MapXtreme Java
- Un Servidor de aplicaciones
    - Servlet Container, HAHTSite, OAS, etc
- Java 2
    - JDK 1.3.1 o superior

REQUERIMIENTOS DEL SERVIDOR

- Tarjeta de video en el servidor
- 50 MB de disco duro disponible para la instalación
- 250 MB de disco duro para datos
- 64 MB de memoria RAM disponible para MapXtreme Java

HABILIDADES REQUERIDAS

- Programación en Java
- Desarrollo en Web
  - HTML
  - JavaScript
  - XML
  - JSP
  - JNDI
- Alguna experiencia con MapInfo
- En Internet predominan las arquitecturas en tres capas
- Capas
  - Cliente
    - Interface de usuario
  - Middleware
    - Presentación Web
    - Lógica del negocio
    - Servicio
  - Datos
    - Administración de los datos
- Ligero
  - Baja dinámicamente
  - HTML, XHTML / JavaScript, XML,Java Applet, etc.

- Plataformas para el cliente
  - Una simple aplicación web puede soportar clientes de múltiples plataformas
  - La plataforma del cliente depende del dispositivo
    - Web browser
    - Dispositivo móvil, etc.
- Compuesta por tres sub capas
  - Presentación de la web, lógica del negocio, servicios

- Para ambientes Java la capa de en medio corre dentro de un servidor de aplicaciones
  - Los servidores de aplicaciones constan de un servlet container y, opcionalmente de un Enterprise Java Beans (EJB) container.
- Presentación Web
  - Se apoya en Java Server Pages (JSP), etc.
  - Para aplicaciones web que soportan múltiples clientes, regresan el "mejor" formato para un cliente específico
  - Los JSPs se compilan en servlets y requieren un servlet container
- Lógica del Negocio
  - Pueden implementarse como servlets y/o EJBs
- Servicios
  - Librerías/módulos de terceros invocadas por la lógica del negocio
- Administración de los datos
  - Almacenamiento
  - Validación
  - Acceso
- Acceso a datos Los datos de la empresa están siempre disponibles y actualizados

  MapJ y el MapXtremeServlet pueden "vivir" en el mismo contenedor de servlets y espacio de proceso
- No es necesario Java en el cliente (si se despachan páginas de html puro)

- Mejor capacitado para publicaciones en internet

A continuación revisaremos:

- Vista rápida del modelo del objeto
- Inicializando
- Rendering
- Creando el primer mapa
- Control de Visualización

- MapJ mantiene el estado del mapa

- capas en el mapa
- sistema de coordenadas
- unidades de distancia
- limites del mapa (bounds)

- Instanciar un objeto MapJ

- MapJ myMap = new MapJ();

- Cargar un archivo de geoset, o.....

- myMap.loadgeoset(geosetname, dataDir);

- El nombre del geoset es la ubicación completa de este

"c:\\mapxtreme\\maps\\world.gst"
dataDir es la ubicación de los archivos TAB referenciados en el geoset

- .....Crear un MapDefContainer y cargar una definición de mapa (MapDefinition)
- Un MapDefContainer contiene definiciones de mapas basados en XML

Un MapDefContainer es un directorio
MapDefContainer mdc = new FileMapDefContainer(dir);
- Un JDBCMapDefContainer es una tabla en una RDBMS
MapDefContainer mdc = new JDBCMapDefContainer (driver, url, user, password);

- Charger una definicion de mapa

- myMap.loadMapDefinition(mapDefContainer, name);
Name es la designación del MapDefinition
- Es preferible usar MapDefinitions que geosets
  Reconocen todo tipo de DataProviders
  Pueden perderse características al ir de MapDefinition a un geoset

El tamaño de de la imagen del mapa puede ser modificada con MapJ.setDeviceBounds()
-myMap.setDeviceBounds(new DoubleRect(0,0,800,600));
Hay que establecer los límites antes de dibujar
El tamaño de la imagen por default es 640x480

Instanciar un objeto renderer
```
MapXtremeImageRenderer renderer = new
  MapXtremeImageRenderer (mapxtremeServletUrl);
```
Asignar este objeto al objeto de MapJ
```
    myMap.render(renderer);
```
Generar el mapa en un archivo
```
    renderer.toFile("myMap.gif");
```
Liberar el renderer
```
     renderer.dispose();
```

- Instanciar un objeto MapJ
- Usar MapJ para cargar la definición de mapa o el geoset
- Establecer las dimensiones del mapa
- Instanciar un renderer
- Asignar el renderer a MapJ
- Generar el mapa
- Llamar dispose() en el renderer

MapJ tiene metodos para el control de visualización:
Pan: setCenter()
```
myMap.setCenter(worldpoint);
```

Zoom: set Zoom()
```
myMap.setZoom(myMap.getZoom() *2.0);
```

Pan y Zoom - setZoomAndCenter()
mymap.setZoomAndCenter(myMap.getZoom() /2.0 , worldpoint);

Algunos metodos de MapJ como setCenter dependen de la localizacion de un punto
Algunas veces el punto es el resultado de el click del usuario en una posición especifica del mapa
El punto donde el usuario hizo click típicamente se expresa en pixeles
Los métodos de MapJ necesitan el punto en el mapa en "coordenadas geográficas"

- MapJ.transformScreenToNumeric(), convierte el punto a coordenadas de mapa
- Este método requiere de un objeto com.mapinfo.util.DoublePoint

DoublePoint es un punto definido con coordenadas de doble precisión (x,y)

Ejemplo
//Crear el punto en la pantalla
screenpoint = new DoublePoint(event.getX(), event.getY());
//Crear un punto al mundo real
worldpoint = myMap.transformScreenToNumeric (screenpoint);

- Añadir/Remover/insertar
  - Se agregan capas usando el método "add()" de la colección layers

myLayerSet.add(dpRef,MyDeskhelper);
- Se quitan capas usando el método "remove()" de la colección layers
myMap.getLayers().remove("myLayer") ;
- Insert nos permite especificar la ubicación en que se insertara la capa
myLayerSet.insert(dpRef,myDesHelper,0);

- Una capa es una colección de figuras y sus atributos
- Una capa tiene métodos que permiten consultar la información asociada con las capas
  searchAll; searchAtPoint; searchByAttribute;
  searchByPrimaryKey; searchWithinRadius;
  searchWithinRectangle; searchWithinRegion
- Una vez que una capa es creada, esta puede ser tratada genéricamente
- La información que define la capa se especifica cuando se crea, despues de eso queda oculta

- Se requiere crear un objeto que implemente cada unas de las siguientes interfaces
  - DataProviderHelper
  - TableDescHelper
  - DataProviderRef
- Define la fuente de los datos asociados
  - Para Capas que estén siendo definidas en términos de datos almacenados en un formato de archivo, el "data source" es solo el directorio que contiene el archivo
    Ejemplo"/opt/mapxtremejava/maps"
- Para información proveniente de una RDBMS el DataProviderHelper corresponde a la información requerida para hacer una conexión JDBC a el DataSource

Describe la información específica contenida dentro del "data source" que debe definir la capa
- Para formato de datos basado en archivos este es solo un nombre de archivo. Ej."world.tab"
- Para datos provenientes de RDBMS es usualmente un nombre de una tabla dentro de la base de datos Ej "username.world"
- Las capas que están basadas sobre datos RDMBS pueden ser definidos también con una consulta SQL
Ej. "SELECT c.Country, c.GEOLOGIC FROM Counties c, States s WHERE s.State = "New HampShire" AND MDSYS.SDO_RELATE (c.GEOLOGIC,'mask=INSIDE+COVEREDBY') = 'TRUE' "

Describe quien es responsable para obtener datos del "Data Source"
Concepto similar al modelo Images, ImageConsumer, ImageProducer de Java

- Con un "LocalDataProviderRef" la capa (su dataprovider interno) es responsable de accesar directamente los datos del DataSource.

- Los recursos necesarios para accesar los datos deben estar disponibles para el proceso en el cual MapJ y la capa están corriendo
- Cuando se accesa una RDBMS con una capa creada con una LocalDataProviderRef el controlador JDBC debe estar en la classpath del proceso que ejecuta a MapJ

- Las Capas se refieren a MapXtremeServlet para conseguir los datos
- La capa solo contendra un stub Dataprovider mientras el MapXtremeServlet realizara el trabajo real y trasmitira los resultados a la capa
- Crear el TableDescHelper
- Crear el DataProviderHelper
- Crear el DataProviderRef que requiere el DPHelper como entrada
- Usar el metodo de capas add el cual toma al DataProviderRef y al TableDescHelper como entrada

MapXtreme Java 4 permite combinar la información gráfica contenida en una tabla MapInfo (.Tab) con información obtenida de una fuente JDBC
Para este propósito se utiliza el objeto DataBindingTableDescHelper

- Crear un TableDescHelper para cada tabla
- Crear un DataBindingTableDescHelper. El contructor toma como parámetros los dos DescHelper del paso anterior. Además, indicar los nombres de los campos a utilizar para el databinding.
- Crear un DataProviderHelper para cada tabla
- Crear un DataBindingDataProviderHelper, que toma como parámetro de entrada los DPH del paso anterior
- Crear un DataProviderRef pasando como parámetro el DataBindingDataProviderHelper
- Agregar la capa a la colección de capas

Las capas para anotaciones permiten agregar nuevos datos
- Las figuras pueden ser usadas para marcar ciertas áreas del mapa
- Se crea por el AnnotationTableDescHelper y AnnotationDataProvider
- Una vez creada es tratada como cualquier otra capa
- La capa de anotación reside en la memoria en el espacio del proceso de MapJ
- Siempre usa un LocalDataProviderRef ya que MapJ debe "producir" los datos

```
AnnotationTableDescHelper tableHelper =
New AnnotationTableDescHelper(layer name) ;
AnnotationDataProviderHelper dpHelper =
New AnnotationDataProviderHelper ( );
LocalDataProviderRef dpRef
New LocalProviderRef (dpHelper) ;
Mapj.getLayers( ) .add(dpRef , tableHelper,
"myAnnotations");
```

- Dado que la capa Annotation es una capa en memoria se requiere muy poca informacion para describirla

- Por default, las capas Annotations no tienen estructura asociada
- Puede usarse un TableInfoImpl para especificar una estructura de tablas para la capa Annotations
- Se necesita un objeto TableInfoImpl para crear temas que necesitan un nombre de columna o para etiquetar por nombre de columna
- Crear el objeto TableInfoImpl
- Usar el constructor de capas Annotations que toma el TableInfoImpl como un parámetro

```
String colNames []= newString [2]
colNames[0] = new String ("CityName");
colNames[1] = new String ("Population")
```

```
Int colTypes[] = new int[2];
colTypes[0] = TableInfo.COLUMN_TYOE_STRING;
colTypes[1] = TableInfo.COLUMN_TYPE_INT;
Int pkCol[] = new Int(1);
pkCol[0]=0;
TableInfoImpl til = new TableInfoImpl
("Annotations",myMap.GetDisplayCoordSys(), colTypes,1Pkcol,false);

AnotationsDataproviderHelper tab Helper =  new AnotationsDataproviderHelper(til);
AnnotationTableDeschelper  tableHelper =
                                        new AnnotationTableDescHelper("Annotations")
```

- Crear el TableDescHelper
- Crear el DataProviderHelper
- Crear el DataProviderRef que requiere el DPHelper como entrada
- Usar el método de capas add el cual toma al DataProviderRef y al TableDescHelper como entrada

MapXtreme Java 4 permite combinar la información gráfica contenida en una tabla MapInfo (.Tab) con información obtenida de una fuente JDBC
Para este propósito se utiliza el objeto DataBindingTableDescHelper

- Crear un TableDescHelper para cada tabla
- Crear un DataBindingTableDescHelper. El constructor toma como parámetros los dos DescHelper del paso anterior. Además, indicar los nombres de los campos a utilizar para el databinding.
- Crear un DataProviderHelper para cada tabla
- Crear un DataBindingDataProviderHelper, que toma como parámetro de entrada los DPH del paso anterior
- Crear un DataProviderRef pasando como parámetro el DataBindingDataProviderHelper
- Agregar la capa a la colección de capas

Las capas para anotaciones permiten agregar nuevos datos:

- Las figuras pueden ser usadas para marcar ciertas áreas del mapa
- Se crea por el AnnotationTableDescHelper y AnnotationDataProvider
- Una vez creada es tratada como cualquier otra capa
- La capa de anotación reside en la memoria en el espacio del proceso de MapJ
- Siempre usa un LocalDataProviderRef ya que MapJ debe "producir" los datos

```
AnnotationTableDescHelper tableHelper =
New AnnotationTableDescHelper(layer name) ;
AnnotationDataProviderHelper dpHelper =
New AnnotationDataProviderHelper ( );
LocalDataProviderRef dpRef
New LocalProviderRef (dpHelper) ;
Mapj.getLayers( ) .add(dpRef , tableHelper,
"myAnnotations");
```

- Dado que la capa Annotation es una capa en memoria se requiere muy poca información para describirla

- Por default, las capas Annotations no tienen estructura asociada
- Puede usarse un TableInfoImpl para especificar una estructura de tablas para la capa Annotations
- Se necesita un objeto TableInfoImpl para crear temas que necesitan un nombre de columna o para etiquetar por nombre de columna

- Crear el objeto TableInfoImpl
- Usar el constructor de capas Annotations que toma el TableInfoImpl como un parámetro

```
String colNames []= newString [2]
colNames[0] = new String ("CityName");
colNames[1] = new String ("Population")
Int colTypes[] = new int[2];
colTypes[0] = TableInfo.COLUMN_TYOE_STRING;
colTypes[1] = TableInfo.COLUMN_TYPE_INT;
Int pkCol[] = new Int(1);
pkCol[0]=0;
TableInfoImpl til = new TableInfoImpl
("Annotations",myMap.GetDisplayCoordSys(), colTypes,1Pkcol,false);
AnotationsDataproviderHelper tab Helper =  new AnotationsDataproviderHelper(til);
AnnotationTableDeschelper(tableHelper=                                          new
AnnotationTableDescHelper("Annotations");
```

Accesa datos el formato tab de mapinfo
- Creada por TABTableDescHelper y por TableDataProviderHelper
- En el paquete com.mapinfo.dp.tab
- Puede Accesar datos nativos, tablas seamless, tablas geotiff y tablas grid de MapInfo

- Cada tipo de capa basada en RDBMS tiene su propio constructor para su respectivo TableDescHelper y DataProviderHelper
- También comparte un tipo de constructor común para el DataProviderHelper

- Este constructor tiene parametros de entrada
  - String Url
  - Properties connectionPropierties (user, password, etc)
  - String driverClassName

```
String [] idColumn = ("ROW");
OraSoTableDescHelper oraTDH = new
OraSoTableDescHelper ("CANADA", bQuotes, idColumn, "GEOLOGIC",
CoordSys.LongLatDatumless, 2, null);
Propieties connProps = new Properties ();
connProps.put ("user", "mxtj");
connProps.put ("password", "secret")
OraSoDataProviderHelper oraDPH = nw
OraSoDataProviderHelper ("jdbc:oracle:thin: hostmachine:1521:s
Id", connProps, "oracle.jdbc.driver.oracledriver");
```

- Usados para mapear con dispositivos conocidos

Las constantes DriverType se usan para mapear dispositivos conocidos
Ej. DriverType.thick=OCI driver on Oracle 8i

```
XYTableDescHelper tabDesc = new
XYTableDescHelper ("myTable, owner,
bQuotes "longitude", "latitude", "idCol",
cSys;
```

