UNIVERSIDAD DE EL SALVADOR
FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICA
ESCUELA DE MATEMÁTICA

"DISEÑO DE UNA ARQUITECTURA PARA LA IMPLEMENTACION DE INTERFACES MULTIPLATAFORMA DE SIG BASADA EN ESTANDARES OGC".

PRESENTADO POR:

ALBA ELIZABETH FLORES VILLALOBOS
MARIO ALEJANDRO SÁNCHEZ GARCÍA

PARA OPTAR AL GRADO DE:

LICENCIADO EN ESTADÍSTICA

ASESORES:

ING. FRANCISCO JOSÉ DELGADO OLIVARES
ING. KELLY XIOMARA AGUILAR

CIUDAD UNIVERSITARIA, FEBRERO DEL 2007

UNIVERSIDAD DE EL SALVADOR

RECTORA:

Dra. María Isabel Rodríguez

SECRETARIA GENERAL:

Licda. Alicia Margarita Rivas de Recinos.

FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICA

DECANO EN FUNCIONES:

M.Sc. José Héctor Elías Díaz

SECRETARIO:

Licda. Martha Noemí Martínez de Rosales

ESCUELA DE MATEMÁTICA

DIRECTOR:

Lic. Mauricio Hernán Lovo Córdova

TRABAJO DE GRADUACIÓN APROBADO POR:

COORDINADOR:

Lic. Mauricio Hernán Lovo Córdova

ASESOR:

Ing. Kelly Xiomara Aguilar.

ASESOR ADJUNTO:

Ing. José Francisco Delgado Olivares

Con sabiduría se edificará la casa,
Y con prudencia se afirmará;
Y con ciencia se llenarán las cámaras
De todo bien preciado y agradable.
El hombre sabio es fuerte,
Y de pujante vigor el hombre docto.
Porque con ingenio harás la guerra,
Y en la multitud de consejeros está la victoria.

Proverbios 24: 3-6

DEDICATORIA

Escribir los agradecimientos de esta tesis significa haber concluído un intenso trabajo realizado en colaboración con muchos profesionales que nos han apoyado durante el tiempo que hemos dedicado a la realización de este proyecto. Trabajar junto a ellos ha sido para nosotros un gran honor, y vaya de antemano nuestro agradecimiento a todos.

Quisieramos que estas palabras no sean consideradas como una simple formalidad para cumplir un requisito impuesto por la costumbre, sino un sentimiento de profundo reconocimiento a todas aquellas personas que nos han ayudado y han podido hacer realidad esta Tesis.

En primer lugar agradezco a Dios, por la sabiduría que me brindó, y acompañarme siempre en este largo camino, ya que sin él nada de esto hubiera sido posible.

A mi familia, por el apoyo incondicional que me ha brindado, especialmente a mi madre que siempre me enseñó el bien y la perseverancia en el trabajo, y a mis dos seres amados que ya no están, porque sé que desde el cielo siempre me dieron su bendición. No tengo palabras más que el reconocer que mi mayor deuda es con mi familia, por su aliento a mis estudios, pues sin su apoyo y constante comprensión jamás habría terminado.

Alba Elizabeth Flores Villalobos.

Le agradezco a Dios, por darme la vida, por estar en todo momento conmigo iluminando mí camino, por poner en mi camino a personas maravillosas así como también por las bendiciones y regalos que recibo cada día.

A mi familia a quien agradezco de todo corazón por su amor, cariño y comprensión.

Mario Alejandro Sánchez García.

CAPITULO I
ANÁLISIS Y DETERMINACIÓN DE REQUERIMIENTOS

ÍNDICE

CAPITULO II
ANÁLISIS Y DETERMINACIÓN DE REQUERIMIENTOS

INDICE

CAPÍTULO III
DISEÑO DE LA ARQUITECTURA

INDICE

CAPÍTULO III
DISEÑO DE LA ARQUITECTURA

CAPÍTULO IV
IMPLEMENTACIÓN DE LA ARQUITECTURA

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE DE FIGURAS

ANEXOS

CAPÍTULO I
ESTUDIO PRELIMINAR

1.1 INTRODUCCIÓN

Los Sistemas de Información Geográfica (*SIG*) se han constituido durante los últimos años en una de las mas importantes herramientas de trabajo para investigadores, analistas y planificadores, etc., en todas sus actividades que tienen que ver con el manejo de la información (Bases de Datos) relacionada con diversos tópicos espaciales o territoriales, lo cual esta creando la necesidad de que estos usuarios de información espacial conozcan acerca de esta tecnología. Aunque algunos Sistemas de Información Geográfica tienen gran capacidad de análisis, estos no pueden existir por si mismos, deben tener una organización, personal y equipamiento responsable para su implementación y sostenimiento, adicionalmente este debe cumplir un objetivo y estar garantizados los recursos para su mantenimiento.

Al principio, los programas de SIG se diseñaron como un Sistema de Información original y sencillo en su concepción; se trataba de una herramienta que proporcionaba una aplicación de elaboración de mapas y una Base de Datos, ambos de lo más tradicional.

En los últimos años, los SIG se han convertido en verdaderos Sistemas de Información, se han beneficiado de los avances de los Sistemas Gestores de Bases de Datos (SGBD) y de los inicios de la normalización de la información cartográfica digital, así como de los lenguajes de consulta que son de gran utilidad en la recuperación y análisis de la información.

En la actualidad, existen diferentes empresas que desarrollan software SIG las cuales incorporan los estándares de ***Open GIS Consortium***[1]. El problema es que la mayoría de estos software son de licencia comercial.

[1] *Organización conformada por las principales empresas desarrolladoras de SIG, también se conoce como OGC (Open GIS Consortium).*

En el desarrollo de este trabajo se investigó las características del estándar creado por ***OGC***[2] que permite integrar fuentes de datos SIG y Bases de Datos Relacionales, así como también se indagó y evaluó los distintos Gestores de Bases de Datos que soportan este tipo de integración.

Se realizó un estudio sobre las distintas herramientas que existen actualmente, con las cuales se puede llevar a cabo la integración de las fuentes de datos SIG y las Bases de Datos Relacionales; desarrollándola a un bajo costo.

Luego, tras la investigación y evaluación de las herramientas se comenzó la construcción del diseño de la arquitectura el cual se desarrolló a partir de las herramientas seleccionadas en el apartado anterior.

Finalizada la etapa anterior se implementó una interfaz multiplataforma de SIG basada en Estándares OGC en el área de Meteorología del Servicio Nacional de Estudios Territoriales (SNET).

Es por ello que el objetivo principal de este trabajo fue investigar y evaluar las opciones presentes que existen sobre las distintas herramientas que sirven para diseñar e implementar la arquitectura de sistemas que permita construir aplicaciones que integren con facilidad fuentes de datos de SIG y Bases de Datos Relacionales, valiéndose de ***Estándares Open GIS***[3], de manera que las organizaciones que lo requieran puedan implementar soluciones que integren estas dos fuentes de información en aplicaciones orientadas a usuarios finales y no hacia especialistas de SIG.

[2] *Open GIS Consortium*
[3] *Formato estándar sobre SQL con soporte de geometrías, orientada al uso remoto mediante servidores de Base de Datos Relacionales*

1.2 DEFINICIÓN DEL PROBLEMA

En El Salvador la implementación de los SIG en las diferentes Instituciones u Organizaciones se ha incrementado, debido a la utilidad que estos proporcionan en las diferentes áreas de investigación tales como Cartografía, Catastro, Vulcanología, Meteorología, entre otras.

Uno de los problemas básicos y comunes que se tiene en el campo de los SIG es la forma de almacenamiento de la Información Geográfica, que tienen las diferentes Instituciones u Organizaciones en el país, ya que existen diferentes formatos para almacenar datos geográficos.

Esto surge debido a que dichas instituciones utilizan diferentes programas SIG, lo cual lleva a una proliferación de información dispersa con diferentes formatos en cada máquina; lo que generalmente dificulta la creación de un repositorio de datos, que pueda consultarse con facilidad y a una ardua tarea de actualización de los mismos.

Actualmente muchos de los datos de un SIG son almacenados en sistemas de archivos o Bases de Datos con un formato específico y sólo a través del respectivo SIG es posible consultar y analizar estos datos. Como consecuencia, los usuarios no pueden utilizar un solo producto SIG para tener acceso a todos los datos que deseen, lo que resulta en un alto costo si se adquiere alguno o varios productos SIG. La situación anterior, es con frecuencia bastante problemática, pero hoy en día existen muchas herramientas de conversión; además, la mayoría de herramientas de SIG modernas ofrecen acceso a los datos, la mayoría en modo lectura, en su formato nativo aunque sean de otro fabricante.

Lo anterior nos lleva a tener “islas” desintegradas y descoordinadas de información con estructuras y sistemas de codificación poco o nada compatibles entre sí.

Además otro problema que podemos describir es con respecto al tipo de archivo que se genera ya que la organización de la información en sistemas de archivos, con frecuencia ocasiona un nivel de duplicación innecesario y dificultades para la localización de los datos a través de toda una red de computadoras.

Otro problema que se tiene es el tiempo que se requiere para conocer y entender algún producto específico, además del tiempo y los recursos que se utilizan para instalar y configurar cada producto en diferentes plataformas de trabajo.

En nuestro país, el Servicio Nacional de Estudios Territoriales, ***SNET,*** es una de las entidades responsables de producir Información Geográfica, por lo que el uso de los SIG para esta Institución es de gran utilidad en áreas como Meteorología, Vulcanología, Geología e Hidrología.

Para desarrollar sus actividades, SNET se basa en la recopilación, manejo y producción de información, y por lo tanto, requiere de un amplio soporte tecnológico-informático. Es por ello que cuenta con una Unidad de Servicios Informáticos (USI), la cual es la encargada del manejo de la información proporcionada por las diferentes áreas que la constituyen.

Esta como otras instituciones que manejan información espacial, es uno de los tantos ejemplos en que se vuelve importante el estandarizar la Información Geográfica que allí se produce, allí se nos hizo posible el poder investigar los problemas antes mencionados.

1.3 ANÁLISIS DE LA PROBLEMÁTICA

Para corroborar la problemática en la cual se basa este trabajo de graduación, se realizó una entrevista dirigida al jefe de la unidad USI de la institución SNET, utilizando como instrumento de recolección de información, el cuestionario.

A través de dicha entrevista se identificaron diversos factores referentes al tratamiento de la información y a las herramientas que utilizan para su manejo y almacenamiento, tales como:

- Equipos.
- Sistemas Operativos
- Software.

Primero comenzaremos hablando sobre los tipos de equipo que posee dicha institución en los que cabe destacar que poseen servidores, computadoras personales así como también estaciones de trabajo y computadoras portátiles.
En cuanto a los Sistemas Operativos, se determinó que utilizan diferentes tipos entre los que se destacan:

- Windows
  - Windows 98
  - Windows XP
  - Windows 2000
  - Windows 2003 Server
  - Windows ME
  - Windows NT

- Unix
  - Sun Solaris
  - Hp-Ux
- Linux
  - Mandrake
  - Debian
  - Red Hat
  - Suse

En cuanto a las herramientas relacionadas con el manejo y construcción de un software SIG, SNET cuenta con muchas licencias libres y comerciales como por ejemplo: ArcView, ArcExplorer, ArcEditor, ArcInfo, TatukGis, ILWIS, Netbeans, Oracle, DB2, MySQL, etc.

Para el almacenamiento de la información, en un principio utilizaron sistemas de archivos, pero se observó que tenían la desventaja de una proliferación de información, es decir, duplicidad de archivos con el mismo contenido en los discos duros; por ejemplo en el caso de los datos meteorológicos, o geológicos, que cambian continuamente, no es posible crear un nuevo mapa por cada modificación.

Pero existe una segunda forma de almacenamiento que decidieron utilizar, y es en Bases de Datos Relacionales, esta tiene la ventaja que la información modificada puede ser almacenada en un mismo ***Repositorio.***[4]

Al utilizar las Bases de Datos Relacionales surge la necesidad de integrar dichas Bases de Datos con los SIG y de esta manera obtener el máximo provecho de los Sistemas de Información Geográfica, es decir, incrementar la posibilidad de convertir un Sistema SIG en un verdadero ***Sistema Multiusuario***[5]**.**

[4] *Lugar donde se guarda o deposita toda la información por medio de Bases de Datos Relaciones*
[5] *Capacidad de un sistema de gestionar al mismo tiempo, información para varios usuarios.*

Cabe mencionar que SNET, trabaja con distintas organizaciones externas a la institución que ayudan en las diferentes áreas, entre estas: La Facultad de Ciencias Naturales y Matemática de la Universidad de El Salvador (UES), la cual cuenta con una unidad de SIG, en la que se desarrollan diversas investigaciones que aportan información de gran utilidad a SNET como es el área de vulcanología.

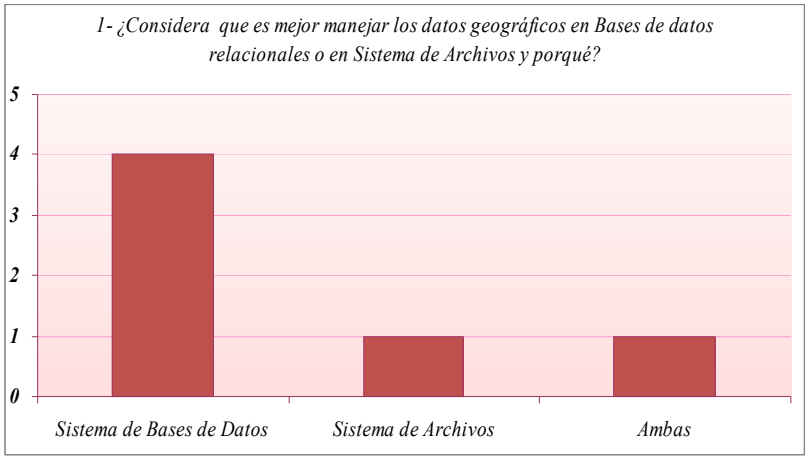
Otro de los problemas existentes, es que algunas de estas herramientas de software SIG, resultan ser bastante complicadas para ***Usuarios Promedio***[6], pues las tareas que ellos necesitan implican tanto operaciones de SIG como operaciones de Bases de Datos, y además requieren el enlace con formas más avanzadas de despliegue.