- Especificar las columnas X y Y
- Cuidado con la columna de tipo "real" en el SQL server

Versiones anteriores de MapXtreme Java usaban un esquema de pooling de conexiones interno

El pooling está expuesto en la version 3
Configura el numero de threads por pre-iniciar, el numero máximo de threads para administrar y el tiempo de espera

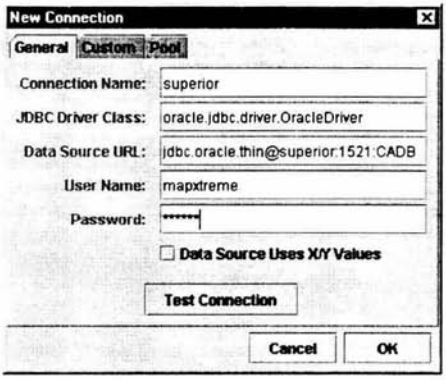
- Miconnections.properties es el nombre del archivo de configuración que contiene el pool de conexiones
- Pueden definirse múltiples conectores JDBC con este archivo
- Se pueden establecer multiples tipos de conexiones JDBC a diferentes data sources

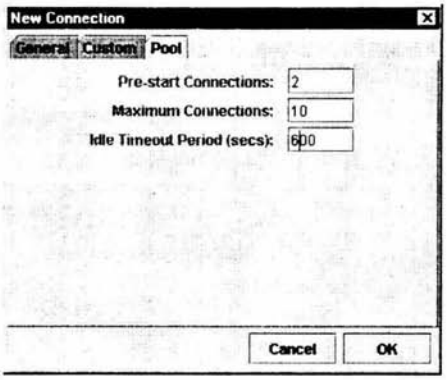
- Oracle 8i, DB2, IUS, XY*, etc

Entrada Ejemplo:

```
Connection1_name=ProjectMaps
Connection1_driver=oracle.jdbc.driver.OracleDriver
Connection1_url=jdbc:oracle:thin:@hostmachine:port:sld
Connection1_user=mapxtreme
Connection1_password=secret
Connection1_is_xy=false
Connection1_prestart=4
Connection1_max=15
Connection1_timeout=300
Connection1_prefetch=75
```

Este archivo puede ser editado manualmente, o administrado a traves del ConnectionsManager





- Este archivo sera usado automáticamente por el MapXtremeServlet, necesita estar en el classpath del servlet
- El pool de conexiones JDBC se crea durante el método init del servlet

- Las conecciones seran establecidas antes de que arrive el primer cliente

- El pool de conecciones es un recurso nombrados
- Todos los DataProviderHelpers para RDBMS ahora comparte un constructor comun
  - Los parametros de entrada son:
    * String Url
    * Properties connectionsProps (usuario, password, prefetch, etc.)
    * String driverClassName

- El pool de conexiones pueden ser accesadas unicamente a través de este constructor por siguiendo una naming convention
- Ningún pool de conexiones se hará mientras se use cualquiera de las otras formas del constructor DPHelper

- La URL de conexión debe estar en la siguente forma

"jdbc:mipool:resource_name"
OraSoDataProviderHelper oraDPH =
New OraSoDataProviderHelper (
"jdbc:mipool:projectmaps",
Null, null)

- Note que el nombre del driver y mas importantemente los parámetros de conexión no están especificados
- La información importante (usuario, password) de la conexión no es trasmitida a través de la red

- Usar el pool de conexiones y de recursos nombrados es un método altamente recomendado para el acceso a data sources JDBC
- Es un medio seguro para que los clientes accesen los data sources de JDBC
- El pool de conexiones incrementara altamente el desempeño de las aplicaciones

GeoTIFFTableDescHelper tabHelper =new
GeoTIFFTableDescHelper (
file:///C:\\image\\dcquad.tif);
GeoTIFFDataProvider dpHelpe =
New GeoTIFFDataProviderHelper ();

- Instanciar una *TableDescHelper
- Instanciar una *DataProviderHelper
- Instanciar una *DataProviderRef
- Use los metodos add o insert para capas
- Recordatorio: si se está usando un driver JDBC, asegurese de que está en el classpath

- Con MapXtremeJava las capas pueden ser creadas a partir de los siguentes data sources:
  - Annotation (en memoria)
  - TAB
  - Oracle 8i
  - IUS Spatialware
  - DB2 Spatialware
  - JDBC XY
  - Geo Tiff Data Provider
  - Shape
  - Mapinfo Grid

- El renderer es una interface
- Los objetos que implementan la interface pueden producir imágenes de mapas
- Dos implementaciones de render

- MapXtremeImageRenderer
- LocalRenderer

- Similar en concepto al MapXtremeDataProviderRef

- Usado cuando un cliente MapJ quiere dirigir el rendering de mapas a el MapXtremeServlet
- Un MapXtremeImageRenderer está construido con la URL de donde está montado el MapXtremeServlet y, opcionalmente un tipo MIME
    - El tipo de MIME por default es GIF

```
MapXtremeImageRenderer renderer = new
MapXtremeImageRenderer (http://servermachine/mxtj30/s
ervlet/mapxtremeservlet);

MapXtremeImageRenderer renderer = new
MapXtremeImageRenderer (http://servermachine/mxtj30/s
ervlet/mapxtremeservlet, "image/png");
```

- El método render de MapJ generará una petición de imagen al MapXtremeServlet

```
MapJ.render(renderer);
```

- Paso extra para que MapXtremeServlet regrese el mapa
    - El mapa puede regresarse como un archivo, un flujo o como una imagen.

```
Renderer.toFile("c:\\mxtj\\mymap.gif");
```

- Un MapXtremeImageRenderer puede ser usado mientras el socket se libere de datos después de reusarse.

- Invocar ya sea toFile(), toStream ó toImage() y luego a dispose()

- Hace el render del mapa a un objeto Java Graphics especificado

```
LocalRenderer renderer = new
  LocalRenderer (component);
```

- Usado internamente por VisualMapJ
- Usado internamente por MapXtremeServlet

- El método render causará que toda la información asociada a los mapas sea retomada y redibujados a los gráficos del componente
- Si se usa dentro de un programa cliente aun se requiere administrar cuando regenerar la imagen del mapa
    - Peticion de Zoom In -> re-genera la imagen
    - Componente obstruido por otra ventana -> no re-genera la imagen.

- Hay dos implementaciones del renderer:

- MapXtremeImageRenderer
- LocalRenderer
    - MapXtremeImageRenderer hace que el MapXtremeServlet dibuje una imagen del mapa
    - El LocalRender dibuja a un objeto Java Graphics

- El objeto renditions encapsula las propiedades de visualizacion para objetos tanto graficos como de texto.
- Un objeto rendition está asociado de manera natural con cada feature
- El rendition de un feature puede modificarse mediante el uso de themes
- La clase rendition en MapXtreme Java Versión 3 implementa el Java2D para ofrecer capacidades de despliegue más sofisticadas

- Nuevo motor de rendering en MXTJ 3.0
    - Java2D
    - Simbolos Vectoriales, lineas paralelas, etc
- Rendiciones por registro
    - Capacidad para guardar un rendition en cada uno de los registros de la base de datos

- Permite un mejor control del rendering sobre datos espaciales de JDBC

- Almacenado en la tabla geometrica
- El formato puede ser
  - MapBasic
  - MapXtremeJava

- El objeto rendition tiene varias propiedades individuales del tipo Rendition.Property que controlan el como luce una figura

- Fill
Controla como se rellena una region
- Stroke
Controla como se dibuja una linea o un borde
- Symbol
Controla como se dibuja un símbolo

- Rendition.FILL

- Valor de tipo Color para rellenos solidos
- Valor de tipo Rendition para relleno de símbolos
  - Una región se rellena "agrupando" símbolos
- Rendition.FILL_OPACITY
- Controla que tan opaco o transparente es el relleno de una región
- Rango 0.0 (transparente) – 1.0 (opaco)

- Rendition.STROKE
  - Valor de tipo Color para pintado sólido
  - Valor de tipo Rendition para pintar símbolos
- Rendition.STROKE_OPACITY
  - Controla la opacidad o transparencia de una línea
  - Rango 0.0(transparente) – 1.0(opaco)
- Rendiction.STROKE_WIDTH
  - Controla el ancho de una linea

- Rendition.STROKE.LINECAP
  - Rendition.LineCap.BUTT
  - Rendition.LineCap.ROUND
  - Rendition.LineCap.SQUARE
- Rendition.STROKE_LINEJOIN
  - Rendition.LineJoin.BEVEL
  - Rendition.LineJoin.ROUND
  - Rendition.LineJoin.MITER

  Rendition.STROKE_PARALLELARRAY
  - Valor de tipo Rendition.ParallelLine[]
  - Rendition.ParallelLine
    - Rendition
    - Offset (float); + (to right), –(to left)

  - Cada línea paralela se dibuja a *offset* unidades de la línea padre usando el Rendition proporcionado

- Rendition.STROKE_DASHARRAY
  - Valor de tipo float[]
  - Especifica el patron de punteo
  - {5,3} Significa 5 unidades de linea , 3 unidades de espacio
- Rendition.STROKE_DASHOFFSET
- Valor de tipo float
- Desplazamiento en la línea antes de que el patron punteado comience

  - Lineas Paralelas

- Actualmente no cierran limpiamente
- Actualmente no trabajan bien con ángulos agudos

- Rendition.STROKE.MARKERARRAY
  - Conjunto de marcadores de simbolos
  - Los simbolos se rotan para coincidir con el segmento

- Rendition.SYMBOL_MODE
  - Font, Imagen (GIF,JPEG, etc), o vector
- True Type Fonts - Familia, Tamaño, Color, Negrita, Itálica, halo Subrayado, etc
  - Ahora mas de un carácter
- Image - URL
  - Soporte para transparencia de imagenes

- Vector
  - Rendition
  - Forma
- Otras propiedades
  - Affine Transform
  - Opacidad del símbolo
  - Color de fondo

- Estableciendo y obteniendo las propiedades

```
Rendition rend = new Rendition ();
rend.setValue(Rendition.STROKE, Color.red);
Color c = rend.getColor (rendition.STROKE);
Float width = rend getFloar (Renditions.STROKE_WIDTH);
```

- El objeto Rendition encapsula las propiedades de despliegue para objetos gráficos y de texto
- Rendition.setValue(property, value) es usado para modificar un rendition

- Una capa de un mapa puede tener uno o mas mapas temáticos asociados
- Un ThemeList de cada capa mantiene la colección de temáticos para una capa

- Un temático puede ser añadido o eliminado de un ThemeList
- Un temático puede insertarse en una posición dada dentro del ThemeList
- Puede accederse a un temático en un ThemeList por su nombre o por su índice
- Los temas pueden ser reordenados dentro de un ThemeList

- Hay 4 tipos de clases de temáticos en MapXtreme Java:

  - Override Theme
  - Ranged Theme
  - Nuevos en la version 3.0
    - Individual Value Theme
    - Selection Theme

- Se utiliza un objeto rendition para sobrecargar las propiedades visuales de todas las figuras dentro de una capa

```
Rendition rend = new Rendition ();
rend.setValue(Rendition.SYMBOL_FOREGROUND, Color.red);

OverrideTheme otheme = new OverrideTheme (rend);
Themelist tl = layer.getThemelist();
tl.add(otheme);
```

- Crear un rendition que sera usada para sobrecargar las opciones de visualización por default de la capa
- Instanciar el objeto OverrideTheme, y especificar la rendition en el constructor
- Añadir el tema al ThemeList

- Crea un rango tematico
- Especifica el numero de rangos para el temático
- Puede dejar que MapXtreme calcule los límites del rango o especificar los límites de los rangos.
- Se puede permitir que MapXtreme extienda los rangos o puede especificarse un rendition para cada rango.

- Contiene Máximo, Mínimo, media, y desviación estándar de los valores en una columna.
- Regreso de el método de capa fetchColumnStatistics()
- Usado en el objeto bucketer para crear un vector de breakpoints
- El tiempo de cálculo puede ser tardado para grandes cantidades de datos
- Cálculo una sola vez de la informacion que está almacenada en el servidor en un archivo .stx
- Si los datos cambian el archivo .stx tiene que ser borrado de el servidor para que las estadísticas se recalculen

```
Calcula los breackpoints para los rangos en un RangedTheme
//trae la columna estadisticas
ColumnStatistics colStas =
lyr.fetchColumnstatistics(themeCol);
//computa la distribucion de datos con 4 paros y rangos iguales
Vector rBreaks =
Bucketer.DISTRIBUTION_TYPE_EQUAL_RANGES) ;
```

- Los valores de los rangos calculados dependen de: - Los valores de datos en la columna
  - El numero de rangos
  - El tipo de distribucion usada
  - Redondeo (si fue usado)

- EqualCount (pretererminado)
  - Mismo numero de records en cada rango
  - Para 100 records y 4 rangos pone aproximadamente 25 records en cada rango
- EqualRanges
  -Divide records a lo largo de rangos de igual tamaño
  -Para records con valores de 1-100 y 4 rangos, los rangos serian 1-25 26-50 51-75 76-100

- Desviacion Estándar
  - Rompe en el rango medio de la media
  - Hace rangos encima y debajo de el rango de en medio son una desviación estándar arriba y abajo del significado

- Puede ser usado para crear breakpoints para los rangos
- Redondea abajo el extremo inferior de el rango
- Redondea arriba el extremo mas alto de el rango

GLOSARIO

ASCII : Caracteres para la utilización de un sistema.

Apache: Es un proyecto (WEB Server) caracterizado por un proceso de colaboración, consenso basado al desarrollo, una licencia abierta del software, y un deseo de crear software de la alta calidad .

Autocad: Sistema para la manipulación de datos de diseño.

AutoDesk: Empresa que ofrece servicios y productos de :Arquitectura , construcción industria y Fabricación Medios digitales.

BEANS: Clases de JAVA que actúan como componente para una futura utilización.

CAD: Diseño asistido por Ordenador

CORBA: Permite la programación distribuida.

DM: Desktop Mapping Sistemas de análisis y visualización integrados entre las aplicaciones Desktop de ordenador personal.

FEATURE: Característica de un elemento geográfico ejemplo un punto un polígono una recta.

GML : Estructura de XML que contiene información geográfica.

HTTP: Protocolo de comunicación utilizado para el intercambio de información

INEGI: Instituto Nacional de Estadística.

JAVA: Lenguaje de Programación.

JDBC: Driver que utiliza JAVA para la comunicación con bases de datos.

MapXtreme: Motor de mapas que permite la interacción con bases de datos geográficas.

MapInfo: Empresa encargada de proporcionar servicios en el ámbito geográfico.

MIDDLEWARE: Lógica del sistema en la capa de negocio.

MVC: Metodología que permite separar la lógica del sistema y del diseño.

NETSCAPE: Navegador para la consulta de paginas en la red.

Package: Palabra reservada por JAVA para la integración de clases por grupos.

RDBMS: (Relational DataBase Management System - Sistema de gestión de bases de datos relacional).

SERVLETS: Clases de JAVA que permiten el envío de mensajes por el protocolo HTTP estas clases actúan en la capa de la lógica del sistema

SICORI: Es la dependencia de PEMEX que a nivel corporativo, proporciona servicios de Información geográfica relacionados con la industria petrolera.

SIG: Sistema de Información Geográfica

SVG: Estructura de XML que contiene las características de figuras por medio de un visualizador.

Tomcat: Web Server que permite dar respuestas a clientes dinámicamente vía WEB.