Existen diferentes causas o motivos por los que resulta necesario incorporar la Información Geográfica a una Base de Datos Relacional, utilizando los Estándares OpenGIS Consortium definidos para tal fin.
Para sustentar más la necesidad de incorporar la Información Geográfica a una Base de Datos Relacional se realizó una entrevista dirigida a las organizaciones que más utilizan los Sistemas de Información Geográfica, tales como:

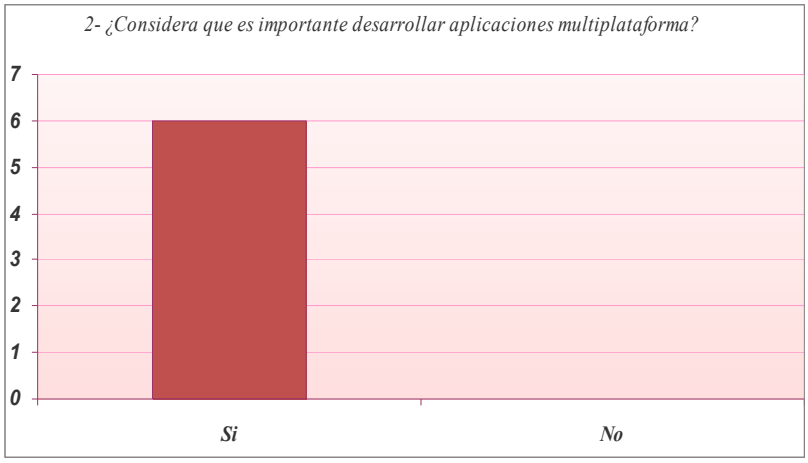
- Ministerio de Medio Ambiente (MARN).
- Centro Nacional de Registro (CNR).
- Ministerio de Obras Públicas (Sección VMVDU) [7].
- Ministerio de Agricultura y Ganadería (MAG).
- Dirección General de Estadísticas y Censos (DIGESTYC).
- Oficina de Planificación del Área Metropolitana de San Salvador (OPAMSS).

[6] *Son aquellos que necesitan realizar solamente unas pocas operaciones que involucran interfaces de mapas.*
[7] *Viceministerio de Vivienda y Desarrollo Urbano.*

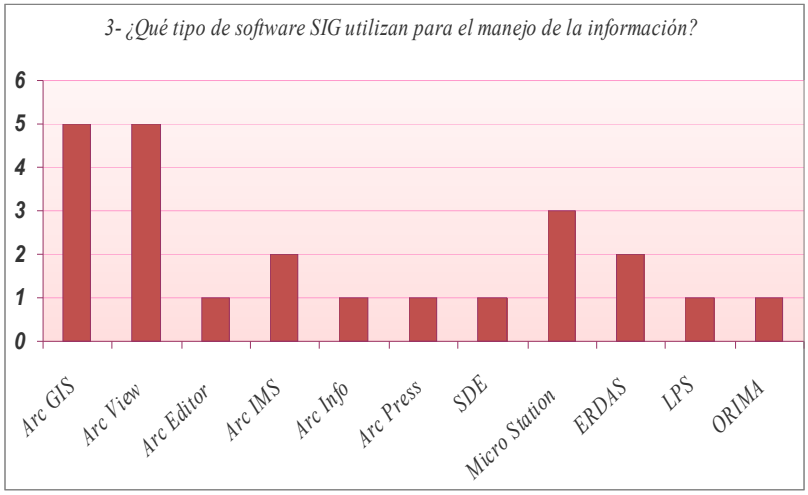
Los resultados fueron los siguientes:



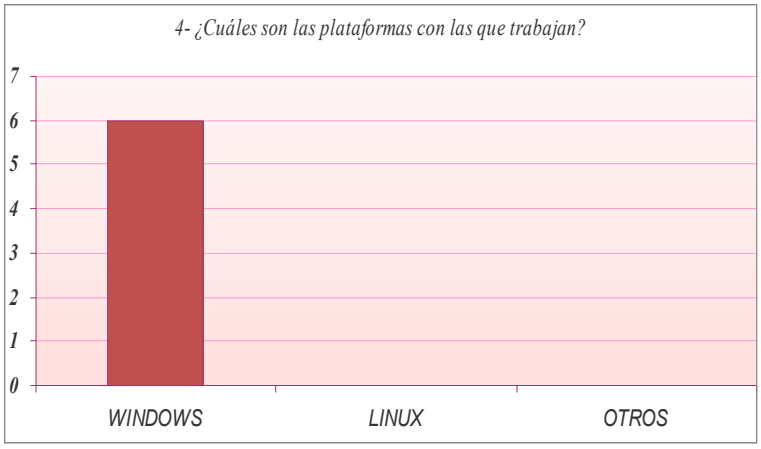
Para las personas que consideran que es mejor trabajar con Bases de Datos Relacionales las razones son las siguientes: información más organizada, permite trabajar con querys y fácil manejo de la información.



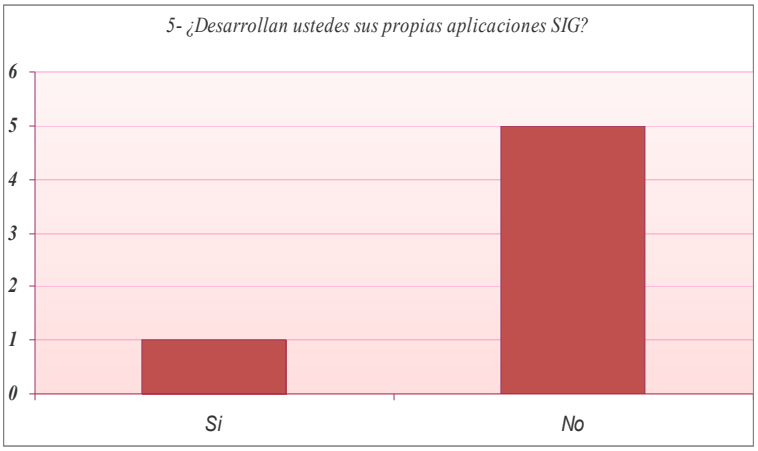
Todas las personas entrevistadas consideran que es importante ya que permitiría la compatibilidad con diversas plataformas, no se encontrarían amarrados a una sola, y serían libres de utilizar la que quisieran.



Podemos observar que la mayoría de personas entrevistadas utilizan los software Sig comerciales propietarios de ESRI



Todas las personas entrevistadas nos dijeron que solamente se trabaja bajo la plataforma de Windows, ya que se dificulta el trabajo con otras plataformas por no tener gente especializada.



La gran mayoría de las personas entrevistadas no desarrollan sus propias aplicaciones, ya que son usuarios finales, es decir, personas que utilizan un SIG para la toma de desiciones.

En las preguntas 6, 7 y 8 (Ver Anexo 2), la mayoría desconocía la existencia del Estándar o solo tenían una pequeña idea de lo que era, pero todos coincidían que es necesario tener estandarizada la información.

En la pregunta 9 (Qué ventajas piensa que tiene el desarrollo de éstas aplicaciones orientadas a usuarios finales frente a los software SIG comerciales?), se mencionaron las siguientes:

- Bajo costo
- Se haría no de una manera general, si no más personalizada, y de más fácil manejo.
- Sería más sencillo manipular la información.

Al preguntarle si estarían dispuestos a utilizar esta arquitectura, contestaron lo siguiente: De existir dicha arquitectura, todos estarían dispuestos a utilizarla.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Investigar y evaluar las opciones presentes que existen sobre las distintas herramientas que sirven para diseñar e implementar la arquitectura de sistemas que permita construir aplicaciones que integren con facilidad fuentes de datos SIG y Bases de Datos Relacionales, valiéndose de estándares (Open GIS Consortium), de manera que las organizaciones que lo requieran puedan implementar soluciones que integren estas dos fuentes de información en aplicaciones orientadas a usuarios finales y no hacia especialistas de SIG.

1.4.2 OBJETIVOS ESPECÍFICOS

- Investigar las características del estándar creado por OGC que permita integrar fuentes de datos SIG y Bases de Datos Relacionales.

- Indagar y evaluar sobre las características de los distintos Gestores de Bases que permitan el almacenamiento de información espacial en Bases de Datos Relacionales.

- Examinar y evaluar las características de los software de aplicación que permitan la construcción de aplicaciones SIG multiplataformas.

- Desarrollar una arquitectura para la creación de herramientas SIG que permita realizar: búsqueda, visualización y análisis de Información Geográfica. Que el uso de dichas herramientas, sean más flexibles y menos complicadas para los usuarios finales (no especialistas en SIG).

- Crear e implementar una interfaz que se pueda ejecutar en las distintas plataformas con que cuenta el área de Meteorología de la Institución SNET que cumpla con los requerimientos de estándares Open GIS Consortium.

- Presentar el informe final de la investigación con la debida documentación y haciendo énfasis en los software y herramientas de desarrollo para el diseño y la implementación de la arquitectura.

1.5 GENERALIDADES DE LOS SIG

Sistemas de Información Geográfica (SIG)

Existen diferentes definiciones de Sistema de Información Geográfica, una manera de definirlo es la siguiente:

"Una disciplina basada en conocimientos, metodologías y procedimientos asistidos por computadora, que permiten la incorporación, almacenamiento, manipulación, procesamiento, consulta y presentación de información referenciada geográficamente en formatos gráficos y no gráficos". [8]

Los Sistemas de Información en muchas formas se asemeja a un programa de Base de Datos, ya que analiza y relaciona información almacenada bajo la forma de registros, pero con una diferencia crucial: Cada registro en una Base de Datos SIG contiene información usada para dibujar formas (normalmente un punto, una línea, o un polígono).
Cada una de esas formas representa un lugar único sobre la Tierra al cual le corresponden los datos.

Un SIG almacena y despliega información numérica y alfanumérica, acerca de objetos o lugares que tienen una ubicación espacial.

Por esta razón, un SIG es más que una herramienta para dibujar mapas (sin tener en cuenta que puede realizar esta tarea extremadamente bien): es en realidad un sistema para mapeo y análisis de la distribución geográfica de los datos. Y se entiende por datos a toda aquella información que pueda ser almacenada en la Base de Datos, así como relacionada con una localidad.

[8] *Definido por el Sistema Corporativo de Información Geográfica de PEMEX (SICORI 2001).*

1.6 COMPONENTES DE UN SIG

Los Datos Geográficos no son los únicos componentes de un SIG como se puede ver en la ***Figura 1***. Los componentes de un Sistema de Información Geográfica se describen a continuación:

Figura 1: Componentes de un SIG

Hardware: Conjunto de equipos físicos empleados en el almacenamiento y procesamiento de los datos contenidos en el sistema.

Datos: Posiblemente el componente más importante de un SIG son los datos geográficos y los datos relacionados con estos. Un SIG integra datos espaciales con otros recursos de datos que podrán ser almacenados y administrados con ***SGBD.***[9]

Personal: La tecnología de los SIG es de limitado valor sin la gente que administra al sistema y desarrolla aplicaciones para resolver problemas del mundo real.

Software: Conjunto de programas que proporcionan las funciones y herramientas necesarias para almacenar analizar y desplegar Información Geográfica.

[9] *Sistema Gestor de Bases de Datos*

Los SIG son herramientas cada vez más empleadas para cualquier campo de aplicación. El componente más importante para un SIG es la información. Se requiere de adecuados datos de soporte para que el SIG pueda resolver los problemas y contestar a preguntas de la forma más acertada posible. Se maneja un gran volumen de información, lo cuál hace necesario una forma de almacenamiento, es decir, las Bases de Datos Geográficas, las cuales forman una parte importante del SIG**.**

Los programas de SIG proveen las funciones y las herramientas necesarias para almacenar, analizar y desplegar la Información Geográfica. Los principales componentes de los programas son:

- Herramientas para la entrada y manipulación de la Información Geográfica.
- Sistema Gestor de Base de Datos (SGBD).
- Interfaz gráfica para el usuario final para acceder fácilmente a las herramientas.

1.6.1 HERRAMIENTAS PARA LA ENTRADA Y MANIPULACIÓN DE LA INFORMACIÓN GEOGRÁFICA.

La función de un software es de proveer de una base funcional que sea adaptable y extensible de acuerdo con los requerimientos propios de cada organización.
Además, es el encargado de crear, manipular, y administrar la información contenida para que se garantice el funcionamiento analítico del SIG.

Dentro de las funciones básicas de un Sistema de Información podemos describir la captura de la información, esta se logra mediante procesos de digitalización, procesamiento de imágenes de satélite, fotografías, videos, entre otros.

Entre los diferentes métodos o formas de entrada de información de un software SIG se encuentran:

- Por medio de Escaners.
- Por medio de Ploteadores.
- Por medio de ***Tabletas Digitalizadoras***[10]

[10] Dispositivo destinado a la digitalización, consiste en una superficie (tablero) y un cursor, que permite la entrada de coordenadas para localizar los elementos de un mapa.

1.6.2 SISTEMA GESTOR DE BASES DE DATOS (SGBD)

Una Base de Datos es una colección organizada de datos. Existen diversas estrategias para organizar datos y facilitar el acceso y la manipulación. Un Sistema Gestor de Bases de Datos ***(SGBD)*** proporciona los mecanismos para almacenar y organizar datos de una manera consistente con el formato de la Base de Datos.

El Sistema Gestor de Bases de Datos es la porción más importante del software de un Sistema de Base de Datos. Un SGBD es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

El objetivo primordial de un Sistema Gestor Base de Datos es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer, almacenar y manipular información de la Base de Datos. Todas las peticiones de acceso a la base, se manejan centralizadamente por medio del SGBD, por lo que este paquete funciona como interfaz entre los usuarios y la Base de Datos.

Las funciones principales de un SGBD son:

- Crear y organizar la Base de Datos.
- Establecer y mantener las trayectorias de acceso a la Base de Datos de tal forma que los datos puedan ser accesados rápidamente.
- Manejar los datos de acuerdo a las peticiones de los usuarios.

- Registrar el uso de las Bases de Datos.

- Interacción con el Gestor de Archivos.

  Esto a través de las sentencias en ***DML***[11] al comando del sistema de archivos. Así el Gestor de Base de Datos es el responsable del verdadero almacenamiento de los datos.