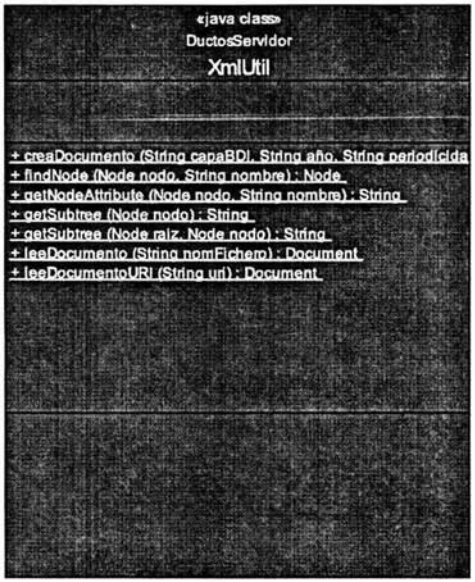
UML: Metodología que permite el modelo de sistemas.

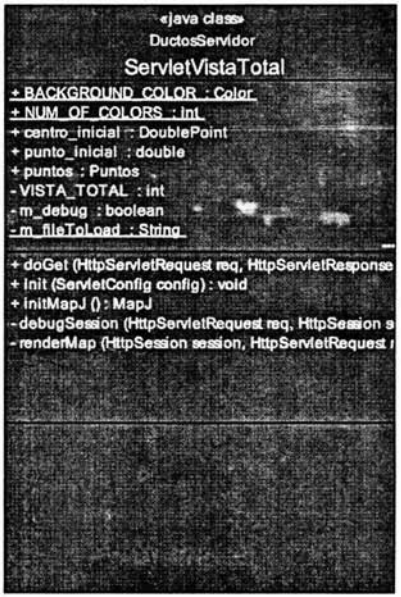
UNIX: Sistema operativo.

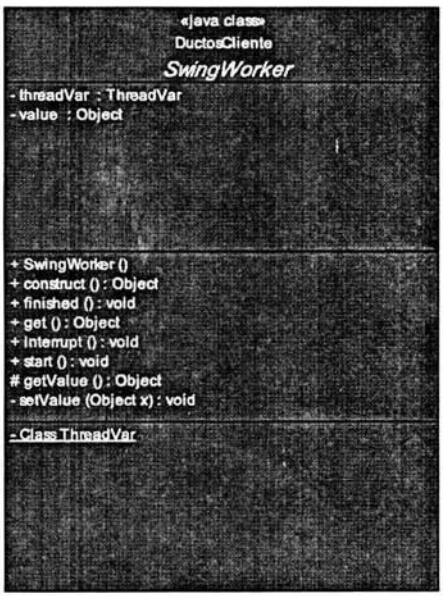
Web Services: Servicios que pueden ser utilizados vía WEB.

WML: Protocolo de comunicación utilizado para el intercambio de información utilizado principalmente por dispositivos móviles.

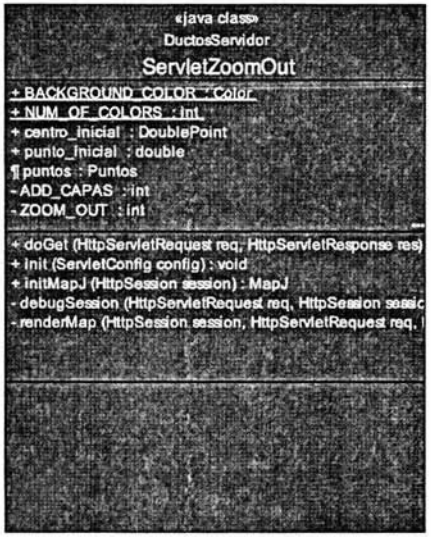
XML: Sintaxis o metalenguaje que nos permite crear nuestros propios lenguajes para estructurar contenidos, utilizando etiquetas o tags.

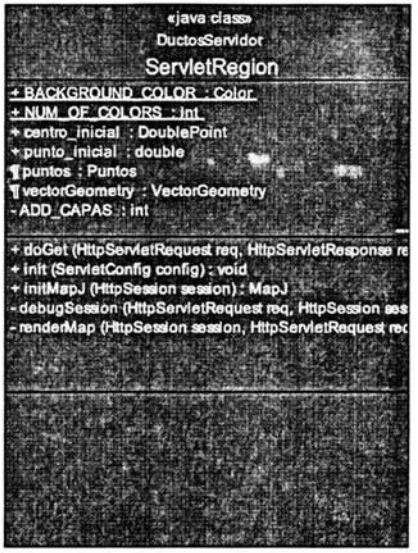

«java class»
DuctosServidor
XmlUtil
+ creaDocumento (String capaBD, String año, String periodicida
+ findNode (Node nodo, String nombre) : Node
+ getNodeAttribute (Node nodo, String nombre) : String
+ getSubtree (Node nodo) : String
+ getSubtree (Node raiz, Node nodo) : String
+ leeDocumento (String nomFichero) : Document
+ leeDocumentoURI (String uri) : Document

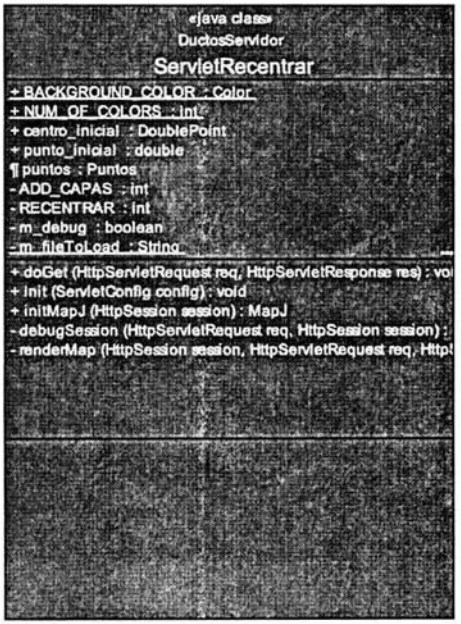

«java class»
DuctosServidor
ServletVistaTotal
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
+ puntos : Puntos
- VISTA_TOTAL : int
- m_debug : boolean
- m_fileToLoad : String
+ doGet (HttpServletRequest req, HttpServletResponse
+ init (ServletConfig config) : void
+ initMapJ () : MapJ
- debugSession (HttpServletRequest req, HttpSession s
- renderMap (HttpSession session, HttpServletRequest

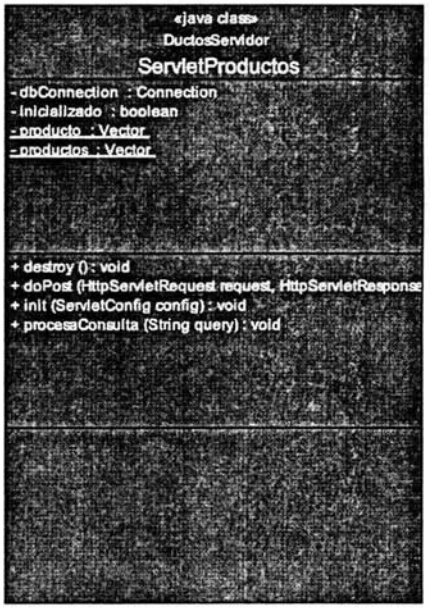

«java class»
DuctosCliente
SwingWorker
- threadVar : ThreadVar
- value : Object
+ SwingWorker ()
+ construct () : Object
+ finished () : void
+ get () : Object
+ interrupt () : void
+ start () : void
# getValue () : Object
- setValue (Object x) : void
- Class ThreadVar

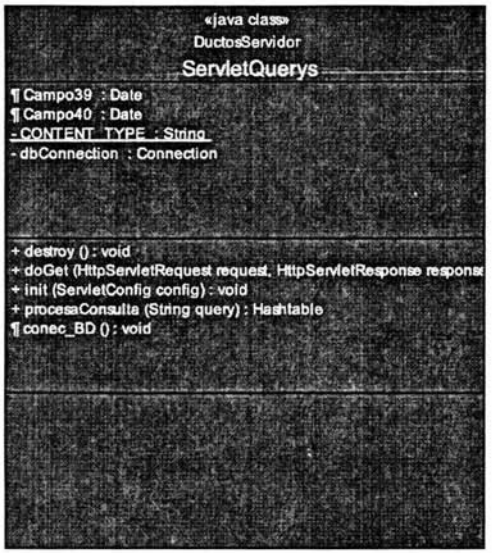

«java class»
DuctosServidor
ServletZoomIn
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
- ADD_CAPAS : int
- ZOOM_IN : int
- m_debug : boolean
- m_fileToLoad : String
- m_mapHeight : int
- m_mapPath : String
- m_mapWidth : int
- m_mxtURL : String
+ doGet (HttpServletRequest req, HttpServletResponse re
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
- debugSession (HttpServletRequest req, HttpSession ses
- renderMap (HttpSession session, HttpServletRequest re

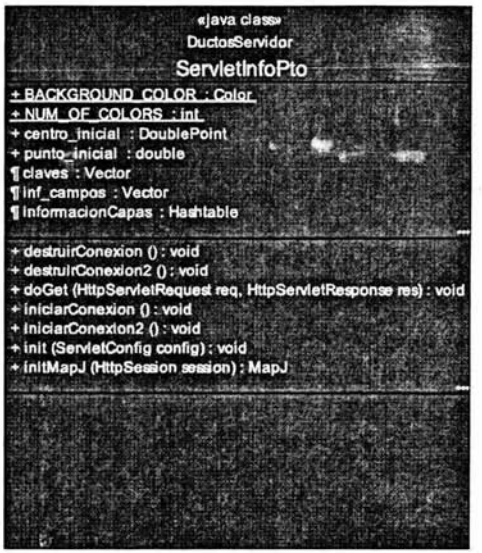

«java class»
DuctosServidor
ServletZoomOut
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ puntos : Puntos
- ADD_CAPAS : int
- ZOOM_OUT : int
+ doGet (HttpServletRequest req, HttpServletResponse res)
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
- debugSession (HttpServletRequest req, HttpSession sessic
- renderMap (HttpSession session, HttpServletRequest req,

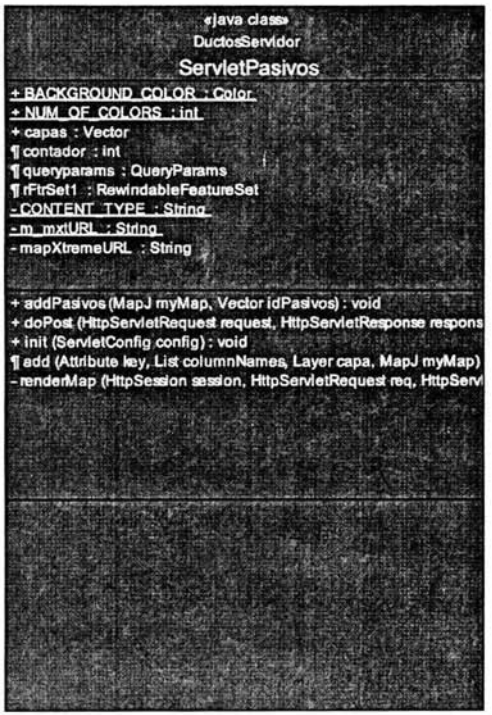

«java class»
DuctosServidor
ServletRegion
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ puntos : Puntos
¶ vectorGeometry : VectorGeometry
- ADD_CAPAS : int
+ doGet (HttpServletRequest req, HttpServletResponse re
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
- debugSession (HttpServletRequest req, HttpSession ses
- renderMap (HttpSession session, HttpServletRequest rec

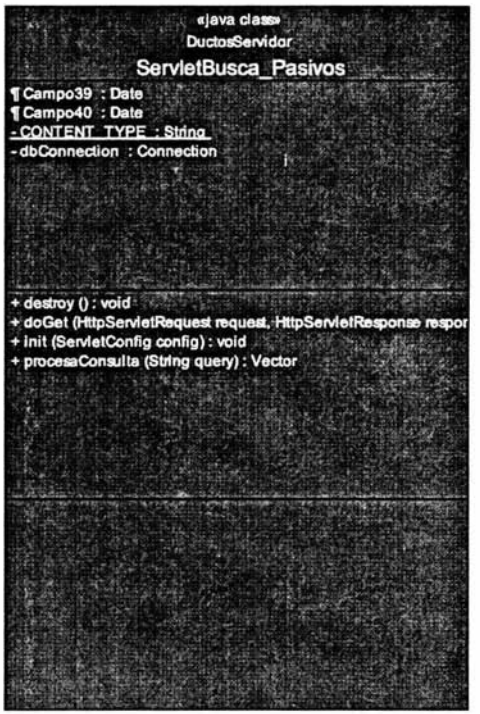

«java class»
DuctosServidor
ServletRecentrar
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ puntos : Puntos
- ADD_CAPAS : int
- RECENTRAR : int
- m_debug : boolean
- m_fileToLoad : String
+ doGet (HttpServletRequest req, HttpServletResponse res) : vo
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
- debugSession (HttpServletRequest req, HttpSession session) :
- renderMap (HttpSession session, HttpServletRequest req, Http

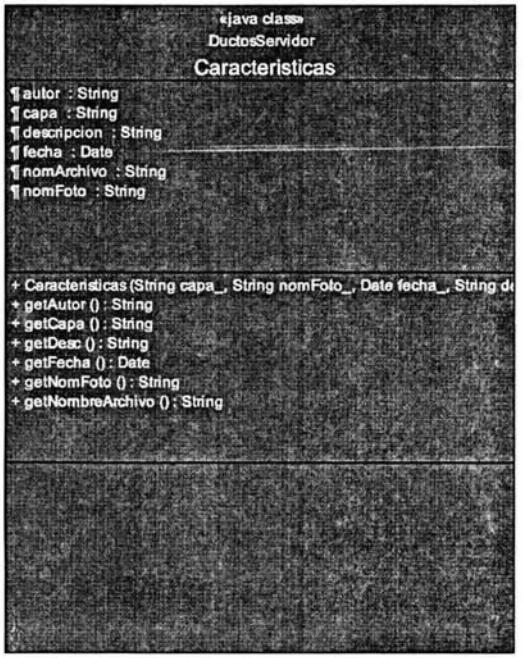

«java class»
DuctosServidor
ServletProductos
- dbConnection : Connection
- inicializado : boolean
- producto : Vector
- productos : Vector
+ destroy () : void
+ doPost (HttpServletRequest request, HttpServletResponse
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : void

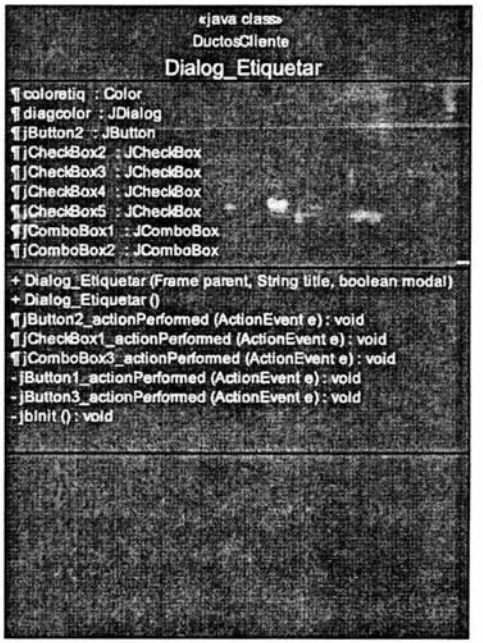

«java class»
DuctosServidor
ServletQuerys
¶ Campo39 : Date
¶ Campo40 : Date
- CONTENT_TYPE : String
- dbConnection : Connection
+ destroy () : void
+ doGet (HttpServletRequest request, HttpServletResponse respons
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : Hashtable
¶ conec_BD () : void