- Respaldo y recuperación.

  Consiste en contar con mecanismos implantados que permitan la fácil recuperación de los datos en caso de ocurrir fallas en el Sistema de Base de Datos.

- Control de concurrencia.

  Consiste en controlar la interacción entre los usuarios concurrentes para no afectar la inconsistencia de los datos.

- Seguridad e integridad.

  Consiste en contar con mecanismos que permitan el control de la consistencia de los datos evitando que estos se vean perjudicados por cambios no autorizados o previstos.

Los Gestores de Bases de Datos son también conocidos como DBMS.

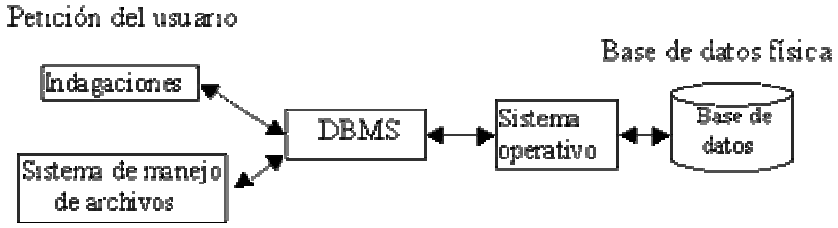


Figura 2: El SGBD

[11] *Lenguaje de Manipulación de Datos.*

La ***Figura 2*** muestra el SGBD como interfaz entre la Base de Datos física y las peticiones del usuario. El SGBD interpreta las peticiones de entrada/salida del usuario y las manda al Sistema Operativo para la transferencia de datos entre la unidad de memoria secundaria y la memoria principal.
En sí, un Sistema Gestor de Base de Datos es el corazón de la Base de Datos ya que se encarga del control total de los posibles aspectos que la puedan afectar.

1.6.2.1 ADMINISTRADOR DE BASES DE DATOS

Denominado por sus siglas como: ***DBA, Database Administrator.***

Es la persona encargada y que tiene el control total sobre el Sistema de Base de Datos, sus funciones principales son:

- ***Definición de Esquema:***
  Es el esquema original de la Base de Datos se crea escribiendo un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.

- ***Definición de la estructura de almacenamiento del método de acceso.***
  Estructuras de almacenamiento y de acceso adecuados se crean escribiendo un conjunto de definiciones que son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.

- ***Concesión de autorización para el acceso a los datos.***

  Permite al administrador de la Base de Datos regular las partes de las Bases de Datos que van a ser accedidas por varios usuarios.

- ***Especificación de limitantes de integridad.***

  Es una serie de restricciones que se encuentran almacenados en una estructura especial del sistema que es consultada por el Gestor de Base de Datos cada vez que se realice una actualización al sistema.

1.6.2.2 USUARIOS DE LAS BASES DE DATOS

Podemos definir a los usuarios como toda persona que tenga todo tipo de contacto con el Sistema de Base de Datos desde que este se diseña, elabora, termina y se usa.

Los usuarios que accesan a una Base de Datos pueden clasificarse como:

- ***Programadores de Aplicaciones.***

  Los profesionales en computación que interactúan con el sistema por medio de llamadas en DML (Lenguaje de Manipulación de Datos), las cuales están incorporadas en un programa escrito en un Lenguaje de Programación (Por ejemplo, COBOL, PL/I, Pascal, C, Java, etc.)

- ***Usuarios Sofisticados.***

  Los usuarios sofisticados interactúan con el sistema sin escribir programas. En cambio escriben sus preguntas en un lenguaje de consultas de Base de Datos.

- ***Usuarios Especializados.***

  Algunos usuarios sofisticados escriben aplicaciones de Base de Datos especializadas que no encajan en el marco tradicional de procesamiento de datos.

- ***Usuario Final.***

  Los usuarios no sofisticados interactúan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente en el Sistema de Base de Datos, podemos mencionar que el usuario final utiliza el Sistema de Base de Datos sin saber nada del diseño interno del mismo por ejemplo: un cajero.

1.6.2.3 ESTRUCTURA GENERAL DEL SISTEMA

Un Sistema de Base de Datos se encuentra dividido en módulos cada uno de los cuales controla una parte de la responsabilidad total de sistema. En la mayoría de los casos, el Sistema Operativo proporciona únicamente los servicios más básicos y el sistema de la Base de Datos debe partir de esa base y controlar además el manejo correcto de los datos. Así el diseño de un Sistema de Base de Datos debe incluir la interfaz entre el Sistema de Base de Datos y el Sistema Operativo.

Los componentes funcionales de un Sistema de Base de Datos, son:

- ***Gestor de Archivos***

  Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar información.

- ***Gestor de Bases de Datos***

  Sirve de interfaz entre los datos y los programas de aplicación.

- ***Procesador de Consultas.***

  Traduce las proposiciones en lenguajes de consulta a instrucciones de bajo nivel. Además convierte la solicitud del usuario en una forma más eficiente.

- ***Compilador de DDL.***

  Convierte las proposiciones ***DDL***[12] en un conjunto de tablas que contienen metadatos, estas se almacenan en el diccionario de datos.

- ***Archivo de datos.***

  En él se encuentran almacenados físicamente los datos de una organización.

- ***Diccionario de datos.***

  Contiene la información referente a la estructura de la Base de Datos.

- ***Índices.***

  Permiten un rápido acceso a registros que contienen valores específicos.

Una forma gráfica de representar los componentes antes mencionados y la relación que existe entre ellos sería la siguiente. (***Figura 3***)

[12] *DLL (Dynamic Linking Library) significa Bibliotecas de Enlace Dinámico, término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda del programa por parte del Sistema Operativo.*

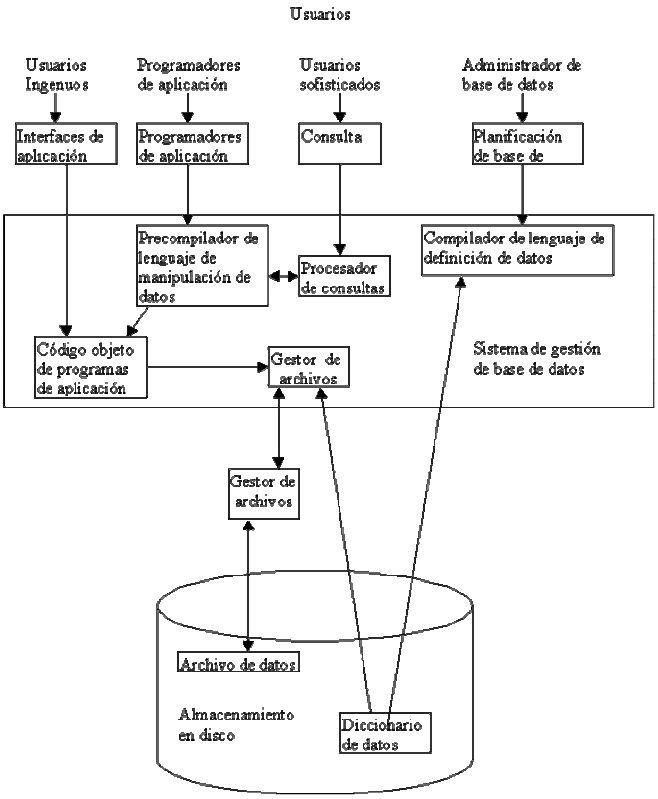


Figura 3: Componentes funcionales de un Sistema de Base de Datos

1.6.3 INTERFAZ GRÁFICA PARA EL USUARIO FINAL PARA ACCEDER FÁCILMENTE A LAS HERRAMIENTAS

La interfaz de usuario es aquella que permite al usuario elegir comandos y otras opciones utilizando representaciones visuales como íconos, menús, las listas de elementos del menú; y todos aquellos medios que permitan la comunicación entre el hombre y la computadora.

Con el uso de una interfaz gráfica se evita que el usuario realice procedimientos rudimentarios, es decir usuarios que no podrían realizar una consulta complicada que implique seleccionar campos en diferentes tablas de una Base de Datos; lo que se le facilitaría con la implementación de una interfaz amigable.

Entre las funciones principales de la interfaz gráfica podemos mencionar:

- Manipulación de la información.
  Aquí, se provee los mecanismos para la comunicación entre los datos físicos (extraídos por los procesos de almacenamiento de la información y los procesos de análisis de dicha información).

- Extracción de la información.
  Las formas de extraer o recuperar información son muy variadas y pueden llegar a ser muy complejas.

- Edición de la información.

Permite la modificación y actualización de la información. Las funciones deben incluir: mecanismos para la edición de entidades gráficas (cambio de color, posición, escala, dibujo de nuevas entidades gráficas, entre otros.) así como también, mecanismos para la edición de datos descriptivos (modificación de atributos, cambios en la estructura de archivos, actualización de datos, generación de nuevos datos, entre otros.)

1.7 MODELOS GEOMÉTRICOS

Existen dos modelos para representar datos espaciales dentro de un Mapa Geográfico, los cuales son:

- Modelo Vectorial.
- Modelo Raster.

1.7.1 MODELO VECTORIAL

El modelo vectorial es una estructura de datos utilizada para almacenar Datos Geográficos. Los datos vectoriales constan de líneas o arcos, definidos por sus puntos de inicio y fin, y puntos donde se cruzan varios arcos, los nodos. La localización de los nodos y la estructura topológica se almacena de forma explícita. Las entidades quedan definidas por sus límites solamente y los segmentos curvos se representan como una serie de arcos conectados. El almacenamiento de los vectores implica el almacenamiento explícito de la topología, sin embargo solo almacena aquellos puntos que definen las entidades y todo el espacio fuera de éstas no está considerado.

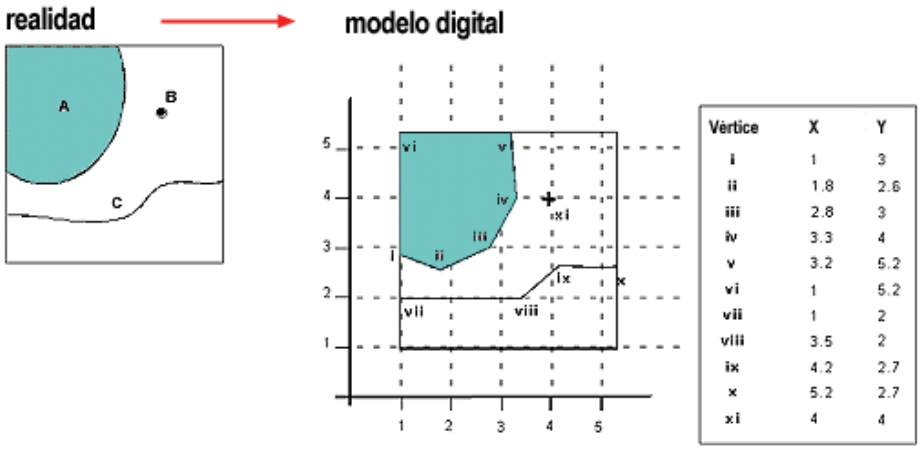
Un SIG vectorial se define por la representación vectorial de sus Datos Geográficos. De acuerdo a las peculiaridades de este modelo de datos, los objetos geográficos se representan explícitamente y, junto a sus características espaciales, se asocian sus valores temáticos.

Hay dos formas de organizar esta Base de Datos doble (espacial y temática). Normalmente, los sistemas vectoriales tienen dos componentes: uno que almacena los datos espaciales y otro los datos temáticos. A éste se le denomina sistema de organización híbrido, por unir una Base de Datos Relacional, para los aspectos temáticos, con una Base de Datos topológica, para los geográficos.

Un elemento clave en este tipo de sistemas es el **identificador** de cada objeto. Éste es único y diferente para cada objeto y permite la conexión entre ambas Bases de Datos.

REPRESENTACIÓN VECTORIAL DE LOS DATOS

En el modelo de datos vectorial (***Figura 4***), los Datos Geográficos se representan en forma de coordenadas. Las unidades básicas de Información Geográfica en los datos vectoriales son puntos, líneas (arcos) y polígonos. Cada una de éstas se compone de uno o más pares de coordenadas, por ejemplo, una línea es una colección de puntos interconectados, y un polígono es un conjunto de líneas interconectadas.



Vértice	X	Y
i	1	3
ii	1.8	2.6
iii	2.8	3
iv	3.3	4
v	3.2	5.2
vi	1	5.2
vii	1	2
viii	3.5	2
ix	4.2	2.7
x	5.2	2.7
xi	4	4

Figura 4: Modelo de Datos Vectorial

Coordenada

Pares de números que expresan las distancias horizontales a lo largo de ejes ortogonales, o tríos de números que miden distancias horizontales y verticales, o n-números a lo largo de n-ejes que expresan una localización concreta en el espacio n-dimensional. Las coordenadas generalmente representan localizaciones de la superficie terrestre relativas a otras localizaciones.

Punto

Abstracción de un objeto de cero dimensiones representado por un par de coordenadas X, Y. Normalmente un punto representa una entidad geográfica demasiado pequeña para ser representada como una línea o como una superficie; por ejemplo, la localización de un edificio en una escala de mapa pequeña.

Línea

Conjunto de pares de coordenadas ordenados que representan la forma de entidades geográficas demasiado finas para ser visualizadas como superficies a la escala dada (curvas de nivel, ejes de calles, o ríos), o entidades lineales sin área (límites administrativos). Una línea es sinónimo de arco.

Polígono

Entidad utilizada para representar superficies. Un polígono se define por las líneas que forman su contorno y por un punto interno que lo identifica. Los polígonos tienen atributos que describen al elemento geográfico que representan.

Los elementos integrantes de un SIG vectorial son un Sistema de Gestión de Bases de Datos (SGBD) para los atributos temáticos, y un sistema que gestiona las relaciones topológicas. En algunos paquetes SIG, el SGBD está basado en un software ya existente como p.ej. dBASE.

MODELO ENTIDAD – RELACIÓN

En este enfoque se consideran tres elementos: **(a)** Las **Entidades**, es decir, los objetos que son relevantes para la Base de Datos a elaborar. En un SIG lo integra cualquier hecho que pueda ser localizado espacialmente. **(b)** Los **Atributos** o características asociadas a cada entidad. Cada atributo tiene un dominio de valores posibles, por ejemplo, el estado de una carretera puede ser *malo, regular, bueno,* o *muy bueno*. **(c)** Las **Relaciones** o mecanismos que permiten relacionar unas entidades con otras. Algunos ejemplos son: "situado en", "incluido en", "cruzarse con", etc.