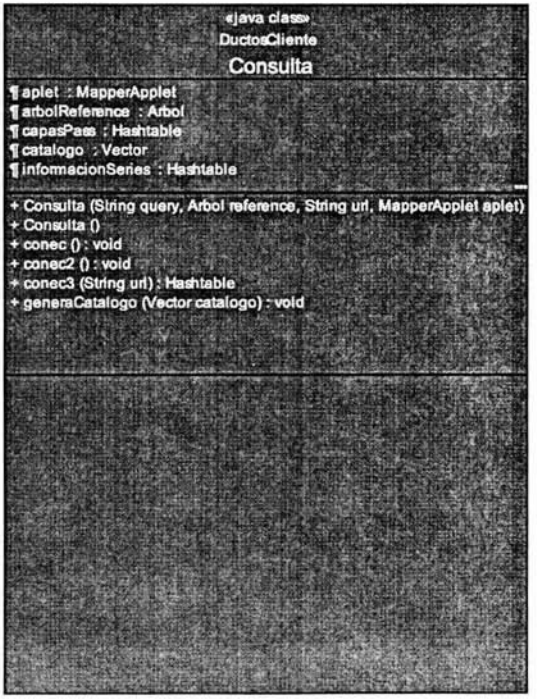

«java class»
DuctosServidor
ServletInfoPto
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ claves : Vector
¶ inf_campos : Vector
¶ informacionCapas : Hashtable
+ destruirConexion () : void
+ destruirConexion2 () : void
+ doGet (HttpServletRequest req, HttpServletResponse res) : void
+ iniciarConexion () : void
+ iniciarConexion2 () : void
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ

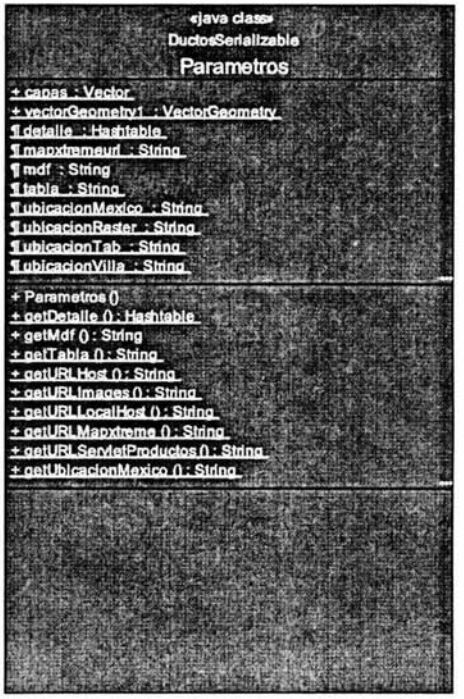

«java class»
DuctosServidor
ServletPasivos
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ capas : Vector
¶ contador : int
¶ queryparams : QueryParams
¶ rFtrSet1 : RewindableFeatureSet
- CONTENT_TYPE : String
- m_mxtURL : String
- mapXtremeURL : String
+ addPasivos (MapJ myMap, Vector idPasivos) : void
+ doPost (HttpServletRequest request, HttpServletResponse respons
+ init (ServletConfig config) : void
¶ add (Attribute key, List columnNames, Layer capa, MapJ myMap)
- renderMap (HttpSession session, HttpServletRequest req, HttpServl

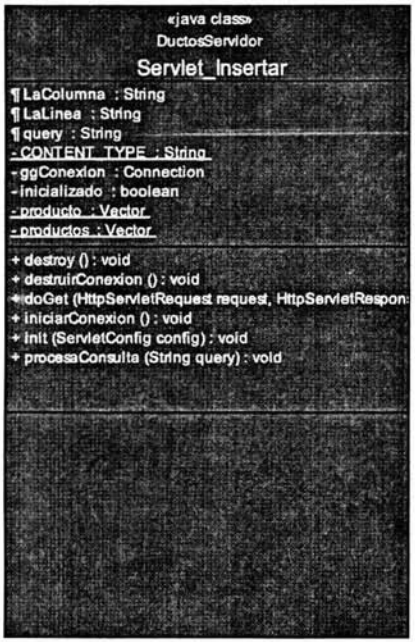

«java class»
DuctosServidor
ServletBusca_Pasivos
¶ Campo39 : Date
¶ Campo40 : Date
- CONTENT_TYPE : String
- dbConnection : Connection
+ destroy () : void
+ doGet (HttpServletRequest request, HttpServletResponse respor
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : Vector

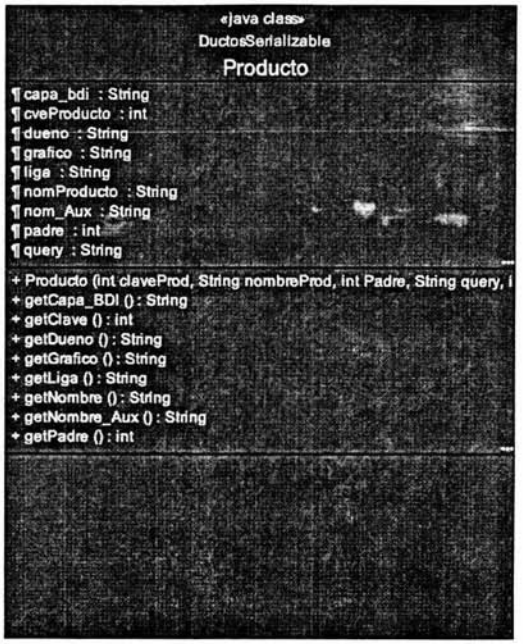

«java class»
DuctosServidor
Caracteristicas
¶ autor : String
¶ capa : String
¶ descripcion : String
¶ fecha : Date
¶ nomArchivo : String
¶ nomFoto : String
+ Caracteristicas (String capa_, String nomFoto_, Date fecha_, String d
+ getAutor () : String
+ getCapa () : String
+ getDesc () : String
+ getFecha () : Date
+ getNomFoto () : String
+ getNombreArchivo () : String

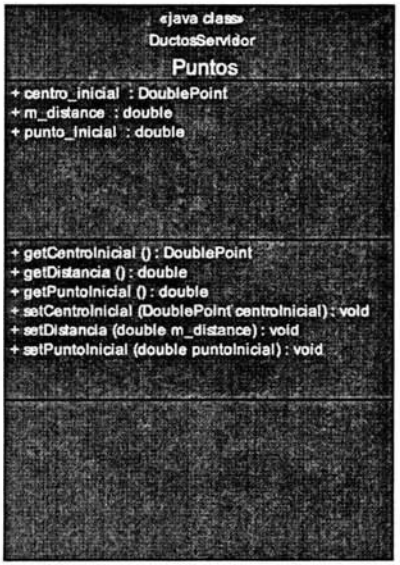

«java class»
DuctosCliente
Dialog_Etiquetar
¶ coloretiq : Color
¶ diagcolor : JDialog
¶ jButton2 : JButton
¶ jCheckBox2 : JCheckBox
¶ jCheckBox3 : JCheckBox
¶ jCheckBox4 : JCheckBox
¶ jCheckBox5 : JCheckBox
¶ jComboBox1 : JComboBox
¶ jComboBox2 : JComboBox
+ Dialog_Etiquetar (Frame parent, String title, boolean modal)
+ Dialog_Etiquetar ()
¶ jButton2_actionPerformed (ActionEvent e) : void
¶ jCheckBox1_actionPerformed (ActionEvent e) : void
¶ jComboBox3_actionPerformed (ActionEvent e) : void
- jButton1_actionPerformed (ActionEvent e) : void
- jButton3_actionPerformed (ActionEvent e) : void
- jbInit () : void

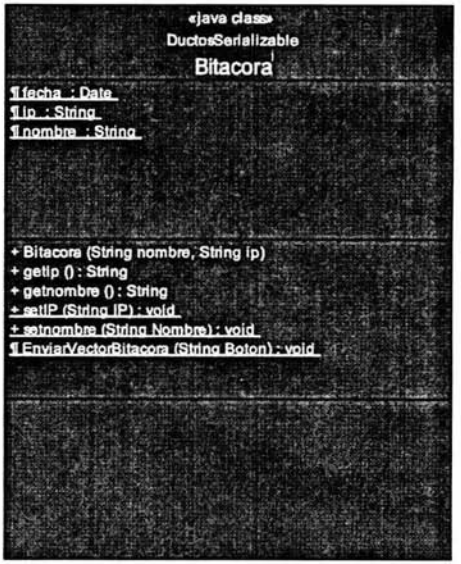

«java class»
DuctosCliente
Consulta
¶ aplet : MapperApplet
¶ arbolReference : Arbol
¶ capasPass : Hashtable
¶ catalogo : Vector
¶ informacionSeries : Hashtable
+ Consulta (String query, Arbol reference, String url, MapperApplet aplet)
+ Consulta ()
+ conec () : void
+ conec2 () : void
+ conec3 (String url) : Hashtable
+ generaCatalogo (Vector catalogo) : void

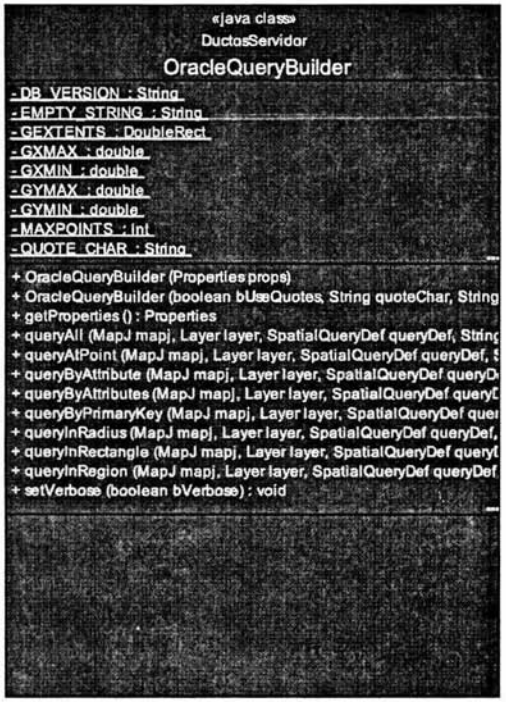

«java class»
DuctosSerializable
Parametros
+ capas : Vector
+ vectorGeometry1 : VectorGeometry
¶ detalle : Hashtable
¶ mapxtremeurl : String
¶ mdf : String
¶ tabla : String
¶ ubicacionMexico : String
¶ ubicacionRaster : String
¶ ubicacionTab : String
¶ ubicacionVilla : String
+ Parametros ()
+ getDetalle () : Hashtable
+ getMdf () : String
+ getTabla () : String
+ getURLHost () : String
+ getURLImages () : String
+ getURLLocalHost () : String
+ getURLMapxtreme () : String
+ getURLServletProductos () : String
+ getUbicacionMexico () : String

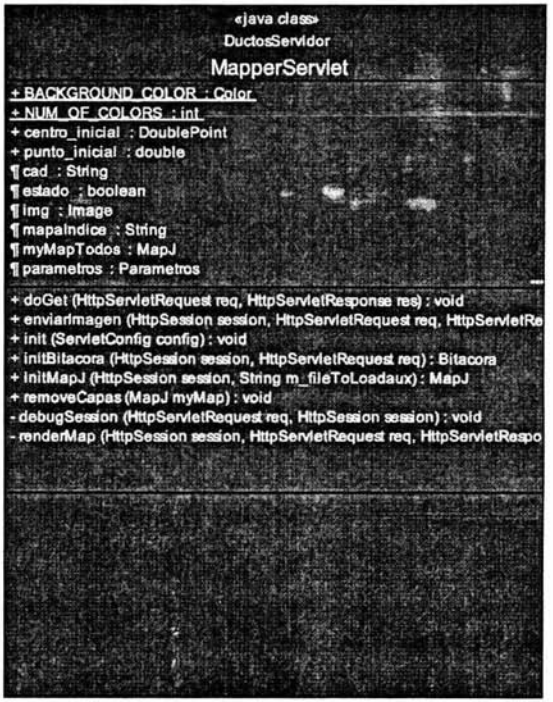

«java class»
DuctosServidor
Servlet_Insertar
¶ LaColumna : String
¶ LaLinea : String
¶ query : String
- CONTENT_TYPE : String
- ggConexion : Connection
- inicializado : boolean
- producto : Vector
- productos : Vector
+ destroy () : void
+ destruirConexion () : void
+ doGet (HttpServletRequest request, HttpServletRespon
+ iniciarConexion () : void
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : void


«java class»
DuctosSerializable
Producto
¶ capa_bdi : String
¶ cveProducto : int
¶ dueno : String
¶ grafico : String
¶ liga : String
¶ nomProducto : String
¶ nom_Aux : String
¶ padre : int
¶ query : String
+ Producto (int claveProd, String nombreProd, int Padre, String query, I
+ getCapa_BDI () : String
+ getClave () : int
+ getDueno () : String
+ getGrafico () : String
+ getLiga () : String
+ getNombre () : String
+ getNombre_Aux () : String
+ getPadre () : int

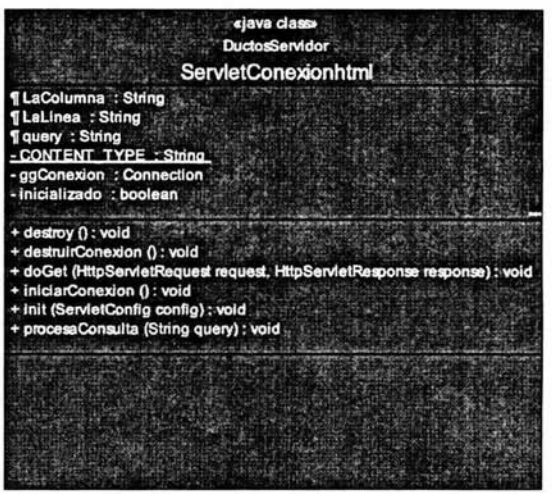

«java class»
DuctosServidor
Puntos
+ centro_inicial : DoublePoint
+ m_distance : double
+ punto_inicial : double
+ getCentroInicial () : DoublePoint
+ getDistancia () : double
+ getPuntoInicial () : double
+ setCentroInicial (DoublePoint centroInicial) : void
+ setDistancia (double m_distance) : void
+ setPuntoInicial (double puntoInicial) : void

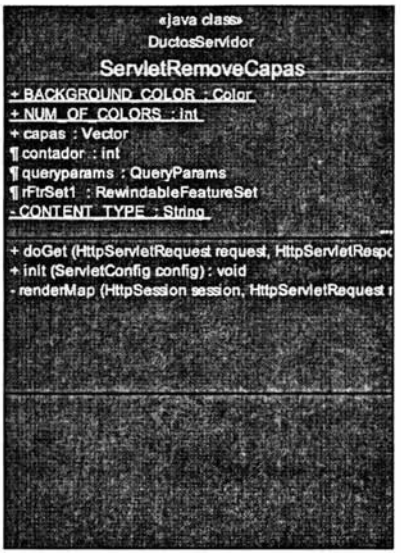

«java class»
DuctosSerializable
Bitacora
¶ fecha : Date
¶ ip : String
¶ nombre : String
+ Bitacore (String nombre, String ip)
+ getIp () : String
+ getnombre () : String
+ setIP (String IP) : void
+ setnombre (String Nombre) : void
¶ EnviarVectorBitacora (String Boton) : void