Por lo general las Bases de Datos utilizadas en SIG son de tipo relacional. En una Base de Datos Relacional, los datos se almacenan en tablas en las que las filas se refieren a los objetos o entidades y las columnas a los atributos temáticos o variables asociados. Normalmente una Base de Datos se compone de muchas tablas cuya interrelación es posible a través de un identificador común que es único para cada entidad. La mayoría de las Bases de Datos de los SIG tienen dos variables con identificadores, uno de ellos es único y correlativo, puede ser numérico o alfanumérico, y el segundo puede repetirse y ayuda a organizar la tabla de atributos.

Las ventajas de utilizar este tipo de Base de Datos son:

- El diseño se basa en una metodología con fundamentos teóricos importantes, lo que ofrece mayor confianza en su capacidad de evolucionar.

- Es muy fácil de implementar, sobre todo en comparación con los otros modelos como el jerárquico y en red.

- Es muy flexible. Las nuevas tablas se pueden añadir fácilmente.

- Por último, existen muchos SGBD potentes que usan este enfoque, dotados de lenguajes de consulta (como SQL) que facilitan incluir este instrumento en cualquier SIG. De este modo, algunos SIG comerciales incluyen SGBD preexistentes.

1.7.2 MODELO RASTER

El modelo raster es un método para el almacenamiento, el procesado y la visualización de Datos Geográficos. Cada superficie a representar se divide en filas y columnas, formando una malla o rejilla regular. Cada celda ha de ser rectangular, aunque no necesariamente cuadrada. Cada celda de la rejilla guarda tanto las coordenadas de la localización como el valor temático.

La localización de cada celda es implícita, dependiendo directamente del orden que ocupa en la rejilla, a diferencia de la estructura vectorial en la que se almacena de forma explícita la topología.

Las áreas que contienen idéntico atributo temático son reconocidas como tal, aunque las estructuras raster no identifican los límites de esas áreas como polígonos en sí.

Los datos raster son una abstracción de la realidad, representan ésta como una rejilla de celdas o píxeles (***ver figura 5***), en la que la posición de cada elemento es implícita según el orden que ocupa en dicha rejilla. En el modelo raster el espacio no es continuo sino que se divide en unidades discretas. Esto le hace especialmente indicado para ciertas operaciones espaciales como por ejemplo las superposiciones de mapas o el cálculo de superficies.

Las estructuras raster pueden implicar en ocasiones un incremento del espacio de almacenamiento, ya que almacenan cada celda de la matriz sin tener en cuenta si se trata de una entidad o simplemente de un espacio “vacío”.

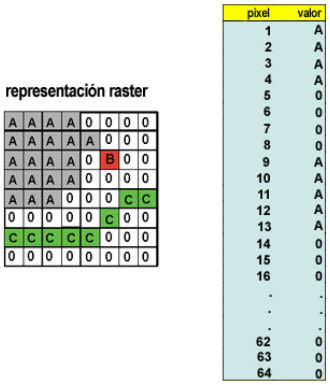


Figura 5: Modelo de Datos Raster

Ventajas

- Es una estructura de datos simple.

- Las operaciones de superposición de mapas se implementan de forma más rápida y eficiente.

- Cuando la variación espacial de los datos es muy alta el formato raster es una forma más eficiente de representación.

- El método raster es requerido para un eficiente tratamiento y realce de las imágenes digitales.

Desventajas

- La estructura de datos raster es menos compacta. Las técnicas de compresión de datos pueden superar frecuentemente este problema.

- Ciertas relaciones topológicas son más difíciles de representar.

- La salida de gráficos resulta menos estética, ya que los límites entre zonas tienden a presentar la apariencia de bloques en comparación con las líneas suavizadas de los mapas dibujados a mano. Esto puede solucionarse utilizando un número muy elevado de celdas más pequeñas, pero entonces pueden resultar ficheros inaceptablemente grandes.

CAPÍTULO II

ANÁLISIS Y DETERMINACIÓN DE REQUERIMIENTOS

2.1 METODOLOGÍA PARA EVALUACIÓN DE HERRAMIENTAS A UTILIZAR.

En el capítulo anterior, se describió acerca de los aspectos relacionados con los Sistemas de Información Geográfica. Además se describieron los principales componentes de un programa SIG, los cuales son:

- Las herramientas para la entrada y manipulación de la Información Geográfica.
- Los Sistemas Gestores de Bases de Datos.
- Interfaz Gráfica de Usuario.

En este capítulo, presentamos un análisis sobre las distintas herramientas que se utilizaron para poder diseñar e implementar una arquitectura de sistemas, que permita construir aplicaciones que integren fuentes de datos SIG y Bases de Datos Relacionales, utilizando estándares Open GIS.

Dicho de otra manera, se evaluará las características de los distintos software de aplicación y Gestores de Bases de Datos, que permitan construir aplicaciones con interfaces orientadas a usuarios finales.

Antes de realizar nuestros análisis definiremos algunos conceptos básicos relacionados con los software y Gestores de Bases de Datos:

Código abierto (del inglés ***open source***) es el término por el que se conoce al software distribuido y desarrollado en forma libre.

Desde el punto de vista de una "traducción estrictamente literal", el significado obvio de "código abierto" es que "se puede mirar el código fuente", por lo que puede ser interpretado como un término más débil y flexible que el del software libre. Basado en ello se argumenta que un programa de código abierto puede ser software libre o incluso no libre. Sin embargo, por lo general, un programa de código abierto puede ser y de hecho es software libre.

Código cerrado es el código fuente que no se encuentra disponible para cualquier usuario, es decir no se hace público. Se le llama así en contraposición al código abierto.
El software no libre utiliza un código cerrado.

Software comercial es el software, libre o no, que es comercializado, es decir, que las compañías que lo producen, cobran dinero por el producto, su distribución o soporte.

Software gratis o gratuito (denominado usualmente Freeware) incluye en algunas ocasiones el código fuente; sin embargo, este tipo de software *no es libre* en el mismo sentido que el *software libre*, a menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

Software libre (en inglés ***free software***) es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio de costo de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente.

Software no libre (también llamado **software propietario**, **software privativo**, **software privado** y **software con propietario**) se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o que su código fuente no está disponible o el acceso a éste se encuentra restringido.

Software de dominio público. Este último es aquel por el que no es necesario solicitar ninguna licencia y cuyos derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquel cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es dominio público.

Licencia GPL (*General Public License* o licencia pública general). Está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Licencia BSD. Es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

2.2 CARACTERÍSTICAS DE LOS DISTINTOS GESTORES DE BASES DE DATOS (SGBD).

Existen diversos Gestores de Base de Datos que soportan el estándar de OpenGIS, que son código libre como:

- PostgreSQL (Licencia BSD)
- My SQL (GPL o uso comercial)

Entre otros, pero por la magnitud del tema se investigarán las características de los gestores antes mencionados.

De la misma forma existen SGBD comerciales, como:

- Oracle (Software Propietario), entre otros.

De igual manera únicamente se analizará uno de ellos.

En la siguiente tabla se muestran las características de algunos Gestores de Bases de Datos, para poder elegir el más adecuado.

CARACTERÍSTICAS	ORACLE		POSTGRE SQL		MY SQL	
	SI	NO	SI	NO	SI	NO
MULTIPLATAFORMA	X		X		X	
SEGURIDAD	X		X		X	
SOPORTA EL ESTANDAR OPENGIS	X		X		X	
GRATUITO		X	X		X	
LIBRE		X	X		X	
CODIGO ABIERTO		X	X		X	
VELOCIDAD *	X			X	X	
ROBUSTEZ	X		X		X	
PODEROSA EN MULTICONEXIONES	X		X		X	
VARIEDAD DE TIPOS DE DATOS	X		X		X	
FUNCIONES GEOMÉTRICAS	X		X		X	
TRIGGERS **	X		X			X
ESCALABILIDAD	X		X		X	

Tabla 1: Características de los SGBD

Teniendo en cuenta las ventajas e inconvenientes de los SGBD analizados anteriormente, y los objetivos de este Trabajo de Graduación, podemos llegar a los siguientes razonamientos:

Los tres Gestores que hemos mencionado, poseen las características más importantes que se necesitan para poder trabajar con el manejo de la Información Geográfica. Por lo que queda a criterio del desarrollador, la elección del Gestor más adecuado, de acuerdo a los requerimientos que en su caso necesite.

* Se refiere a la velocidad de inserción de tuplas en una Base de Datos, y consulta en la misma.
** Las versiones actuales (Desde la versión 5.0.2), ya implementan el uso de triggers, aunque están en proceso de prueba.

Para el desarrollo de la aplicación en este trabajo, Oracle no es una buena opción, porque no se cuenta con la licencia y su costo es elevado.

MySQL y PostgreSQL, tienen características muy similares, teniendo en cuenta que si se elige MySQL, se debe elegir las versiones que ya incorporen el uso de triggers que son de suma importancia, porque ayudan a mejorar la administración de la Base de Datos.

Pero tiene el inconveniente que al poseer una Licencia dual, es decir, tanto GPL como comercial, si el desarrollador desea incorporarlo en productos privativos, debe comprar a la empresa una licencia que le permita ese uso, caso contrario con PostgreSQL, que posee la licencia BSD, que le permite incorporarlo tanto en aplicaciones libre o no.

De esta manera se decide trabajar con la extensión para el manejo de la Información Geográfica de PostGreSQL llamada PostGIS.

2.2.1 POSTGRESQL

Existen varias maneras de medir el software: características, desempeño, fiabilidad, soporte y precio.

Características

PostgreSQL posee características de grandes SGBDs comerciales, tales como transacciones, subconsultas, triggers, vistas, integridad referencial con llaves externas, y bloqueo sofisticado. También posee algunas características que no tienen las otras, como tipos definidos por el usuario, herencia, reglas, y control de concurrencia multi-versión para reducir el bloqueo de controversias.

Desempeño

El desempeño de PostgreSQL es comparable con el de otras bases de datos comerciales y de código abierto. Es más rápida para algunas cosas, más lenta para otras.

Fiabilidad

Un SGBD debe ser fiable, o es inútil, PostgreSQL provee versiones estables y sólidas que se encuentran listas para su uso en producción.

Soporte

PostgreSQL proporciona contacto con un gran grupo de desarrolladores y usuarios que ofrecen ayuda para resolver cualquier problema encontrado.

El acceso directo a desarrolladores, la comunidad de usuarios, manuales y el código fuente suelen hacer que el soporte de PostgreSQL sea superior al de otras SGBDs.

Precio

PostgreSQL se encuentra disponible para cualquier uso, ya sea comercial o no.

SGBD Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas.

Altamente Extensible

PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la Base de Datos.

API Flexible

La flexibilidad del ***API***[13] ***(****Interfaz de Comunicación de Aplicaciones****)*** de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el ***RDBMS***[14] PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. En MySQL o Access, en ocasiones en la lectura se tiene que esperar para acceder a información de la Base de Datos. La espera está provocada por usuarios que están escribiendo en la Base de Datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros. Mediante el uso de MVCC, PostgreSQL evita este problema por completo.

[13] *Son aquellos que necesitan realizar solamente unas pocas operaciones que involucran interfaces de mapas.*
[14] *Es un conjunto de especificaciones de comunicación entre componentes software.*

Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

Write Ahead Logging (WAL)

La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la Base de Datos al registro de cambios antes de que estos sean escritos en la Base de Datos. Esto garantiza que en el hipotético caso de que la Base de Datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la Base de Datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la Base de Datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la Base de Datos.

PostgreSQL es el Sistema Gestor de Bases de Datos (SGBD) de código abierto que posibilitó el desarrollo de soluciones corporativas con una mejor relación costo por beneficios. Un punto fuerte de este SGBD es su capacidad de tratar grandes volúmenes de datos con escalabilidad, es decir, su arquitectura puede ser continuamente ampliada de acuerdo con la demanda de los usuarios.

PostgreSQL Es extremadamente modular, facilitando el trabajo de los desarrolladores que desean implementar nuevas funcionalidades. Esta característica posibilitó la creación del modulo ***PostGIS,*** conteniendo incontables funcionalidades para el desarrollo de aplicaciones que tratan la Geoinformación.

2.2.1.1 ANÁLISIS DEL MÓDULO PostGIS

PostGIS es un módulo que añade entidades geográficas a PostgreSQL. Inicialmente, PostgreSQL ya soporta geometrías espaciales, sin embargo PostGIS añade la capacidad de almacenamiento/recuperación según la especificación SFS (Simple Features Specification) del consorcio internacional Open Gis Consortium (OGC). Además del almacenamiento de Datos Geográficos, este módulo también implementa diversas funcionalidades topológicas, posibilitando el desarrollo de Sistemas de Información Geográfica (SIG) Corporativos. La topología también forma parte de la especificación SFS (Open GIS®), garantizando a PostGIS interoperabilidad con incontables sistemas que también adoptan el SFS.

El licenciamiento del PostGIS es definido por la ***GNU***[15] ***GPL*** (General Public License o Licencia Pública General), garantizando todas las libertades de un software libre.

Para tratar grandes volúmenes de datos espaciales con mayor eficiencia, PostGIS implementa la indexación ***RTree***[16] sobre la indexación GiST (Generalized Search Trees) nativa de PostgreSQL.

PostGIS añade el soporte para objetos geográficos a la Base de Datos Objeto-Relacional PostgreSQL. De esta forma podemos decir que PostGIS proporciona la capacidad espacial a un servidor PostgreSQL, permitiendo ser utilizado como un cliente GIS de la Base de Datos.

[15] *Es una licencia creada por la Free Software Foundation a mediados de los 80, y esta orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre.*

[16] *Son estructuras de datos de tipo árbol, que se utilizan para métodos de acceso espacial, es decir, para indexar información multidimensional*

PostGIS más PostgreSQL forman una Base de Datos espacial donde se almacena y gestiona tanto la geometría de los elementos geográficos como los atributos temáticos de los mismos; permite al SGBD orientado a objetos PostgreSQL, la gestión de objetos geográficos.