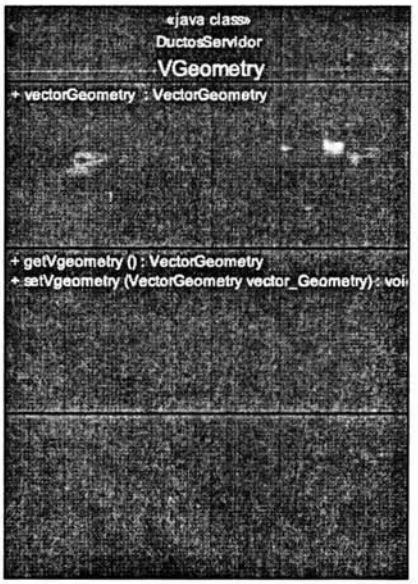

«java class»
DuctosServidor
OracleQueryBuilder
- DB_VERSION : String
- EMPTY_STRING : String
- GEXTENTS : DoubleRect
- GXMAX : double
- GXMIN : double
- GYMAX : double
- GYMIN : double
- MAXPOINTS : int
- QUOTE_CHAR : String
+ OracleQueryBuilder (Properties props)
+ OracleQueryBuilder (boolean bUseQuotes, String quoteChar, String
+ getProperties () : Properties
+ queryAll (MapJ mapj, Layer layer, SpatialQueryDef queryDef, Strin
+ queryAtPoint (MapJ mapj, Layer layer, SpatialQueryDef queryDef,
+ queryByAttribute (MapJ mapj, Layer layer, SpatialQueryDef queryD
+ queryByAttributes (MapJ mapj, Layer layer, SpatialQueryDef queryD
+ queryByPrimaryKey (MapJ mapj, Layer layer, SpatialQueryDef quer
+ queryInRadius (MapJ mapj, Layer layer, SpatialQueryDef queryDef,
+ queryInRectangle (MapJ mapj, Layer layer, SpatialQueryDef queryD
+ queryInRegion (MapJ mapj, Layer layer, SpatialQueryDef queryDef
+ setVerbose (boolean bVerbose) : void

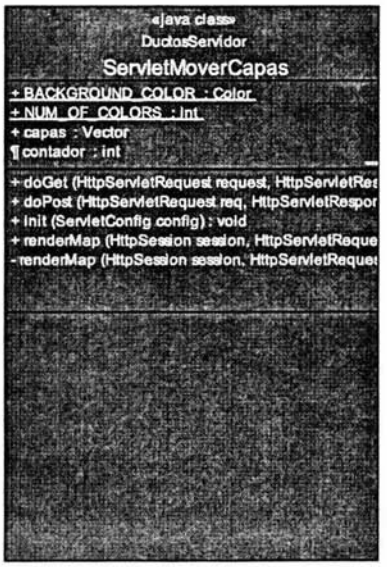

«java class»
DuctosServidor
MapperServlet
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ cad : String
¶ estado : boolean
¶ img : Image
¶ mapaIndice : String
¶ myMapTodos : MapJ
¶ parametros : Parametros
+ doGet (HttpServletRequest req, HttpServletResponse res) : void
+ enviarImagen (HttpSession session, HttpServletRequest req, HttpServletRe
+ init (ServletConfig config) : void
+ initBitacora (HttpSession session, HttpServletRequest req) : Bitacora
+ initMapJ (HttpSession session, String m_fileToLoadaux) : MapJ
+ removeCapas (MapJ myMap) : void
- debugSession (HttpServletRequest req, HttpSession session) : void
- renderMap (HttpSession session, HttpServletRequest req, HttpServletRespo

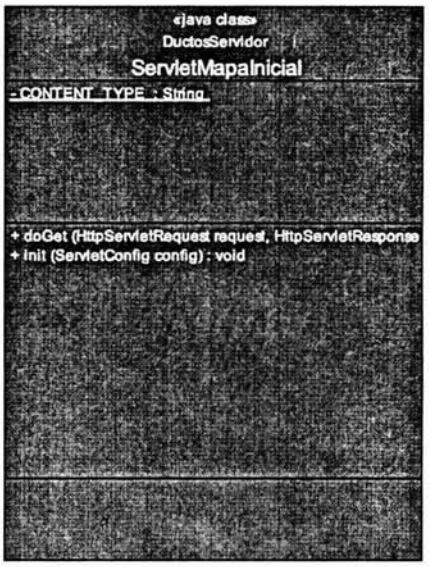

«java class»
DuctosServidor
InformacionPunto
+ InformacionPunto ()
+ obtInfPunto (DoublePoint punto, MapJ myMap) : void

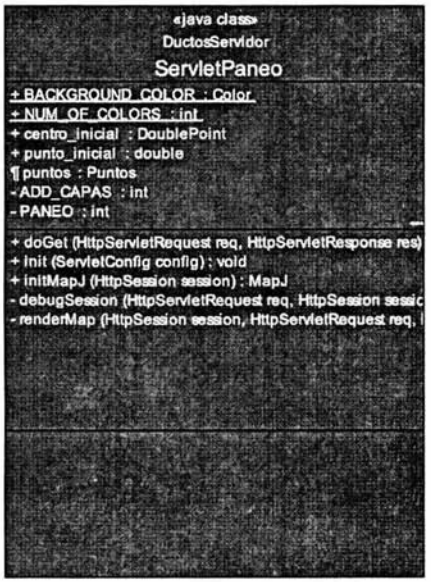

«java class»
DuctosServidor
ServletConexionhtml
¶ LaColumna : String
¶ LaLinea : String
¶ query : String
- CONTENT_TYPE : String
- ggConexion : Connection
- inicializado : boolean
+ destroy () : void
+ destruirConexion () : void
+ doGet (HttpServletRequest request, HttpServletResponse response) : void
+ iniciarConexion () : void
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : void

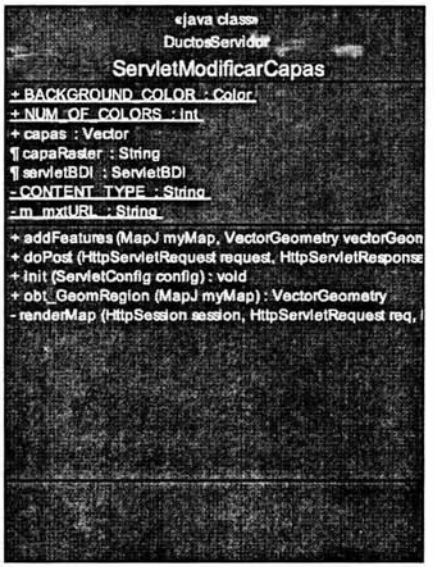

«java class»
DuctosServidor
ServletRemoveCapas
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : Int
+ capas : Vector
¶ contador : int
¶ queryparams : QueryParams
¶ rFtrSet1 : RewindableFeatureSet
- CONTENT_TYPE : String
+ doGet (HttpServletRequest request, HttpServletRespo
+ init (ServletConfig config) : void
- renderMap (HttpSession session, HttpServletRequest

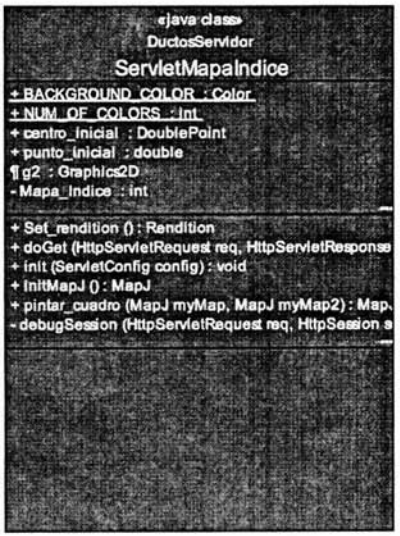

«java class»
DuctosServidor
VGeometry
+ vectorGeometry : VectorGeometry
+ getVgeometry () : VectorGeometry
+ setVgeometry (VectorGeometry vector_Geometry) : voi

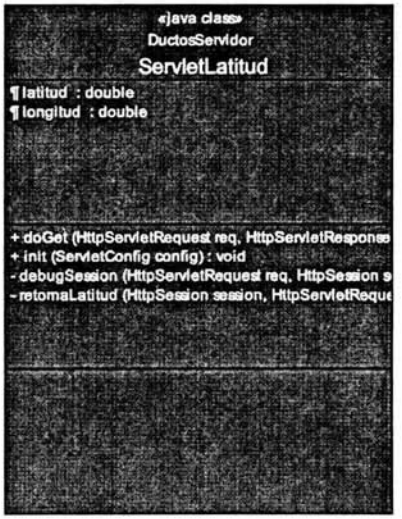

«java class»
DuctosServidor
ServletMoverCapas
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : Int
+ capas : Vector
¶ contador : int
+ doGet (HttpServletRequest request, HttpServletRes
+ doPost (HttpServletRequest req, HttpServletRespor
+ init (ServletConfig config) : void
+ renderMap (HttpSession session, HttpServletReque
- renderMap (HttpSession session, HttpServletReque

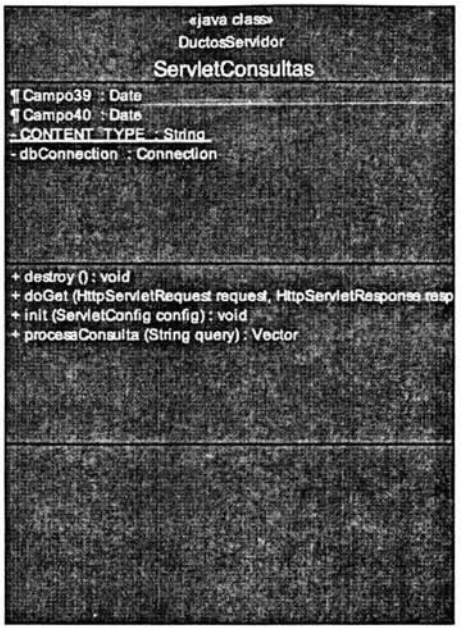

«java class»
DuctosServidor
ServletMapaInicial
- CONTENT_TYPE : String
+ doGet (HttpServletRequest request, HttpServletResponse
+ init (ServletConfig config) : void

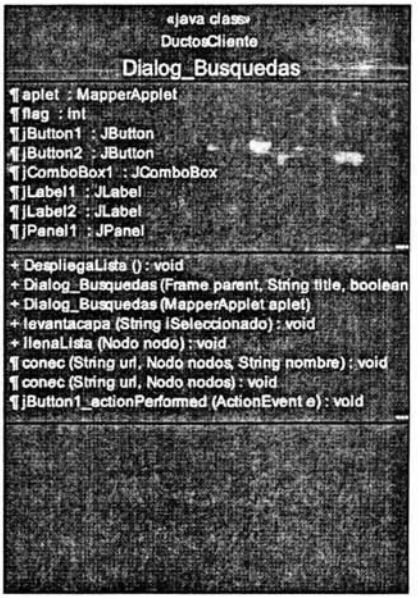

«java class»
DuctosServidor
ServletPaneo
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ puntos : Puntos
- ADD_CAPAS : int
- PANEO : int
+ doGet (HttpServletRequest req, HttpServletResponse res)
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
- debugSession (HttpServletRequest req, HttpSession sessic
- renderMap (HttpSession session, HttpServletRequest req,

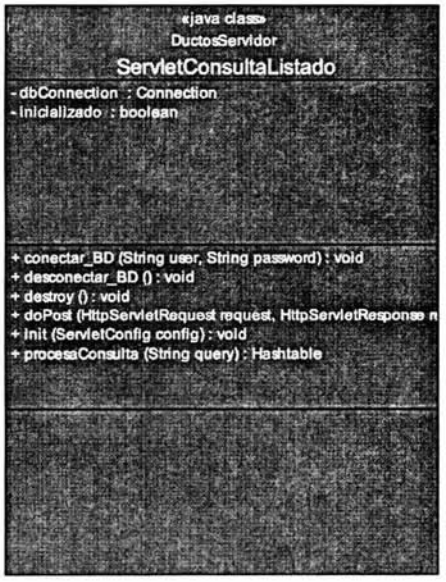

«java class»
DuctosServidor
ServletModificarCapas
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ capas : Vector
¶ capaRaster : String
¶ servletBDI : ServletBDI
- CONTENT_TYPE : String
- m_mxtURL : String
+ addFeatures (MapJ myMap, VectorGeometry vectorGeon
+ doPost (HttpServletRequest request, HttpServletResponse
+ init (ServletConfig config) : void
+ obt_GeomRegion (MapJ myMap) : VectorGeometry
- renderMap (HttpSession session, HttpServletRequest req,

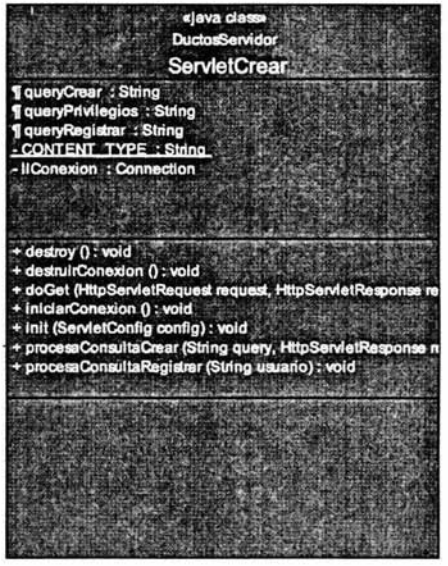

«java class»
DuctosServidor
ServletMapaIndice
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : Int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ g2 : Graphics2D
- Mapa_Indice : int
+ Set_rendition () : Rendition
+ doGet (HttpServletRequest req, HttpServletResponse
+ init (ServletConfig config) : void
+ initMapJ () : MapJ
+ pintar_cuadro (MapJ myMap, MapJ myMap2) : Map
- debugSession (HttpServletRequest req, HttpSession s

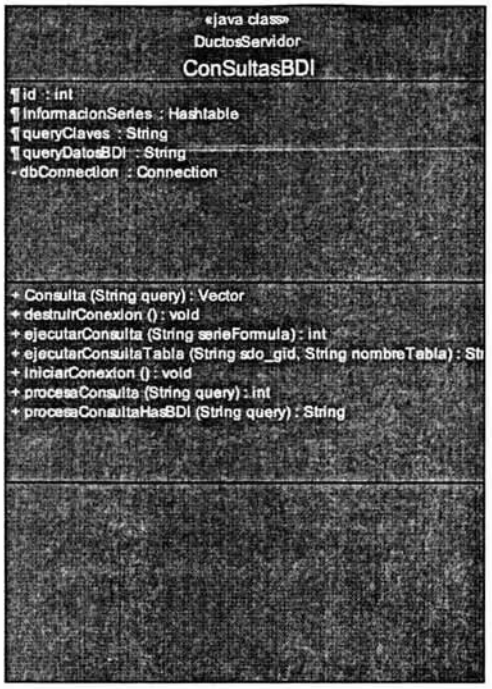

«java class»
DuctosServidor
ServletLatitud
¶ latitud : double
¶ longitud : double
+ doGet (HttpServletRequest req, HttpServletResponse
+ init (ServletConfig config) : void
- debugSession (HttpServletRequest req, HttpSession s
- retomaLatitud (HttpSession session, HttpServletReque