Además consigue que el servidor de Bases de Datos PostgreSQL pueda manejar objetos geográficos, capacitándolo para funcionar como soporte de datos espaciales en un Sistema de Información Geográfica.

Con PostGIS podemos utilizar todos los objetos que aparecen en las especificaciones Open GIS, como por ejemplo puntos, líneas, polígonos, multilíneas, multipuntos y colecciones geométricas.

Otra de las características que podemos mencionar, es que PostGIS, además de soportar objetos geográficos almacenados en Bases de Datos Relacionales, y estar apegado a estándares OpenGIS, PostGIS tiene definidas diferentes funciones para poder tratar información espacial proveniente de archivos en formato Raster (Archivos Shapefile).

Por todo esto, PostGIS es la solución tecnológicamente hablando más avanzada expuesta en este análisis (velocidad, manejo de gran volumen de datos, información centralizada, etc.).

2.3 CARACTERÍSTICAS DE LOS DISTINTOS LENGUAJES DE PROGRAMACIÓN.

En la siguiente tabla se muestran las características de algunos Lenguajes de Programación, para poder elegir el más adecuado.

CARACTERÍSTICAS	C++		JAVA		VISUAL BASIC.NET	
	SI	NO	SI	NO	SI	NO
SENCILLEZ		X	X		X	
SEGURIDAD *		X	X		X	
ORIENTADO A OBJETOS	X		X		X	
DISTRIBUIDO		X	X		X	
ROBUSTEZ **		X	X		X	
ARQUITECTURA NEUTRAL		X	X			X
THREADS		X	X		X	
VELOCIDAD	X			X	X	
GARBAJE COLLECTION		X	X		X	
LIBRE	X		X			X
MULTIPLATAFORMA	X		X			X

Tabla 2: Características de los Lenguajes de Programación

* Un lenguaje de programación debe estar diseñado para proveer la máxima seguridad, en redes públicas ante virus o posibles invasiones o accesos incorrectos, archivos basura.

** Un lenguaje de programación para que sea robusto, debe de realizar acciones como comprobación de punteros, arrays, gestión de excepciones y errores así como también la verificación del código fuente; todo esto para crear un software altamente confiable.

Una de las características que pone en desventaja al Lenguaje de Programación Visual Basic.net, es que trabaja únicamente bajo la plataforma Windows, y en este proyecto es de gran importancia que funcione para múltiples plataformas, otra de las desventajas que tiene este Lenguaje de Programación, es que no es código libre.

C++ si posee la característica de ser código libre, además de ser un Lenguaje de Programación Orientado a Objetos (POO); pero posee muchas desventajas que hacen que se descarte para este proyecto, como la falta de seguridad, la arquitectura neutral que es importante, pues los programas no son compilados en una sola máquina.

Java es creado de una manera muy similar a C++, con la diferencia que se han corregido muchos errores que se cometían al programar con C++, por ejemplo, Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución, lo cual lo hace dinámico. Esta y otras muchas características que posee Java, dieron la pauta para su elección; por lo que se presentan otras características que dejan más claro el porqué se eligió Java.

2.2.1 JAVA

Las características principales que nos ofrece Java respecto a cualquier otro lenguaje de programación, son:

Sencillez

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.

Seguridad

Los niveles de seguridad que presenta son:

- Fuertes restricciones al acceso a memoria, como son la eliminación de punteros aritméticos y de operadores ilegales de transmisión.

- Rutina de verificación de los *códigos byte* que asegura que no se viole ninguna construcción del lenguaje.

- Verificación del nombre de clase y de restricciones de acceso durante la carga.

- Sistema de seguridad de la interfaz que refuerza las medidas de seguridad en muchos niveles.

Orientado a Objetos

Soporta las tres características propias del paradigma de la orientación a objetos: Encapsulación, herencia y polimorfismo.

Distribuido

Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como HTTP y FTP. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

La verdad es que Java en sí no es distribuido, sino que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

Robusto

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución, lo que hace que se detecten errores lo antes posible, normalmente en el ciclo de desarrollo. Algunas de estas verificaciones que hacen que Java sea un lenguaje robusto son:

- Verificación del *código byte*.
- Gestión de excepciones y errores.
- Comprobación de punteros y de límites de vectores.

Arquitectura Neutral

Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.

Portabilidad

Un programa Java puede ser ejecutado en diferentes entornos, más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo.

Threads

Java permite múltiples hilos (*multithreading*) antes de su ejecución y en tiempo de ejecución. La posibilidad de construir pequeños procesos o piezas independientes de un gran proceso permite programar de una forma más sencilla y es una herramienta muy potente.

Garbaje Collection

La recolección de basura (objetos ya inservibles) es una parte integral de Java durante la ejecución de sus programas. Una vez que se ha almacenado un objeto en el tiempo de ejecución, el sistema hace un seguimiento del estado del objeto, y en el momento en que se detecta que no se va a volver a utilizar ese objeto, el sistema vacía ese espacio de memoria para un uso futuro.

Esta gestión de la memoria dinámica hace que la programación en Java sea más fácil.

Multiplataforma

No es necesario volver a compilar el código para ejecutar el programa en otra máquina. Un solo código funciona para todos los navegadores compatibles con Java o donde se tenga una Máquina Virtual de Java (Mac's, PC's, Sun's, etc).

2.4 CARACTERÍSTICAS DE LAS HERRAMIENTAS DE DESPLIEGUE Y CONSULTA PARA LA PRESENTACIÓN DE INFORMACIÓN GEOGRÁFICA.

Las herramientas de despliegue y consulta para presentación de Información Geográfica, son librerías o componentes que permiten gestionar desde sus aplicaciones, imágenes Vectoriales y/o Raster asociadas a Bases de Datos, para crear Sistemas de Información Geográfica.

Existen muchas y diferentes herramientas de despliegue de Información Geográfica que pueden ser utilizadas para desarrollar trabajos de este tipo, en nuestro trabajo de graduación, se investigó algunas de estas herramientas, y se colocaron aquellas que consideramos tenían más información bibliográfica en Internet, así como también se tomó en cuenta un aspecto muy importante e indispensable para los que buscan desarrollar este tipo de aplicaciones, el cual es el soporte técnico que tiene cada una de estas herramientas, ya que en muchas de ellas el material bibliográfico para ayuda escaso.

Las herramientas que se presentan a continuación son marcas registradas de distintas empresas, MapObjects Java Edition es desarrollada por la empresa Esri, GeoObjects es desarrollada por Blue Marble Geographics y TatukGis desarrollada por TatukGis Company.

En la siguiente tabla se muestran las características de estas Herramientas de Despliegue y Consulta para la presentación de Información Geográfica, para poder elegir la más adecuada.

CARACTERÍSTICAS	MO-JAVA		GEO-OBJECT		TATUK GIS DEVELOPER KERNEL	
	SI	NO	SI	NO	SI	NO
INDEPENDIENTE DEL ENTORNO DE DESARROLLO (IDE)	X			X		X
NUMEROSAS FUNCIONES GIS	X		X		X	
MULTIPLATAFORMA	X			X		X
SOPORTA DIFERENTES FORMATOS DE DATOS SIMULTANEAMENTE	X		X		X	
ANALISIS ESPACIAL	X		X		X	
TAREAS DE GEOCODIFICACION	X		X		X	
LIBRE		X		X		X

Tabla 3: Características de las Herramientas de despliegue

Como ya se mencionó antes los tres componentes o librerías anteriores, se utilizan para poder desarrollar o construir aplicaciones SIG, que satisfagan las necesidades concretas de cada usuario.

Dos de ellas poseen una gran desventaja, que no son multiplataforma lo cual es indispensable en este proyecto, además de no ser independientes del entorno de desarrollo.

En la siguiente tabla, se muestran los formatos soportados de cada una de las librerías o componentes evaluados:

MAPOBJECT JAVA EDITIONS	GeoObjects	TATUK GIS DEVELOPER KERNEL
V E C T O R I A L		
Shapefile (Creacion, visualización y edición)	Blue Marble Layer (BML)	Shapefile
CAD: DFX, DWG, DGN	S-57	MIF\MID
Visualización de coberturas de ArcInfo	Tiger\Line	TAB
Visualización de información SDC (Smart Data Compress)	Native MapInfo Table (DBF)	DXF
Visualización de información VPF (Vector Product Format)	Native MapInfo Interchange Format (MIF)	DGN
Visualización de capas de ArcSDE	Native ESRI Shape (SHP y DBF)	TIGER
Visualización de servicios de Vectores de ArcIMS	Native AutoCAD (DWG y DXF)	GML\KML
	Native Microstations (DNG)	GDF\GPX
R A S T E R		
TIFF\LZW y GeoTiff	TIFF, TIF y GeoTIFF	TIFF, TIF y GeoTIFF
GIF y JPG	JPG y JPEG	JPEG, JPEG2000
PNG	PNG	PNG
DIB	ER Mapper ECW	BMP
BMP	CADRG\ADRG	BIL\SPOT
BIL, BIP, BSQ	BSN\CAP	ECW\ECWP
MrSID	MrSID	MrSID
ERDAS IMAGINE GIS o LAN	PCX	IMG
Formatos Militares NITF y RPF	TGA	DTM
Sun RasterFiles RS, SUN y RAS	BMP	RPF
Catálogo de Imágenes	JP2	SDTF
Servicios de Imágenes de ArcIMS		

Tabla 4: Formatos soportados por las Herramientas de despliegue

Como podemos ver Map Object Java Edition es la librería que más tipos de formato soporta, ya sea en formato vectorial o formato raster.

Otra característica muy importante que se evaluó fue la disponibilidad de información, soporte y ejemplos relacionados con dichas librerías; es muy necesario que exista un buen soporte técnico así como la variabilidad de ejemplos, de esta forma se facilita el trabajo a los programadores para realizar sus aplicaciones.

Map Object posee una buena documentación acerca del uso de sus librerías, además de tener un buen soporte técnico. Geo Object y Tatuk Gis Developer kernel, no tiene desarrollada mucha información que sirva de guía a un programador.

Por ultimo, se decidió evaluar el precio de dichos componentes, dicha información se encuentra en la tabla siguiente:

	N De Licencias	Precio
Map Object Java Edition	1	$ 5,000
Geo Object	1	$ 1,200
TATUK GIS DEVELOPER KERNEL	1	$ 1,490

Tabla 5: Precios de licencias de las Herramientas de despliegue[17]

Esta es una característica en la que podemos observar que Map Object posee los precios mas elevados, pero para el desarrollo de este trabajo, no es un problema, pues ya contamos con la licencia para poder trabajar con las librerías de Map Object la cual ha sido proporcionada por SNET y dado que se nos facilita el uso de esta herramienta porque hay suficiente información que puede ser utilizada para el manejo de la misma, y un soporte técnico que nos ayudaría a aclarar las dudas que se pudiesen presentar al desarrollar la aplicación.

Vale la pena aclarar que para los desarrolladores que quisieran utilizar alguna de estas herramientas, queda a criterio de ellos la elección de la misma tomando en cuenta el precio y la disponibilidad de la empresa para poder comprar la licencia.
Por tanto se trabajará con MapObject Java Edition, el cual se describirá a continuación.

[17] *Estos Precios fueron tomados en las siguientes direcciones, en la fecha indicada.*
www.tatukgis.com (mayo 2006)
www.bluemarblegeo.com (mayo 2006)
www.esri.com/software/mojava/index.html (mayo 2006)

2.4.1 MAPOBJECT JAVA EDITION

MapObject se puede usar para implementar funciones de mapeo en aplicaciones. La gran ventaja de desarrollar un SIG usando MapObject radica en que la aplicación final es independiente de cualquier otra herramienta de desarrollo, dándole total autonomía y versatilidad al sistema. De ahí la importancia de realizar un diseño de código eficiente y legible que permita fácilmente su reutilización o actualización.

MapObjects Java Edition es un conjunto de componentes desarrollados en su totalidad en Java, que permite a los desarrolladores construir aplicaciones SIG multiplataforma que satisfagan las necesidades concretas de cada usuario.

Estos componentes, se agrupan en librerías, y son los mismos componentes que se han utilizado para desarrollar las aplicaciones cliente de **ArcIMS .**

Las principales características de MapObjects Java Edition son:

- Es independiente del entorno de desarrollo utilizado y de la plataforma.

- Permite incluir en los desarrollos, numerosas funciones SIG (cartografía temática con control de simbología y etiquetado de entidades, herramientas de navegación sobre el mapa, medición de distancias, consultas espaciales y alfanuméricas, etc...).

- Las aplicaciones desarrolladas con MapObjects Java, pueden distribuirse tanto en un entorno Desktop como a través de Internet o de una intranet. El desarrollador tiene la posibilidad de elegir en qué nivel de la arquitectura (cliente o servidor), se va a alojar la aplicación, y es posible por tanto, construir servicios de mapas, páginas JSP y servlets o Enterprise JavaBeans personalizadas.

- Las aplicaciones desarrolladas con MapObjects Java Edition, pueden trabajar con múltiples fuentes y formatos de datos de forma simultánea.

- MapObjects Java Edition incorpora numerosas herramientas que permiten construir de forma sencilla una interfaz de usuario que incluya elementos como barras de herramientas, control dinámico de simbología, diálogos de consulta, mapa de vista general, etc...

- MapObjects Java Edition incorpora una documentación muy amplia, y material de referencia en la que se incluye una completa librería de ***JavaBeans***[18]**.**

FUNCIONALIDAD

- Las aplicaciones desarrolladas con MapObjects Java Edition pueden acceder y trabajar de forma simultánea con multitud de formatos de datos, tanto vectoriales como raster.

- Funcionalidad para realizar operaciones geométricas y de análisis espacial: Zonas de influencia, diferencia, unión, intersección, clip, etc.

- Gracias a la posibilidad que proporcionan las clases de MapObjects Java Edition de incorporar la funcionalidad SIG en el lado del servidor, es posible hacer un desarrollo propio que permita servir mapas directamente en Internet.

- MapObjects Java Edition permite incorporar distintos controles de navegación por el mapa, herramientas de selección alfanumérica y espacial, distintas funciones relacionadas con la visualización de las distintas capas, modificación de las propiedades de simbología y vista en modo layout.

[18] *JavaBeans es un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java.*

- MapObjects Java incluye las herramientas para realizar tareas de edición de ficheros Shapefile.

- Como complemento a la extensa documentación que incluye, MapObjects Java incorpora un tutorial con numerosos applets y su código fuente correspondiente.

2.5 CONCLUSIÓN DE LA EVALUACIÓN Y SELECCIÓN DE LAS DISTINTAS HERRAMIENTAS A UTILIZAR.

En resumen para la elaboración de este proyecto, se utilizaron los programas y herramientas de aplicación que han sido elegidos en los análisis anteriores, los cuales son:

Para manejo y administración de Bases de Datos:

- ***PostgreSQL.***

Para el desarrollo de aplicaciones que tratan la Geoinformación.

- ***PostGIS.***

Como Lenguaje de Programación para el desarrollo de la Interfaz gráfica de Usuario:

- ***Java.***

Como herramientas de despliegue y consulta para la presentación de Información Geográfica:

- ***MapObject Java Edition.***

Como Entorno de Desarrollo Integrado (***IDE***[19]), existen muchas herramientas que pueden ser utilizadas, para este proyecto se utilizó:

- ***NetBeans.***

[19] *Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI, además de proveer un marco de trabajo amigable para la mayoría de los lenguajes de programación.*

CAPÍTULO III
DISEÑO DE LA ARQUITECTURA

3.1 INTRODUCCIÓN

En capítulos anteriores se dio un panorama sobre el Consorcio Internacional Open Gis formado por empresas, Organismos gubernamentales y Universidades que participan en un proceso para el desarrollo de especificaciones de interfaces disponibles para el público en general.

OpenGIS Consortium es la organización que define los estándares SIG, la visión de dicha organización es: "Un mundo en el que todos se beneficien de la Información Geográfica y que los servicios estén disponibles a través de cualquier red, aplicación o sistema."
Además, su misión principal es entregar especificaciones de interfaces espaciales que estén disponibles para su uso global.

En este capítulo se presentan las especificaciones de dicho estándar, de forma breve, las cuales serán utilizadas para la creación de la Interfaz Gráfica.

Además de presentar como se apega PostGIS a dicho estándar, y la forma de realizar operaciones como: Insertar, consultar, etc.

3.2 ESTÁNDARES E INTEROPERABILIDAD

Para comprender mejor los capítulos que se desarrollan, a continuación se presentan una serie de conceptos que serán utilizados en el desarrollo del contenido.

Estándares: Son acuerdos documentados que contienen especificaciones técnicas o criterios precisos que son utilizados consistentemente, como reglas, guías o definiciones de características para asegurar que los materiales, productos, procesos y servicios cumplen con su propósito.

Especificaciones: Son declaraciones que establecen instrucciones para la implementación de estándares, teniendo en cuenta requerimientos de contenido y uso en una materia en particular.
Las especificaciones de producto contienen todos los elementos que se requieren para describir de manera completa, un producto geográfico.

Estándares de Datos: Definiciones sobre el formato de los datos, establecidas con el fin de permitir la transferencia entre diversas aplicaciones informáticas. El estándar define cómo se organizan los datos. Un estándar puede implementarse en un SIG, o el SIG puede procesar formatos estándares de datos. Otros tipos de estándares definen estructuras, contenido, valor y comunicación de los datos.

Estándares de estructura: Se refieren a la definición de un registro de datos y las relaciones de sus campos.

Estándares de contenido: Son reglas que definen cómo se capturan los datos, mediante el uso de convenciones de sintaxis.

Estándares de valor: Son vocabularios controlados que incluyen terminología de una temática específica, nombres y lugares.

Estándares para comunicación o intercambio: Protocolos que definen el contexto técnico para el intercambio de datos entre sistemas. Operan en una institución o entre sistemas de múltiples instituciones.

Interoperabilidad: Capacidad que poseen sistemas distintos, de acceder a múltiples ambientes de geoprocesamiento, entender el significado de los datos, compartir e integrar la información de manera consistente.

A continuación se describen los estándares más importantes definidos por OGC. Existen más de los que aquí se describen:

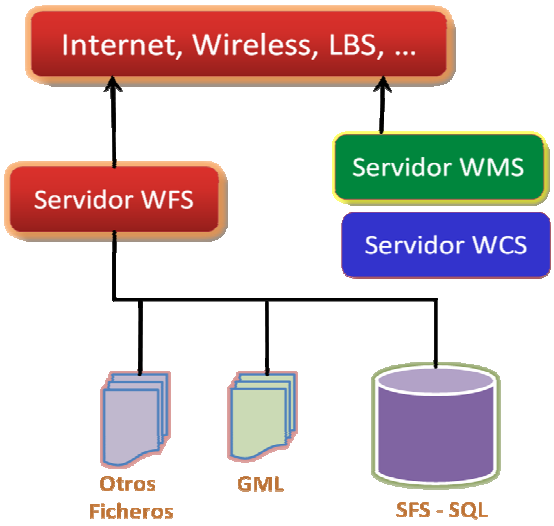


Figura 6: Diagrama de Estándares OGC

- ***Simple Features Specification for SQL – SFS***

Define cómo debe ser el almacenamiento de entidades geográficas en un Sistema Gestor de Bases de Datos Relacionales (SGBDR). Hay que tener en cuenta que un objeto geográfico se define como una geometría con atributos, es decir, la base es un modelo entidad-relación. Esta especificación define un modelo básico para almacenar geometrías en Base de Datos, proporcionando tres posibilidades de almacenamiento de las geometrías: Tipos Geométricos, WKB y WKT.

La representación ***Well-Known Text (WKT)*** de Geometrías está diseñada para intercambiar datos geométricos en formato ASCII.

Ejemplos de representaciones WKT de objetos geométricos son:

- Un Point:

  POINT (15 20)

  Las coordenadas del punto se especifican sin coma separadora.

- Una LineString con cuatro puntos:

  LINESTRING (0 0, 10 10, 20 25, 50 60)

La representación ***Well-Known Binary (WKB)*** de valores geométricos, se utiliza para intercambiar datos como cadenas binarias representadas por valores ***BLOB***[20] que contienen información geométrica WKB.
WKB utiliza enteros sin signo de un byte, enteros sin signo de cuatro bytes, y números de ocho bytes de doble precisión.

[20] *Un BLOB es un objeto binario que puede tratar una cantidad de datos variables.*

Por ejemplo, un valor WKB que corresponde a un POINT(1 1) consiste en esta secuencia de 21 bytes (cada uno representado aquí por dos dígitos hexadecimales):

0101000000000000000000F03F000000000000F03F
La secuencia puede descomponerse en los siguientes componentes:
Orden de byte: 01
Tipo WKB: 01000000
X: 000000000000F03F
Y: 000000000000F03F

El tipo WKB es un código que indica el tipo de geometría. Los valores del 1 al 7 significan Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, y GeometryCollection.

- ***Web Map Service – WMS***

Especifica el comportamiento que debe tener un servidor de mapas para ser considerado estándar. Se trata de una aplicación sin estados, que responde de una forma u otra según sea la petición recibida. Tanto peticiones como respuestas se realizan en código ***XML***[21]. Un servidor de mapas de tipo WMS funciona de la siguiente manera:

Dada una petición de mapa en formato XML, genera un nuevo mapa (en una imagen), y devuelve código XML indicando dónde está esa imagen. La respuesta a peticiones de tipo identificar es código XML con los datos devueltos. Además de esto, un servidor WMS debe responder a la petición getCapabilities. Con esta petición se pretende devolver un fichero XML que permita al cliente saber qué puede pedir a ese servidor.

[21] *e**X**tensible **M**arkup **L**anguage, es un metalenguaje extensible de etiquetas y permite definir la gramática de lenguajes específicos.*

- ***Web Coverage Service – WCS***

Extiende la interface Web Map Server (WMS) para proporcionar acceso a "coberturas" que representan valores o propiedades de emplazamientos geográficos en lugar de los mapas generados por WMS (imágenes).

- ***Web Feature Service – WFS***

El objetivo de la especificación de la interfaz Web Feature Server (WFS) es describir operaciones de manipulación de datos sobre objetos definidos en la especificación Simple Features de modo que los clientes y servidores pueden comunicarse a nivel de objeto. Esto permite enviar objetos (vectores) a los clientes, e incluso la edición de los mismos.

- ***Geography Markup Language – GML***

GML es una codificación XML para el intercambio y almacenamiento de Información Geográfica, incluyendo tanto geometría como atributos de los objetos.

- ***Catalog Interface - CAT***

Define una interfaz común que permite que diversas aplicaciones conformes a esta especificación realicen búsquedas, naveguen y consulten contra servidores de catálogos de Información Geográfica potencialmente heterogéneos.

A continuación se encuentran las entidades simples definidas según los estándares de OGC.

TIPO DE GEOMETRÍA	VALOR
GEOMETRIA	0
PUNTO	1
CURVA	2
LINEA STRING	3
SUPERFICIE	4
POLIGONO	5
COLECCION	6
MULTIPUNTO	7
MULTICURVA	8
MULTILINEA STRING	9
MULTISUPERFICIE	10
MULTIPOLIGONO	11

Tabla 6: Tipos de Entidades

3.3 SIMPLE FEATURE SPECIFICATIONS FOR SQL (SFS FOR SQL)

3.3.1 OBJETIVO

El propósito de esta especificación, es definir un estándar para un esquema de SQL (Structured Query Language) que soporte el almacenamiento, recuperación, consulta y actualización de una colección geoespacial de características simples. Una característica simple es definida por la especificación abstracta de OpenGIS para tener atributos espaciales y no espaciales. Los atributos espaciales son los valores de las geometrías y las características simples están basadas en una geometría en dos dimensiones (X,Y) con Interpolación Lineal entre vértices.

Una colección simple de características geoespaciales está conceptualmente almacenada en tablas cuyas columnas representan a las geometrías y los renglones las características. Esta especificación esta pensada para usar una Base de Datos Relacional (RDBMS).

GID	ETYPE	X	Y
100	1	12.5	34.56
101	1	45.67	898.9

Tabla 7: Geometrías y características

En la ***Tabla 7*** se muestra el campo GID, el cual representa la llave foránea de la tabla de características, el campo ETYPE se refiere el tipo de elemento, como es el caso del 1, el cual representa a la geometría del punto. La X y Y representan las coordenadas del punto.

3.3.2 MODELOS DE OBJETOS GEOMÉTRICOS

Este modelo está pensado para ser usado en una plataforma de cómputo distribuida y emplea la notación UML. La clase base es "Geometría" la cual tiene como subclases, "Punto", "Curva", "Superficie" y "Colección Geométrica". Cada objeto geométrico está asociado con un Sistema Espacial (Spatial Reference System), el cual describe las coordenadas del espacio en las cuales el objeto geométrico esta definido. En la ***Figura 7*** se pueden ver las jerarquías de los tipos de geometrías definidas por OpenGIS.

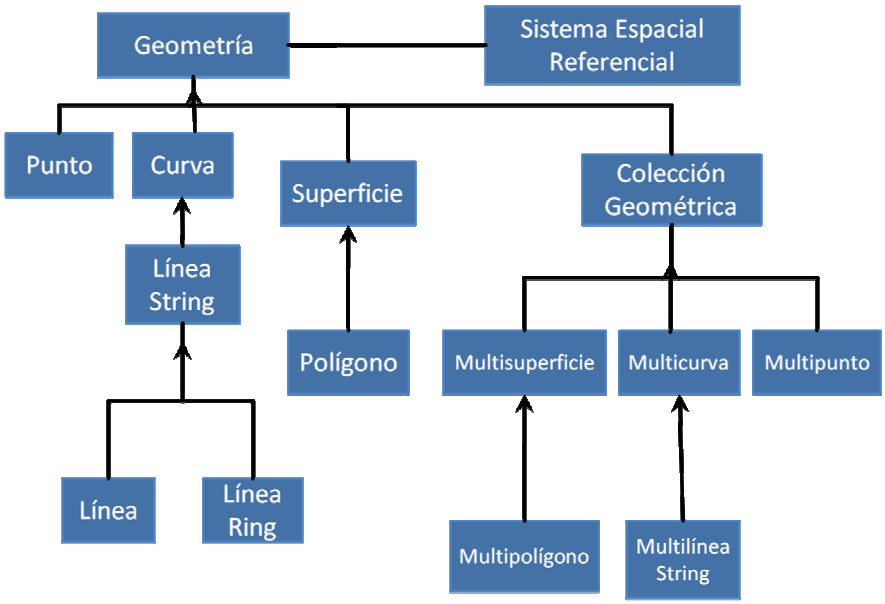


Figura 7: Jerarquía de los tipos de Geometría, es decir los tipos de atributos espaciales de una entidad.

3.3.2.1 DESCRIPCIÓN DE LAS CLASES

3.3.2.1.1 Clase geometría

Es la clase raíz de la jerarquía y es una clase abstracta (No instanciable).

Una geometría puede tener una dimensión de −1, 0, 1, o 2:

- −1 para una geometría vacía.
- 0 para una geometría sin longitud ni área.
- 1 para una geometría con longitud diferente de cero y área igual a cero.
- 2 para una geometría con área diferente de cero.

Todas las clases que se describen en esta especificación son definidas como instancias válidas de una geometría siempre y cuando estén topológicamente cerradas.

3.3.2.1.1.1 Métodos de la Clase Geometría

Los métodos definidos de la clase Geometría se dividen en tres tipos:

1. Métodos básicos: *Dimension( ), GeometryType( ), SRID( ), Envelope( ), AsText( ), AsBinary( ), IsEmpty( ), IsSimple( ), Boundary( ).*

2. Métodos para las relaciones espaciales entre objetos geométricos: *Equals( ), Disjoint( ), Intersects( ), Touches( ), Crosses( ), Within( ), Contains( ), Overlaps( ), Relate( ).*

3. Métodos para el análisis espacial: Distance( ), Buffer( ), ConvexHull( ), *Intersection( ), Union( ), Difference( ), SymDifference( ).*

3.3.2.1.2 Clase colección geométrica

Esta clase es una geometría la cual es una colección de una o más geometrías. Esta relación se puede ver en la figura 2.10 y es la definida por OpenGIS. Todos los elementos de la colección, deben estar en el mismo sistema de referencia espacial. Esta clase no pone restricciones en sus elementos, sin embargo las subclases pueden restringir la pertenencia basada en la dimensión y puede poner también otras restricciones en el grado de empalme espacial entre elementos.