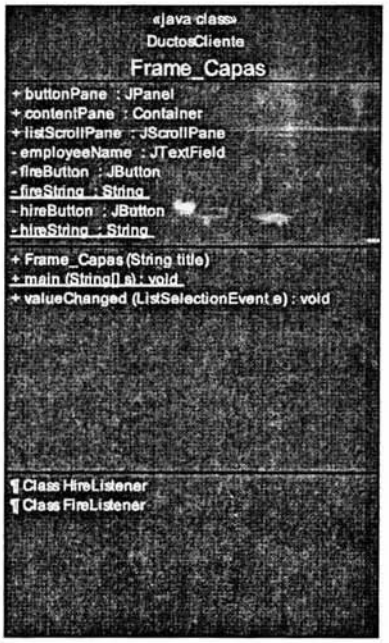

«java class»
DuctosServidor
ServletConsultas
¶ Campo39 : Date
¶ Campo40 : Date
- CONTENT_TYPE : String
- dbConnection : Connection
+ destroy () : void
+ doGet (HttpServletRequest request, HttpServletResponse resp
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : Vector

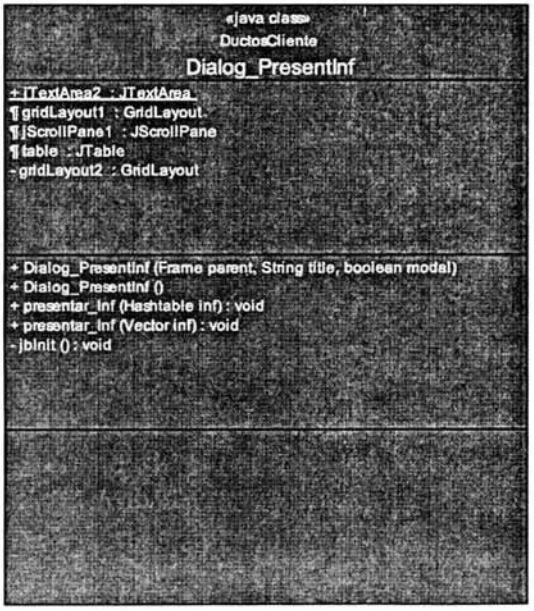

«java class»
DuctosCliente
Dialog_Busquedas
¶ aplet : MapperApplet
¶ flag : int
¶ jButton1 : JButton
¶ jButton2 : JButton
¶ jComboBox1 : JComboBox
¶ jLabel1 : JLabel
¶ jLabel2 : JLabel
¶ jPanel1 : JPanel
+ DespliegaLista () : void
+ Dialog_Busquedas (Frame parent, String title, boolean
+ Dialog_Busquedas (MapperApplet aplet)
+ levantacapa (String lSeleccionado) : void
+ llenaLista (Nodo nodo) : void
¶ conec (String url, Nodo nodos, String nombre) : void
¶ conec (String url, Nodo nodos) : void
¶ jButton1_actionPerformed (ActionEvent e) : void

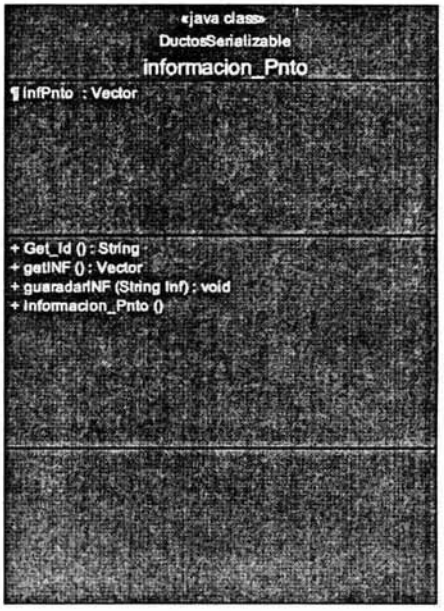

«java class»
DuctosServidor
ServletConsultaListado
- dbConnection : Connection
- inicializado : boolean
+ conectar_BD (String user, String password) : void
+ desconectar_BD () : void
+ destroy () : void
+ doPost (HttpServletRequest request, HttpServletResponse r
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : Hashtable

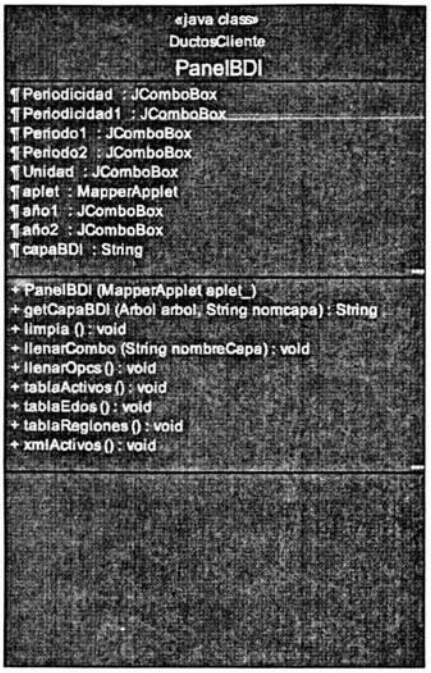

«java class»
DuctosServidor
ServletCrear
¶ queryCrear : String
¶ queryPrivilegios : String
¶ queryRegistrar : String
- CONTENT_TYPE : String
- llConexion : Connection
+ destroy () : void
+ destruirConexion () : void
+ doGet (HttpServletRequest request, HttpServletResponse re
+ iniciarConexion () : void
+ init (ServletConfig config) : void
+ procesaConsultaCrear (String query, HttpServletResponse r
+ procesaConsultaRegistrar (String usuario) : void

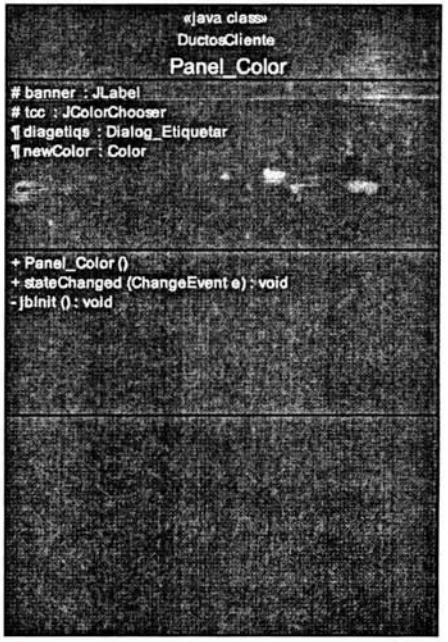

«java class»
DuctosServidor
ConSultasBDI
¶ id : int
¶ informacionSeries : Hashtable
¶ queryClaves : String
¶ queryDatosBDI : String
- dbConnection : Connection
+ Consulta (String query) : Vector
+ destruirConexion () : void
+ ejecutarConsulta (String serieFormula) : int
+ ejecutarConsultaTabla (String sdo_gid, String nombreTabla) : St
+ iniciarConexion () : void
+ procesaConsulta (String query) : int
+ procesaConsultaHasBDI (String query) : String

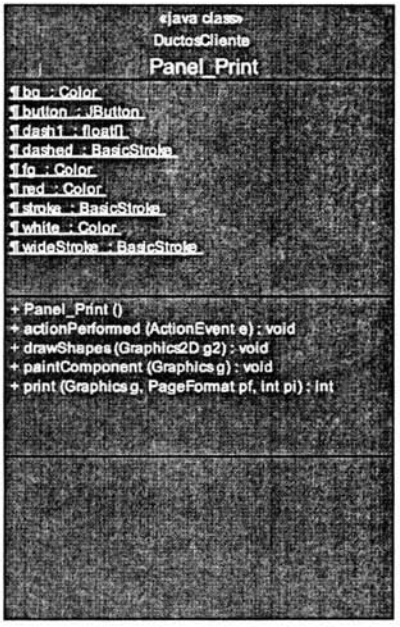

«java class»
DuctosCliente
Frame_Capas
+ buttonPane : JPanel
+ contentPane : Container
+ listScrollPane : JScrollPane
- employeeName : JTextField
- fireButton : JButton
- fireString : String
- hireButton : JButton
- hireString : String
+ Frame_Capas (String title)
+ main (String[] a) : void
+ valueChanged (ListSelectionEvent e) : void
¶ Class HireListener
¶ Class FireListener

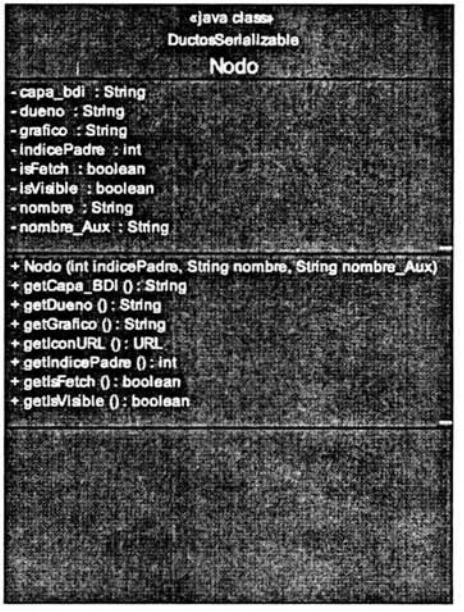

«java class»
DuctosCliente
Dialog_PresentInf
+ jTextArea2 : JTextArea
¶ gridLayout1 : GridLayout
¶ jScrollPane1 : JScrollPane
¶ table : JTable
- gridLayout2 : GridLayout
+ Dialog_PresentInf (Frame parent, String title, boolean modal)
+ Dialog_PresentInf ()
+ presentar_Inf (Hashtable inf) : void
+ presentar_Inf (Vector inf) : void
- jbInit () : void

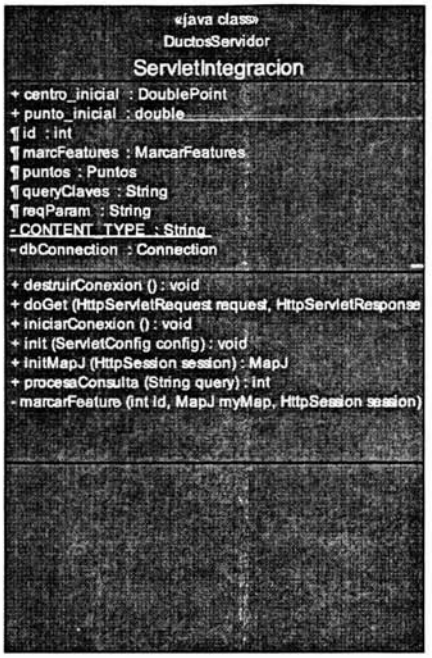

«java class»
DuctosSerializable
informacion_Pnto
¶ infPnto : Vector
+ Get_Id () : String
+ getINF () : Vector
+ guaradarINF (String inf) : void
+ informacion_Pnto ()

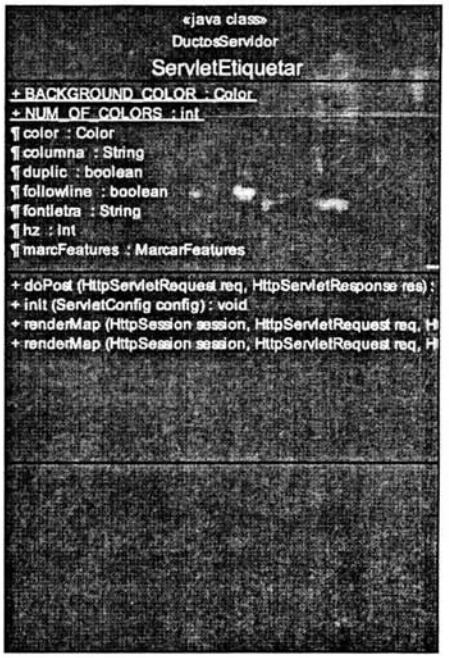

«java class»
DuctosCliente
PanelBDI
¶ Periodicidad : JComboBox
¶ Periodicidad1 : JComboBox
¶ Periodo1 : JComboBox
¶ Periodo2 : JComboBox
¶ Unidad : JComboBox
¶ aplet : MapperApplet
¶ año1 : JComboBox
¶ año2 : JComboBox
¶ capaBDI : String
+ PanelBDI (MapperApplet aplet_)
+ getCapaBDI (Arbol arbol, String nomcapa) : String
+ limpia () : void
+ llenarCombo (String nombreCapa) : void
+ llenarOpcs () : void
+ tablaActivos () : void
+ tablaEdos () : void
+ tablaRegiones () : void
+ xmlActivos () : void

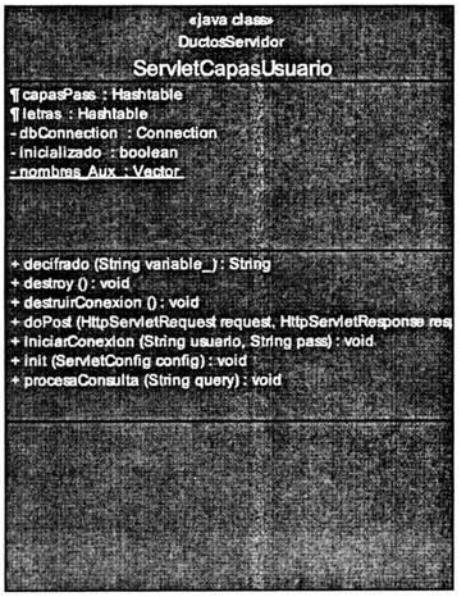

«java class»
DuctosCliente
Panel_Color
banner : JLabel
tcc : JColorChooser
¶ diagetiqs : Dialog_Etiquetar
¶ newColor : Color
+ Panel_Color ()
+ stateChanged (ChangeEvent e) : void
- jbInit () : void

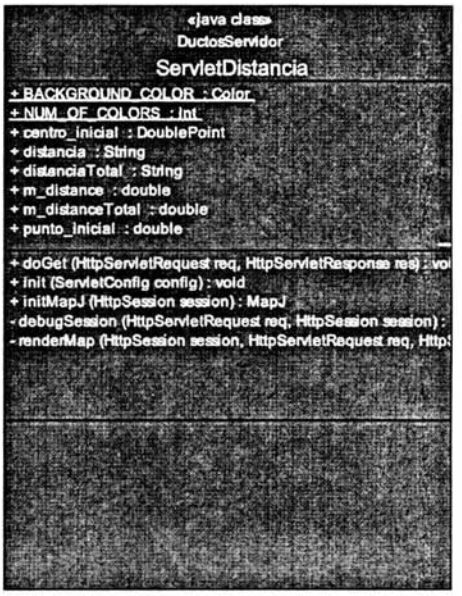

«java class»
DuctosCliente
Panel_Print
¶ bg : Color
¶ button : JButton
¶ dash1 : floatf1
¶ dashed : BasicStroke
¶ fg : Color
¶ red : Color
¶ stroke : BasicStroke
¶ white : Color
¶ wideStroke : BasicStroke
+ Panel_Print ()
+ actionPerformed (ActionEvent e) : void
+ drawShapes (Graphics2D g2) : void
+ paintComponent (Graphics g) : void
+ print (Graphics g, PageFormat pf, int pi) : int

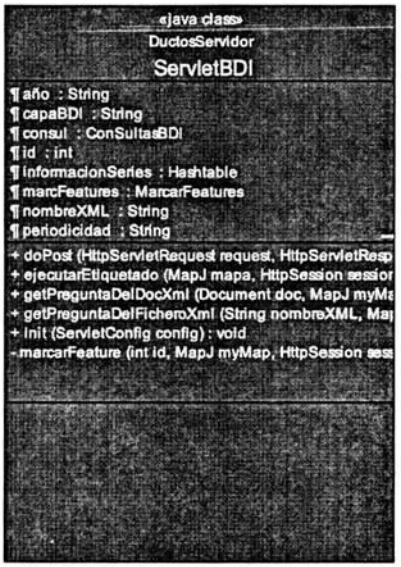

«java class»
DuctosSerializable
Nodo
- capa_bdi : String
- dueno : String
- grafico : String
- indicePadre : int
- isFetch : boolean
- isVisible : boolean
- nombre : String
- nombre_Aux : String
+ Nodo (int indicePadre, String nombre, String nombre_Aux)
+ getCapa_BDI () : String
+ getDueno () : String
+ getGrafico () : String
+ getIconURL () : URL
+ getIndicePadre () : int
+ getIsFetch () : boolean
+ getIsVisible () : boolean