3.3.2.1.1.1 Métodos de la Clase Colección Geométrica

Los métodos definidos en la clase Colección Geométrica son los siguientes:

1. NumGeometries( ): Regresa un entero el cual es el número de geometrías en esta Colección Geométrica.

2. GeometryN(int N): Regresa la N-ésima geometría de la Colección Geométrica.

3.3.2.1.3 Clase punto

Un Punto es una geometría que representa una ubicación única en un espacio de coordenadas.

Ejemplos de Punto

- En un mapa a gran escala del mundo con muchas ciudades. Un objeto Punto podría representar cada ciudad.
- En un mapa de una ciudad, un objeto Point podría representar una parada de bus.

Un Punto tiene las siguientes propiedades

- Valor de la coordenada X.
- Valor de la coordenada Y.
- Punto es definido como una geometría cero-dimensional.
- El límite de un Punto es el conjunto vacío.

Los métodos definidos para la Clase Punto son los siguientes:

1. X( ): Coordenada X del Punto
2. Y( ): Coordenada Y del Punto

3.3.2.1.4 Clase MultiPunto

La clase MultiPunto es una geometría de 0 dimensiones. Los elementos de la clase MultiPunto estan restringidos por la clase Punto. Los puntos no están conectados ni tampoco tienen un orden. Se dice que el MultiPunto es simple si no existen dos puntos con coordenadas idénticas. El límite de un MultiPunto es un conjunto vacío.

3.3.2.1.5 Clase Curva

La clase Curva, es una clase no instanciable, la cual representa un objeto geométrico de una dimensión que usualmente es almacenado como una secuencia de puntos. Este tipo de especificación solamente define un tipo de subclase de Curva, LíneaString, el cual usa interpolación lineal entre puntos. La clase Curva, es simple si no tiene elementos que pasen por un mismo punto dos veces, es cerrada si el punto inicial es igual al final, es un anillo si es simple y cerrada. El límite de una curva no cerrada esta dada por dos puntos finales.

3.3.2.1.5.1 Métodos de la clase Curva

1. Length ( ): Regresa un Doble el cual representa la longitud de la Curva asociado a la referencia espacial.

2. StartPunto ( ): Regresa un Punto el cual es el punto inicial de la Curva.

3. EndPunto ( ): Regresa un Punto el cual es el Punto final de la Curva.

4. IsClosed ( ): Regresa un entero el cual es 1 (TRUE) si (StartPunto( )=EndPunto( ) de otra forma es falso (FALSE).

5. IsRing ( ): Regresa un entero el cual es 1 (TRUE) si esta Curva es cerrada (**IsClosed( )**) y si es simple, es decir que no pasa por un punto dos veces.

3.3.2.1.6 Clases LineString, Line y Linear Ring

Un LineString es una curva con interpolación lineal entre puntos.

Ejemplos de LineString

- En un mapa del mundo, los objetos LineString podrían representar ríos.
- En un mapa de una ciudad, los objetos LineString podrían representar calles.

Propiedades de LineString

- Un LineString tiene coordenadas de segmentos, definidos por cada par consecutivo de puntos.
- Un LineString es una Línea (Line) si contiene exactamente dos puntos.
- Un LineString es un LinearRing si es tanto cerrado como simple.

En la ***Figura 8*** podemos ver que el 1 es un LineString simple, el 2 es un *LineString complejo, el 3 es un LineString cerrado que también es un LinearRing, y el 4 es* un LineString cerrado complejo.

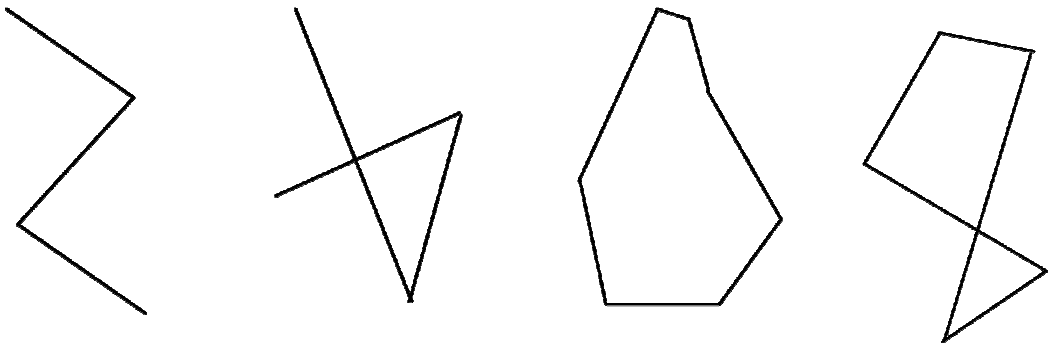


Figura 8: Diferentes tipos de LineString

Los métodos definidos para las clases LineString, Line y LinearRing son:

1. NumPoints( ): Regresa un entero el cual es el número de puntos en este LineString.

2. PointN(N: entero): Regresa un punto el cual es el punto N de este LineString.

3.3.2.1.7 MultiCurva

Una MultiCurva es un Colección Geométrica cuyos elementos son Curvas. En esta especificación la MultiCurva es una clase no instanciable, en ella se definen métodos para sus subclases y está incluída por razones de extensibilidad.
Una MultiCurva es simple si y solo si todos sus elementos son simples, la única intersección ocurre cuando dos puntos cualquiera coinciden en el mismo punto y son puntos terminales del segmento.

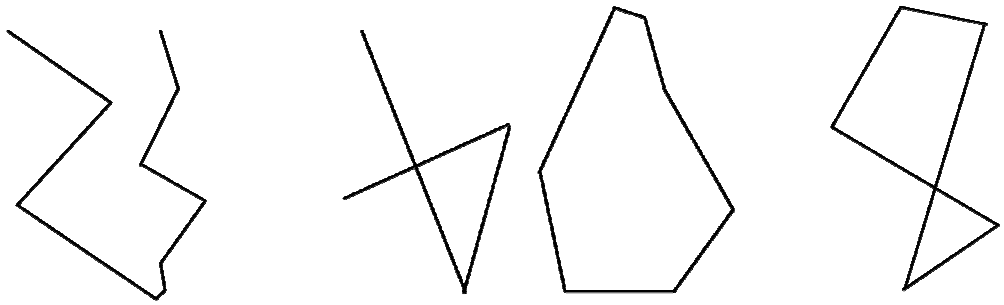
3.3.2.1.7.1 Métodos de la MultiCurva

1. IsClosed( ): Regresa un entero, 1 (TRUE) si todos los elementos son cerrados.

2. Length ( ): Regresa la longitud de la MultiCurva (la suma de longitudes de sus elementos), la cual es un Doble.

3.3.2.1.8 Clase MultiLineString

Una MultiLineString es una multicurva cuyos elementos son LineStrings. En la ***Figura 9*** podemos ver que el inciso 1 es una MultiLineString simple, el inciso 2 es una *MultiLineString compleja y el inciso 3 es una MultiLineString cerrada, compleja de dos* elementos.



(1) **(2)** **(3)**

Figura 9: Diferentes tipos de MultiLineString

3.3.2.1.9 Clase Superficie

Una Superficie (Surface) es una geometría bidimensional. Es una clase no instanciable. Su única subclase instanciable es Polígono

Propiedades de Superficie:

- Una superficie está definida como una geometría bidimensional.
- La especificación OpenGIS define una Superficie simple como una geometría que consiste de un único "trozo" que está asociado a un único límite exterior y cero o más límites interiores.
- El límite de una Superficie simple es el conjunto de curvas cerradas correspondientes a sus límites exterior e interior.

3.3.2.1.9.1 Métodos de la clase Superficie

1. Area( ): Regresa un Doble el cual es el área de este Superficie

2. Centroid ( ): Regresa un Punto el cual es el centro matemático de esta superficie.

3. PuntoOnSuperficie( ): Regresa un Punto, el cual se garantiza que esta en la superficie.

3.3.2.1.10 Clase Polígono

Un Polígono es una superficie plana, definida por un vecindario exterior y cero o más vecindarios interiores. Cada vecindario interior define un hueco en el Polígono.

Las reglas para decir que un Polígono es válido son:

1. Los Polígonos son topológicamente cerrados.
2. El vecindario de un Polígono consiste en conjunto de LinearRings que dividen el vecindario interior del exterior.
3. No se pueden cruzar dos anillos dentro del vecindario del Polígono, el único caso en el que se pueden intersectar es cuando el punto es tangente.
4. Un Polígono no puede tener líneas cortadas, puntos.
5. El interior del Polígono es un conjunto de puntos conectados.
6. El exterior de un Polígono con uno o más huecos no esta conectado. Cada hueco define un componente conectado al exterior.

La combinación de los puntos 1 y 3 hacen que el Polígono sea regular y cerrado.
Los Polígonos son geometrías simples.

3.3.2.1.10.1 Métodos de Polígono

1. ExteriorRing( ): Regresa el anillo exterior del Polígono como LineaString.

2. NumInteriorRings( ): Regresa el número de anillos interiores del Polígono como entero.

3. InteriorRingN(N: Integer): Regresa el anillo interior N

3.3.2.1.11 Clase MultiSuperficie

Una MultiSuperficie (MultiSurface) es una colección geométrica de dos dimensiones cuyos elementos son superficies, es una clase no instanciable. Su única subclase instanciable es MultiPolígono

- Dos superficies de MultiSuperficie no tienen interiores que se intersecten.
- Dos elementos de MultiSuperficie tienen límites que intersectan como máximo en un número finito de puntos.

3.3.2.1.11.1 Métodos de MultiSuperficie

1. Area( ): Regresa el área de esta MultiSuperficie, como Doble.

2. Centroid ( ): Regresa un punto el cual representa el centro matemático de esta MultiSuperficie.

3. PuntoOnSuperficie( ): Regresa un Punto, el cual se asegura que esta dentro de esta MultiSuperficie.

3.3.2.1.12 Clase MultiPolígono

Un MultiPolígono es una MultiSuperficie cuyos elementos son Polígonos.

Reglas de un MultiPolígono

1. El interior de dos Polígono que son elementos de un MultiPolígono no se intersectan.
2. Los vecindarios de dos Polígonos cualquiera que son elementos de un MultiPolígono no se pueden cruzar y se pueden tocar en solamente un número finito de puntos.
3. Un MultiPolígono es definido como topológicamente cerrado.
4. Un MultiPolígono no puede tener líneas cortadas, puntos, un MultiPolígono es regular y cerrado.
5. El número de componentes conectados del interior del MultiPolígono es igual al número de Polígonos en el MultiPolígono.

El vecindario de un MultiPolígono es un conjunto cerrado de Curvas (LineaStrings) correspondientes a los vecindarios de sus elementos Polígonos. Cada Curva en el vecindario de un MultiPolígono esta exactamente en el vecindario de un Polígono del MultiPolígono.

Cada Curva en el vecindario de un elemento Polígono esta en el vecindario de un MultiPolígono.

3.3.3 FORMAS DE IMPLEMENTACIÓN DE LA ESPECIFICACIÓN

La implementación de OpenGIS bajo SQL92 define un esquema de almacenamiento de las tablas de características, geometrías y referencias espaciales de los sistemas de información. La implementación bajo SQL92 no define funciones de SQL para acceso, mantenimiento o indexación de las geometrías, ya que estas especificaciones no pueden ser implementadas de manera uniforme a través de los Sistemas de Bases de Datos que emplean el estándar de SQL92.

La Figura 10 describe el Sistema de Base de Datos necesario para soportar el modelo simple de datos de OpenGIS. En dicho esquema se aprecian las relaciones existentes entre las tablas.

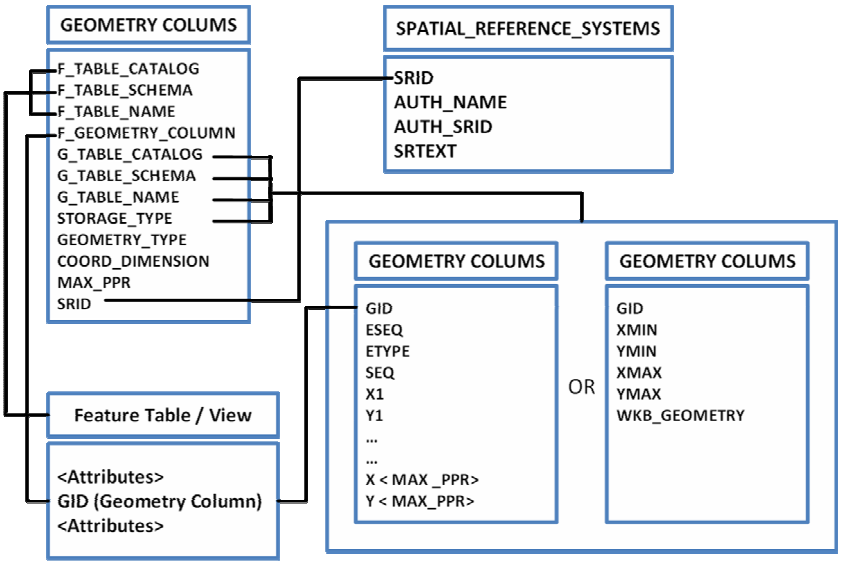


Figura 10: Tablas de la base de datos OpenGIS.

3.3.3.1 Feature Table/View

La feature table es una tabla o vista que tiene una o más llaves foráneas que hacen referencia a las tablas de geometrías o de vistas. Una feature table puede ser cualquier tabla que tenga una o más columnas donde el SQL Type esté definido por el conjunto de geometrías definidas en la ***Tabla 7.***

3.3.3.2 Geometry Columns

Esta tabla consiste en un renglón por cada columna de geometría de la Base de Datos. Los datos almacenados por cada geometría incluyen:

- Identificador de la tabla de características de la cual es miembro.
- Identificador del Sistema de Referencia Espacial.
- Un tipo de geometría por columna.
- En qué dimensión (2D, 3D) está la información de la columna.
- Identificación de las tablas de geometrías que almacenan sus instancias.
- Información necesaria para navegar en las tablas.