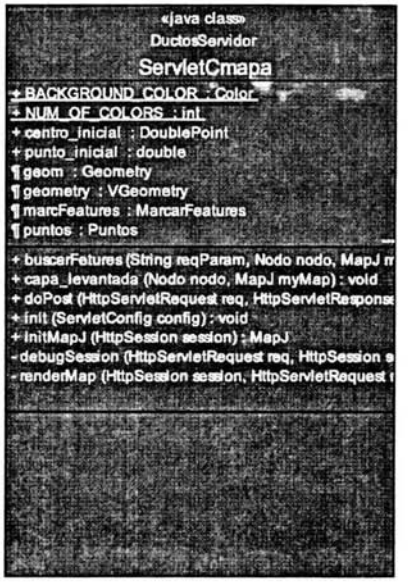

«java class»
DuctosServidor
ServletIntegracion
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ id : int
¶ marcFeatures : MarcarFeatures
¶ puntos : Puntos
¶ queryClaves : String
¶ reqParam : String
- CONTENT_TYPE : String
- dbConnection : Connection
+ destruirConexion () : void
+ doGet (HttpServletRequest request, HttpServletResponse
+ iniciarConexion () : void
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
+ procesaConsulta (String query) : int
- marcarFeature (int id, MapJ myMap, HttpSession session)

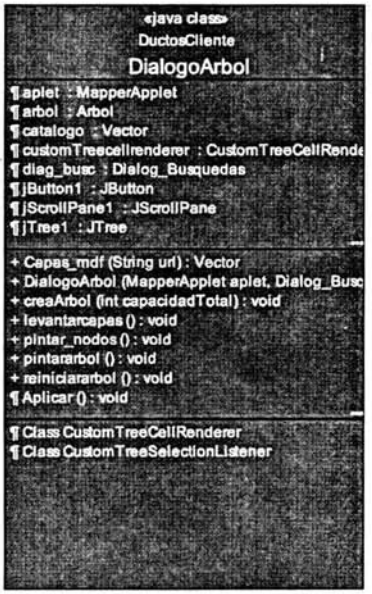

«java class»
DuctosServidor
ServletEtiquetar
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
¶ color : Color
¶ columna : String
¶ duplic : boolean
¶ followline : boolean
¶ fontletra : String
¶ hz : int
¶ marcFeatures : MarcarFeatures
+ doPost (HttpServletRequest req, HttpServletResponse res) :
+ init (ServletConfig config) : void
+ renderMap (HttpSession session, HttpServletRequest req, H
+ renderMap (HttpSession session, HttpServletRequest req, H

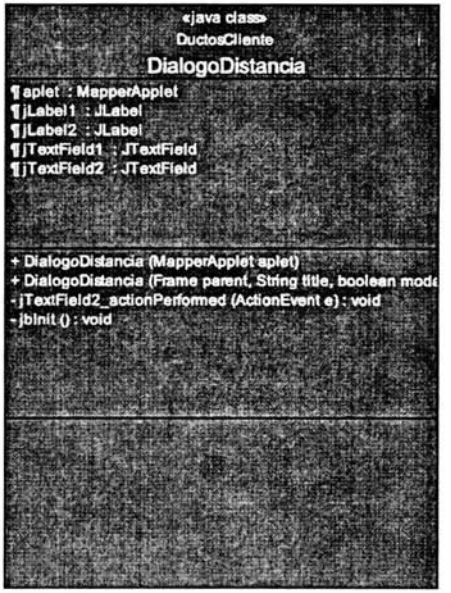

«java class»
DuctosServidor
ServletCapasUsuario
¶ capasPass : Hashtable
¶ letras : Hashtable
- dbConnection : Connection
- inicializado : boolean
- nombres_Aux : Vector
+ decifrado (String variable_) : String
+ destroy () : void
+ destruirConexion () : void
+ doPost (HttpServletRequest request, HttpServletResponse res
+ iniciarConexion (String usuario, String pass) : void
+ init (ServletConfig config) : void
+ procesaConsulta (String query) : void

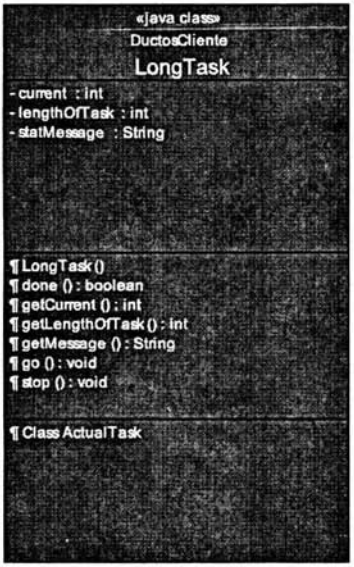

«java class»
DuctosServidor
ServletDistancia
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ distancia : String
+ distanciaTotal : String
+ m_distance : double
+ m_distanceTotal : double
+ punto_inicial : double
+ doGet (HttpServletRequest req, HttpServletResponse res) : vo
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
- debugSession (HttpServletRequest req, HttpSession session) :
- renderMap (HttpSession session, HttpServletRequest req, Http

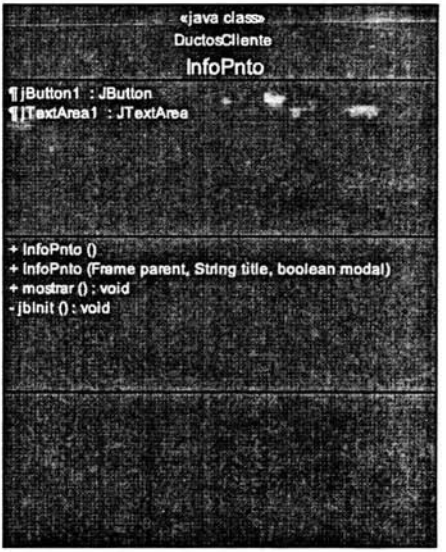

«java class»
DuctosServidor
ServletBDI
¶ año : String
¶ capaBDI : String
¶ consul : ConSultasBDI
¶ id : int
¶ informacionSeries : Hashtable
¶ marcFeatures : MarcarFeatures
¶ nombreXML : String
¶ periodicidad : String
+ doPost (HttpServletRequest request, HttpServletResp
+ ejecutarEtiquetado (MapJ mapa, HttpSession sessior
+ getPreguntaDelDocXml (Document doc, MapJ myMa
+ getPreguntaDelFicheroXml (String nombreXML, Maj
+ init (ServletConfig config) : void
- marcarFeature (int id, MapJ myMap, HttpSession ses

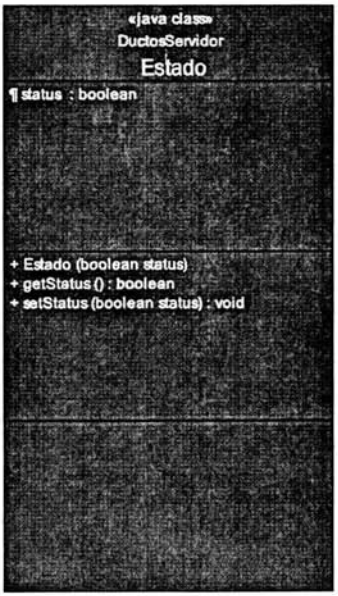

«java class»
DuctosServidor
ServletCmapa
+ BACKGROUND_COLOR : Color
+ NUM_OF_COLORS : int
+ centro_inicial : DoublePoint
+ punto_inicial : double
¶ geom : Geometry
¶ geometry : VGeometry
¶ marcFeatures : MarcarFeatures
¶ puntos : Puntos
+ buscarFetures (String reqParam, Nodo nodo, MapJ m
+ capa_levantada (Nodo nodo, MapJ myMap) : void
+ doPost (HttpServletRequest req, HttpServletRespons
+ init (ServletConfig config) : void
+ initMapJ (HttpSession session) : MapJ
- debugSession (HttpServletRequest req, HttpSession s
- renderMap (HttpSession session, HttpServletRequest

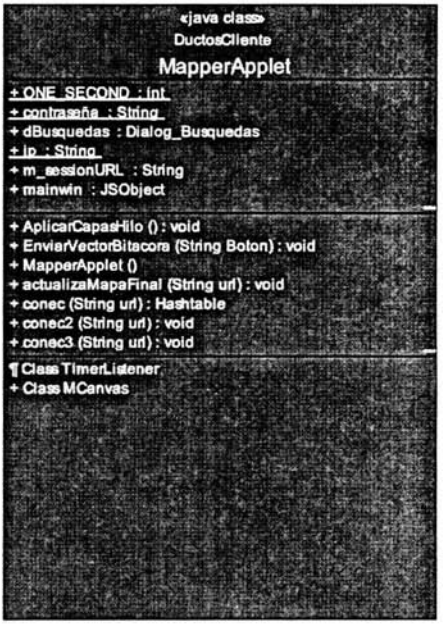

«java class»
DuctosCliente
DialogoArbol
¶ aplet : MapperApplet
¶ arbol : Arbol
¶ catalogo : Vector
¶ customTreecellrenderer : CustomTreeCellRende
¶ diag_busc : Dialog_Busquedas
¶ jButton1 : JButton
¶ jScrollPane1 : JScrollPane
¶ jTree1 : JTree
+ Capas_mdf (String url) : Vector
+ DialogoArbol (MapperApplet aplet, Dialog_Busc
+ creaArbol (int capacidadTotal) : void
+ levantarcapas () : void
+ pintar_nodos () : void
+ pintararbol () : void
+ reiniciararbol () : void
¶ Aplicar () : void
¶ Class CustomTreeCellRenderer
¶ Class CustomTreeSelectionListener

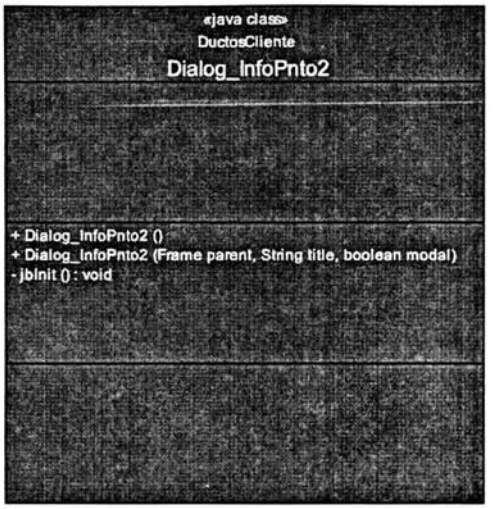

«java class»
DuctosCliente
DialogoDistancia
¶ aplet : MapperApplet
¶ jLabel1 : JLabel
¶ jLabel2 : JLabel
¶ jTextField1 : JTextField
¶ jTextField2 : JTextField
+ DialogoDistancia (MapperApplet aplet)
+ DialogoDistancia (Frame parent, String title, boolean mod
- jTextField2_actionPerformed (ActionEvent e) : void
- jbInit () : void

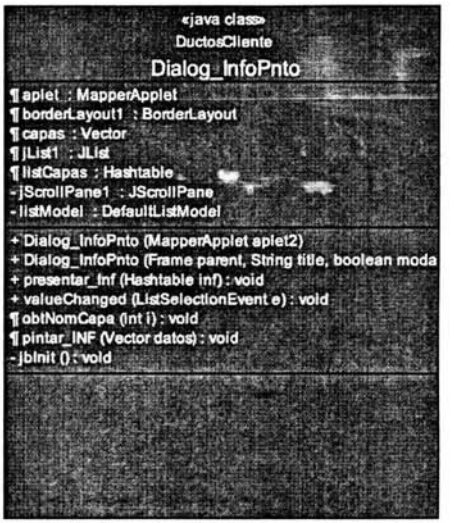

«java class»
DuctosCliente
LongTask
- current : int
- lengthOfTask : int
- statMessage : String
¶ LongTask ()
¶ done () : boolean
¶ getCurrent () : int
¶ getLengthOfTask () : int
¶ getMessage () : String
¶ go () : void
¶ stop () : void
¶ Class ActualTask

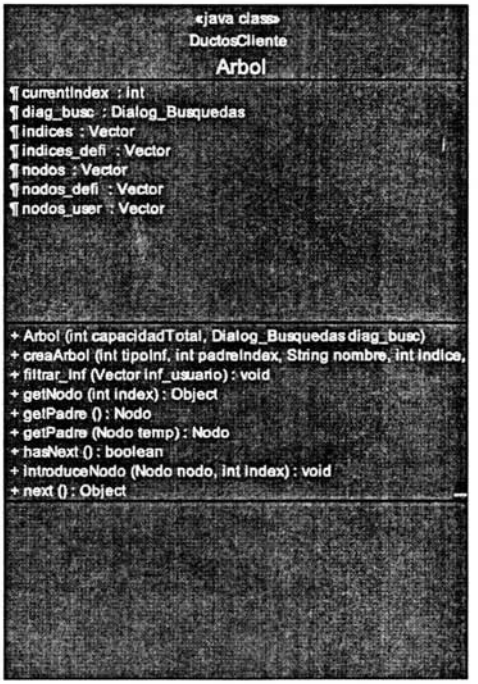

«java class»
DuctosCliente
InfoPnto
¶ jButton1 : JButton
¶ jTextArea1 : JTextArea
+ InfoPnto ()
+ InfoPnto (Frame parent, String title, boolean modal)
+ mostrar () : void
- jbInit () : void

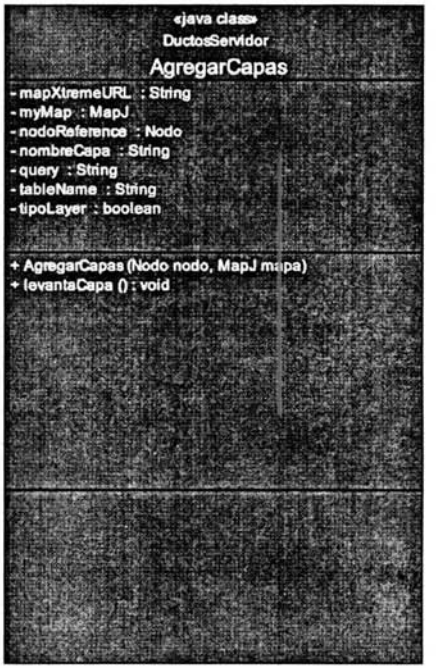

«java class»
DuctosServidor
Estado
¶ status : boolean
+ Estado (boolean status)
+ getStatus () : boolean
+ setStatus (boolean status) : void

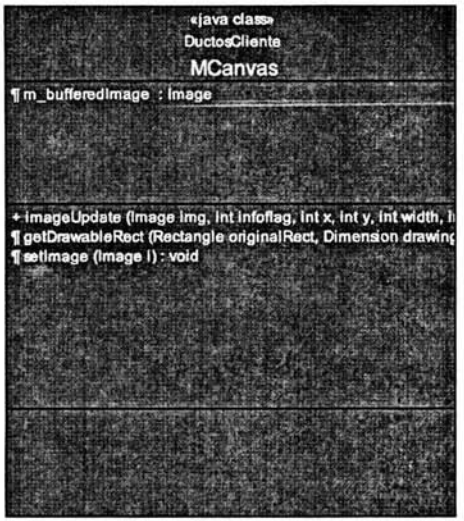

«java class»
DuctosCliente
MapperApplet
+ ONE_SECOND : int
+ contraseña : String
+ dBusquedas : Dialog_Busquedas
+ ip : String
+ m_sessionURL : String
+ mainwin : JSObject
+ AplicarCapasHilo () : void
+ EnviarVectorBitacora (String Boton) : void
+ MapperApplet ()
+ actualizaMapaFinal (String url) : void
+ conec (String url) : Hashtable
+ conec2 (String url) : void
+ conec3 (String url) : void
¶ Class TimerListener
+ Class MCanvas

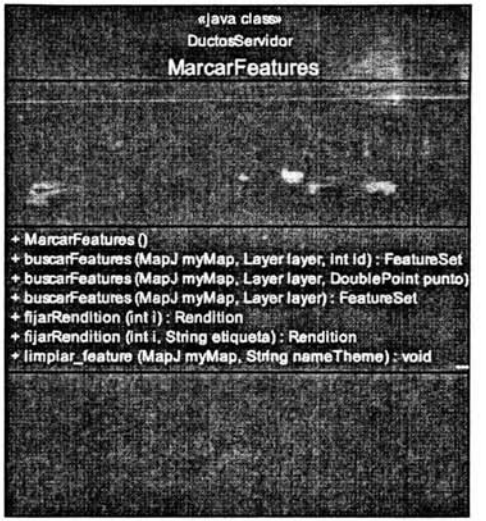

«java class»
DuctosCliente
Dialog_InfoPnto2
+ Dialog_InfoPnto2 ()
+ Dialog_InfoPnto2 (Frame parent, String title, boolean modal)
- jbInit () : void

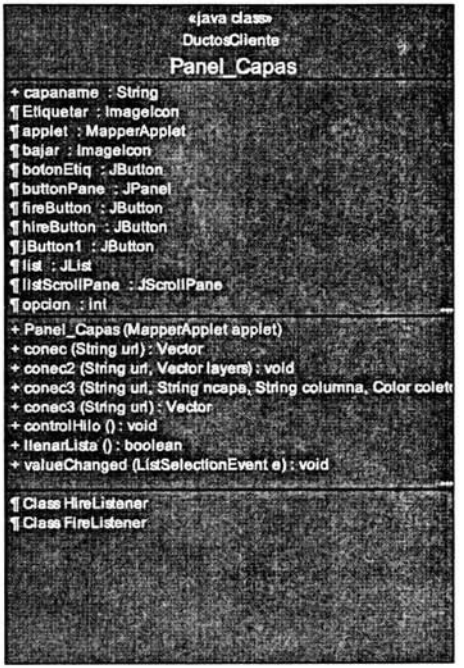

«java class»
DuctosCliente
Dialog_InfoPnto
¶ apIet_ : MapperApplet
¶ borderLayout1 : BorderLayout
¶ capas : Vector
¶ jList1 : JList
¶ listCapas : Hashtable
- jScrollPane1 : JScrollPane
- listModel : DefaultListModel
+ Dialog_InfoPnto (MapperApplet apIet2)
+ Dialog_InfoPnto (Frame parent, String title, boolean moda
+ presentar_Inf (Hashtable inf) : void
+ valueChanged (ListSelectionEvent e) : void
¶ obtNomCapa (int i) : void
¶ pintar_INF (Vector datos) : void
- jbInit () : void

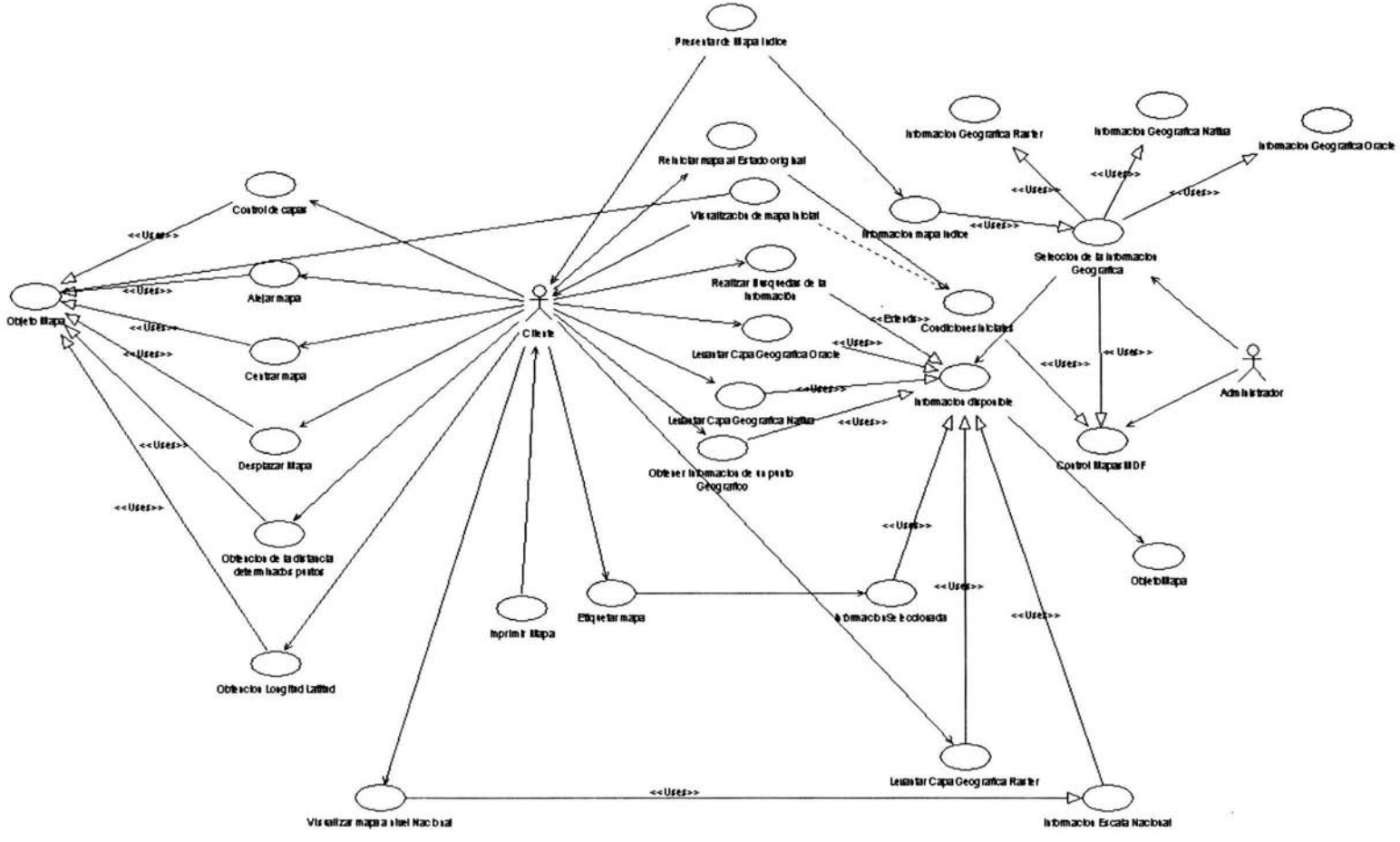

«java class»
DuctosCliente
Arbol
¶ currentIndex : int
¶ diag_busc : Dialog_Busquedas
¶ indices : Vector
¶ indices_defi : Vector
¶ nodos : Vector
¶ nodos_defi : Vector
¶ nodos_user : Vector
+ Arbol (int capacidadTotal, Dialog_Busquedas diag_busc)
+ creaArbol (int tipoInf, int padreIndex, String nombre, int indice,
+ filtrar_Inf (Vector inf_usuario) : void
+ getNodo (int index) : Object
+ getPadre () : Nodo
+ getPadre (Nodo temp) : Nodo
+ hasNext () : boolean
+ introduceNodo (Nodo nodo, int index) : void
+ next () : Object


«java class»
DuctosServidor
AgregarCapas
- mapXtremeURL : String
- myMap : MapJ
- nodoReference : Nodo
- nombreCapa : String
- query : String
- tableName : String
- tipoLayer : boolean
+ AgregarCapas (Nodo nodo, MapJ mapa)
+ levantaCapa () : void


«java class»
DuctosCliente
MCanvas
¶ m_bufferedImage : Image
+ imageUpdate (Image img, int infoflag, int x, int y, int width, I
¶ getDrawableRect (Rectangle originalRect, Dimension drawing
¶ setImage (Image i) : void


«java class»
DuctosServidor
MarcarFeatures
+ MarcarFeatures ()
+ buscarFeatures (MapJ myMap, Layer layer, int id) : FeatureSet
+ buscarFeatures (MapJ myMap, Layer layer, DoublePoint punto)
+ buscarFeatures (MapJ myMap, Layer layer) : FeatureSet
+ fijarRendition (int i) : Rendition
+ fijarRendition (int i, String etiqueta) : Rendition
+ limpiar_feature (MapJ myMap, String nameTheme) : void


«java class»
DuctosCliente
Panel_Capas
+ capaname : String
¶ Etiquetar : ImageIcon
¶ applet : MapperApplet
¶ bajar : ImageIcon
¶ botonEtiq : JButton
¶ buttonPane : JPanel
¶ fireButton : JButton
¶ hireButton : JButton
¶ jButton1 : JButton
¶ list : JList
¶ listScrollPane : JScrollPane
¶ opcion : int
+ Panel_Capas (MapperApplet applet)
+ conec (String url) : Vector
+ conec2 (String url, Vector layers) : void
+ conec3 (String url, String ncapa, String columna, Color colet
+ conec3 (String url) : Vector
+ controlHilo () : void
+ llenarLista () : boolean
+ valueChanged (ListSelectionEvent e) : void
¶ Class HireListener
¶ Class FireListener


Presentar de Mapa Indice
Relacionar mapa al Estado original
Visualizacion de mapa inicial
Realizar Busquedas de la Informacion
Levantar Capa Geografica Oracle
Levantar Capa Geografica Native
Obtener Informacion de un punto Geografico
Etiquetar mapa
Imprimir Mapa
Visualizar mapa a nivel Nacional
Levantar Capa Geografica Raster
Informacion Escala Nacional
Informacion Seleccionada
Informacion mapa indice
Informacion Geografica Raster
Informacion Geografica Native
Informacion Geografica Oracle
Seleccion de la Informacion Geografica
Condiciones iniciales
Informacion disponible
Administrador
Control Mapas MDF
Objeto Mapa
Control de capas
Alejar mapa
Centrar mapa
Desplazar Mapa
Obtencion de la distancia determinados puntos
Obtencion Longitud Latitud
Cliente
ObjetoMapa
<<Uses>>
<<Extends>>

REFERENCIAS BIBLIOGRÁFICAS

The Apache Group, "Apache HTTP Server Project".
http://www.apache.org/

7. Toro, I., G. Arias, J. Duque, M.Ramírez, M. Romero, G. Roveda, C. Sánchez y C. Terán., 1996. Búsqueda de un modelo para el manejo sostenible de los ecosistemas de la Mojana. En: Licania arborea,. Año 1, N¼ 1. Pags: 32-36

Almeida, A. S.; Bettini, C. Curso de Geoestatística Aplicada UFRJ, 1994. Apostila.

M.Crovella, A. Bestavros, Explaining WWW Traffic Self-Similarity, August,1995.
http://www.cs.bu.edu/techreports/95-015-explaining-web-self-similarity.ps.Z

A. Luotonen, K. Altis,"World Wide Web Proxies", Conferencia WWW4.
http://pigeon.elsevier.nl/www94/PrelimProcs.html

J. Nielsen, "Multimedia and Hypertext", Academic Press, San Diego,
http://www.useit.com/jakob/mmhtbook.html

J. Palme, A. Hopmann, N. Shelness, "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)" RFC 2110 http://www.ietf.org/internet-drafts/draft-ietf-mhtml-rev-07.txt

Saborío J. Materiales de prácticas de cursos de Sistemas de información Geográfica con IDRISI, impartidos en el CATIE de 1987 a 1992, en la UCR de 1990 - 1992 y en el ICE de 1993 a 1995, en CIDIAT en 1996.

Eastman R. y otros. 1993 IDRISI User's guide and Technical Reference. Universidad de Clark

Eastman R. y otros. 1993 IDRISI Update Manual, v.4.1 Universidad de Clark

ARANOFF, S., 1993 Geographic Information Systems: A Management Perspective, WDL Publications, Ontario, 294 p.

BURROUG P. A. 1983 Principles of Geographical Information Systems for Land Resources Assessment. CLARENDON PRESS, OXFORD, 194 p.

DEPARTMENT OF AGRICULTURE. 1986 Soil Conservation Service, Cartography and Geographic Information Systems Division. Geographic Information Systems (GIS) United States Washigton, D.C.

GIS WORLD. 1989 News of Geographic Information System Technology in Land, Natural Resources, & Urban Information Management, Ft. Collins, Colorado.

GUEVARA, A. 1987 Guía para la Implementación de un SIG para la Planificación regional y nacional. Environmental Systems Research Institute, California.

LAURINI R., THOMPSON D., 1994 Fundamentals of Spatial Information Systems. Academic Press, Mariland, 680 p.

MARTIN, D.S. 1993 Applications in Coastal Zone Research and Management, vol.3. United Nations Institute for Training and Research, UNITAR European Office, Switzerland, 141 p.

EASTMAN, J.R. y otros. 1992 Technical Reference IDRISI, v.4.1. Clark University, Massachusetts.

EASTMAN, J.R. y otros. 1992 User's Guide IDRISI, v.4.1.Clark University, Massachusetts.

IICA. 1986 La Tecnología de los Sistemas de Información Espacial. Seminario. Escuela de Ciencias Geográficas - Instituto Interamericano de Cooperación para la Agricultura., Costa Rica.

OREAMUNO R. Y SABORIO, J. 1988 Producción Mapas de Erosión. Memorias 3er Seminario Recursos Hidráulicos. , CFIA, Costa Rica.

SABORIO J., 1992 Implementación de un SIG Ambiental para Centro América, Informe final. Proyecto CAM 90/013, PNUD, Guatemala, 150 p.

Spatial Information System Cadcorp SIS ASC , Computer Aided Development Corporation(Cadcorp) Ltd. SIS V5.2Getting Started

Julio 2002 MapInfo Corporation MapInfo MapXtreme
Java Edition 4.5
Troy , NY

Addison Wesley . (2000) Creación de sitios Web con XML y Java
Hiroshi Maruyama Kent Tamura Nahokio Uramoto

John B Dawson(2001) Oracle Corporation
XML:Develop Applications

ALFAOMEGA (2003) Oracle 9i
Administración y Analisis de Bases de Datos
Cesar Perez

Anaya Multimedia 2002 MCGraw-Hill
Oracle 9i Desarrollo Web
Bradley D. Brown

PERSON EDUCACION (2001) Seguridad en Java Edición Especial
Jaime Jaworski
Paul J. Perrone

Copyright 1999 Rational Software Object-Oriented
Análisis and Design Student Additional
Information Appendix
Version 4.2

Copyright 1999 Rational Software Object-Oriented
Análisis and Design Student Additional
Payroll Requirements
Document version 4.2

Octubre (2002-2003) Boletin Mensual SICORI Sistema Corporativo de Información
Geografica