3.3.3.3 Spatial Reference System

Cada columna de las geometrías está asociada con un Sistema de Referencia Espacial, la cual identifica el tipo de sistemas de coordenadas para todas las geometrías almacenadas en las columnas y da significado a los valores geométricos para cualquier instancia de las geometrías almacenadas en la tabla. La tabla de información del Sistema de Referencia Espacial cuenta con los siguientes campos:

- SRID: Identificador del Sistema de Referencia Espacial. Clave única asignada por el administrador de la base.

- AUTH_NAME: Nombre de la autoridad del sistema de referencia espacial.

- AUTH_SRID: Identificador del Sistema de Referencia Espacial de la Autoridad.

- SR_TEXT: provee una representación textual del Sistema de Referencia Espacial.

3.3.4 ESPECIFICACIÓN DE COMPONENTES

Un programador puede decidir implementar por una de las tres alternativas que se describen a continuación, para cumplir con la especificación OpenGIS ODBC/SQL para la colección de características geoespaciales.

1. Usando los tipos numéricos de SQL para almacenamiento de las geometrías usando ODBC para el acceso.

2. Usando los tipos binarios de SQL para el almacenamiento de las geometrías usando ODBC para el acceso.

3. Usando SQL92 con tipos geométricos para el almacenamiento de las características de las tablas ya sea de forma binaria o de texto.

3.3.4.1 DEFINICIÓN DE FUNCIONES SQL

3.3.4.1.1 Funciones SQL para construir valores geométricos teniendo una representación WKT.

GeomFromText ( ): Regresa una geometría, construye un valor geométrico teniendo una representación WKT.

PointFromText ( ): Construye un punto.

LineFromText ( ): Construye una LineString.

PolyFromText ( ): Construye un Polígono.

MPointFromText ( ): Construye un Multipunto.

MLineFromText ( ): Construye un MultiLineString.

MPolyFromText ( ): Construye un Multipolígono.

GeomCollFromTxt ( ): Construye una Colección Geométrica.

BdPolyFromText ( ): Construye un Polígono teniendo una colección arbitraria de LineString cerradas de una representación MultiLineString tipo texto.

BdMPolyFromText ( ): Construye un Multipolígono teniendo una colección arbitraria de LineString cerrada de una representación MultiLineString tipo texto.

3.3.4.1.2 Funciones SQL para construir valores geométricos teniendo representación WKB.

GeomFromWKB ( ): Regresa una geometría, construye un valor geométrico teniendo una representación WKB.

PointFromWKB ( ): Construye un punto.

LineFromWKB ( ): Construye una LineString.

PolyFromWKB( ): Construye un Polígono.

MPointFromWKB ( ): Construye un Multipunto.

MLineFromWKB ( ): Construye un MultiLineString.

MPolyFromWKB ( ): Construye un Multipolígono.

GeomCollFromWKB ( ): Construye una Colección Geométrica.

BdPolyFromWKB ( ): Construye un Polígono teniendo una colección arbitraria de LineString cerradas de una representación binaria MultiLineString.

BdMPolyFromWKB ( ): Construye un Multipolígono teniendo una colección arbitraria de LineString cerrada de una representación binaria MultiLineString.

3.3.4.1.3 Funciones SQL para obtener una representación WKT de una geometría.

AsText ( ): Regresa una representación WKT tipo String.

3.3.4.1.4 Funciones SQL para obtener una representación WKB de una geometría.

AsBinary ( ): Regresa una representación WKB tipo binario.

3.3.4.1.5 Funciones SQL que prueban relaciones espaciales.

Estas funciones operan sobre dos valores geométricos g1 y g2.

Equals ( ): Retorna 1 o 0 para indicar si dos geometrías son iguales, es decir, si g1 es o no igual espacialmente a g2.

Disjoint ( ): Retorna 1 o 0 para indicar si g1 es o no espacialmente disjunto (no intersecta) con g2.

Touches ( ): Retorna 1 o 0 para indicar si g1 toca espacialmente o no a g2. Dos geometrías se tocan espacialmente si los interiores de las dos geometrías no intersecta, pero el límite de una de ellas intersecta con el límite o el interior de la otra.

Within ( ): Compara si una geometría está o no espacialmente contenida dentro de otra geometría; regresa un entero, 1 si es verdadero, 0 si es falso y -1 para valores nulos.

Overlaps ( ):

Retorna 1 o 0 para indicar si g1 se superpone espacialmente o no a g2. El término superpone espacialmente se utiliza si dos geometrías intersectan y la intersección resultante es una geometría de las mismas dimensiones pero no igual a ninguna de las geometrías dadas.

Crosses ( ):

Retorna 1 si g1 cruza espacialmente a g2. Retorna NULL si g1 es un Polígono o un MultiPolígono, o si g2 es un Punto o un MultiPunto. En cualquier otro caso, retorna 0.

El término cruza espacialmente denota una relación espacial entre dos geometrías dadas que tiene las siguientes propiedades:

- Las dos geometrías se intersectan.
- La intersección resulta en una geometría que tiene una dimensión que es una unidad menor que la dimensión máxima de las dos geometrías dadas.
- Su intersección no es igual a ninguna de las dos geometrías dadas.

Intersects ( ): Compara si la intersección de dos geometrías no es vacía; regresa un entero, 1 si es verdadero, 0 si es falso y -1 para valores nulos.

Contains ( ): Compara si una geometría está completamente contenida dentro de otra geometría; regresa un entero, 1 si es verdadero, 0 si es falso y -1 para valores nulos.

Relate ( ): Compara si la relación entre dos valores de geometría existen; regresa un entero, 1 si es verdadero, 0 si es falso y -1 para valores nulos.

3.3.4.1.6 Funciones SQL para relaciones de distancia.

Distance ( ): Regresa la distancia entre una geometría y otra; regresa un valor de doble precisión.

3.3.4.1.7 Funciones SQL que implementan operadores espaciales.

Intersection ( ): Regresa una geometría que es el conjunto intersectado de dos geometrías.

Difference ( ): Regresa una geometría que es el conjunto cerrado de la diferencia de dos geometrías.

Union ( ): Regresa una geometría que es la unión del conjunto de dos geometrías.

SymDifference ( ): Regresa una geometría que es el conjunto cerrado de la diferencia simétrica de dos geometrías.

Buffer ( ): Regresa una geometría definida por la distancia frontal alrededor de una geometría.

ConvexHull ( ): Regresa una geometría que es la capa convexa de dicha geometría.

3.3.4.1.8 Funciones de tipo geométrico.

Estas han sido descritas en la sección 3.3.2.1.

3.4 POSTGIS

Como ya sabemos, con PostGIS podemos utilizar todos los objetos que aparecen definidos en la especificación OpenGIS como puntos, líneas, polígonos, multilíneas, multipuntos y colecciones geométricas.

3.4.1 USO DEL ESTANDAR OPENGIS

La especificación para SQL de características simples de OpenGIS define tipos de objetos SIG estándar, los cuales son manipulados por funciones, y un conjunto de tablas de metadatos.

Dentro de la especificación, existen dos tablas:

- SPATIAL_REF_SYS
- GEOMETRY_COLUMNS

3.4.1.1 SPATIAL_REF_SYS

Esta tabla, contiene un identificador numérico y una descripción textual del Sistema de Coordenadas Espacial de la Base de Datos, dicha tabla se define de la siguiente manera:

```
CREATE TABLE SPATIAL_REF_SYS (
    SRID INTEGER NOT NULL PRIMARY KEY,
    AUTH_NAME VARCHAR(256),
    AUTH_SRID INTEGER,
    SRTEXT VARCHAR(2048),
    PROJ4TEXT VARCHAR(2048)
    )
```

Donde las columnas de la tabla son las siguientes:

- SRID : Valor entero que identifica el Sistema de Referencia Espacial.
- AUTH_NAME : Es el nombre del estándar para el Sistema de Referencia Espacial.
- AUTH_SRID : Es el identificador según el estándar AUTH_NAME.
- SRTEXT : Es una representación WKT para el Sistema de Referencia Espacial.
- PROJ4TEXT : PostGIS utiliza la librería Proj4 la cual provee la capacidad de transformación de coordenadas. La columna PROJ4TEXT contiene una cadena con la definición de las coordenadas de Proj4 para un SRID dado.

3.4.1.2 GEOMETRY_COLUMNS

GEOMETRY_COLUMNS se define de la siguiente manera:

```
CREATE TABLE GEOMETRY_COLUMNS (
  F_TABLE_CATALOG VARCHAR(256) NOT NULL,
  F_TABLE_SCHEMA VARCHAR(256) NOT NULL,
  F_TABLE_NAME VARCHAR(256) NOT NULL,
  F_GEOMETRY_COLUMN VARCHAR(256) NOT NULL,
  COORD_DIMENSION INTEGER NOT NULL,
  SRID INTEGER NOT NULL,
  TYPE VARCHAR(30) NOT NULL
)
```

Donde las columnas de la tabla son las siguientes:

- F_TABLE_CATALOG, F_TABLE_SCHEMA Y F_TABLE_NAME distingue totalmente la tabla de características que contiene la columna geométrica.
- F_GEOMETRY_COLUMN : Es el nombre de la columna geométrica en la tabla de características.
- COOR_DIMENSION : Es la dimensión espacial de la columna (2D, 3D).
- SRID : Es una clave foránea que referencia a SPATIAL_REF_SYS.
- TYPE : Es un tipo de objeto espacial, POINT, LINESTRING, POLYGON, MULTYPOINT, GEOMETRYCOLLECTION. Para tipos de objetos espaciales heterogéneos, se puede utilizar el tipo GEOMETRY.

3.4.2 CREACION DE UNA TABLA ESPACIAL

Para crear una tabla con datos espaciales, se necesita lo siguiente:

1. Creamos una tabla no espacial, como por ejemplo:

CREATE TABLE CALLES_GEOM (ID int4, NAME varchar (25)) ;

2. Se añade una columna o campo especial a la tabla anterior usando la funcion AddGeometryColumn definida en OpenGIS.

   AddGeometryColumn (<db_name>, <table_name>, <column_name>, <srid>, <type>,<dimension>)

   Como por ejemplo:

SELECT AddGeometryColumn (‘calles_bd’, ‘calles_geom’, ‘geom’, 423,
‘LINESTRING’, 2)

3.4.3 INTRODUCIENDO DATOS SIG EN LA BASE DE DATOS ESPACIAL.

Existen dos formas de introducir datos en las tablas de la base de datos:

- Utilizando lenguaje SQL
- Utilizando un cargador.

3.4.3.1 UTILIZANDO LENGUAJE SQL

Podemos convertir los datos que vamos a insertar en una representación textual. Se puede crear un archivo de texto lleno de sentencias INSERT e introducirlo con SQL monitor.

3.4.3.2 UTILIZANDO UN CARGADOR

El cargador de datos 'shp2pgsql' convierte archivos Shape de ESRI a SQL para su inserción en una base de datos PostGIS/PostgreSQL.

3.4.4 RECUPERACION DE DATOS SIG

Al igual que para la inserción de datos, también existen dos formas para recuperar datos:

- Mediante sentencias SQL
- Por medio de un cargador de archivos de Shapefile.

3.4.4.1 UTILIZANDO LENGUAJE SQL

La forma más directa de hacerlo es usando un SELECT. Como por ejemplo:

SELECT ID, AsText (GEOM) AS GEOM, NAME FROM ROADS_GEOM ;

SELECT ID, NAME FROM CALLES_GEOM WHERE GEOM = GeometryFromText ('LINESTRING (191232 243118, 191108 243242) ', -1) ;

Esta última, devuelve todos los registros cuya geometría es igual a la dada. El -1 indica que no se especifica ningún SRID.

3.4.4.2 UTILIZANDO UN CARGADOR (DUMPER).

Pgsql2shp conecta directamente con la base de datos y convierte una tabla en un archivo Shapefile. Su sintaxis es:

Pgsql2shp [< opciones >] <basededatos> <tabla>

Opciones:

-d Escribe un archivo de figuras 3D siendo el 2D el que tiene por defecto.

-f<archivo>	Es el archivo de salida.
-p<puerto>	Es el puerto de conexión con la Base de Datos.
-h<host>	Es el host donde está la Base de Datos.
-u<user>	Es el usuario de acceso.
-p<password>	Indica el password para el acceso.
-g<columna geometría>	Si la tabla tiene varias columnas geométricas, selecciona la columna geométrica a usar.

3.4.5 CLIENTES JAVA (JDBC)

Los clientes java, pueden acceder a los objetos geométricos de PostGIS directamente como una representación textual (WKT) o mediante JDBC con los objetos contenidos en PostGIS.

Para usar el JDBC debemos de tener en el CLASSPATH el paquete del driver JDBC "postgresql.jar".

Veamos el siguiente ejemplo:

```
importjava.sql.*;
importjava.util.*;
importjava.lang.*;
import.org.postgis.*;

public class JavaGIS {
public static void main ( String [ ] args )
{
 java.sql.Connection conn;
 try
  {
/*
*Cargar el driver JDBC y establecer la conexión
*/
Class.forName("org.postgresql.Driver");
string url = "jdbc:postgresql://localhost:5432/database";
conn =  DriverManager.getConnection(url,"postgres","");

/*
```

```
*Se añaden los tipos geométricos a la conexión se debe de hacer una prueba
*a pgsql-specific conection implementation antes de llamar al método
*addDataType()
*/

((org.postgresql.Connection )conn).addDataType("geometry", "org.postgis.PGgeometry" );
((org.postgresql.Connection )conn).addDataType("box3d", "org.postgis.PGbox3d");

/*
*Se crea   una consulta y se ejecuta.
*/


Statement s = conn.createStatement();
ResultSet r = s.executeQuery ("select AsText (geom) as geom, id from geomtable" ) ;
While ( r.next () )
{
/*
*Recupera el objeto como una geometría y hace una prueba a un tipo geométrico.
*/
PGgeometry geom = (PGgeometry) r.getObject (1);
int id = r.getInt (2);
      System.out.println ("Row"+id+":");
System.out.println (geom.toString () );
}
       s.close();
conn.close();
}
catch (Exception e )
{
e.printStackTrace ();
  }
 }
}
```

El objeto PGgeometry es un objeto envoltorio que contiene objetos topológicos específicos (subclases de la clase abstracta Geometry) dependiendo del tipo: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon.

CAPÍTULO IV

IMPLEMENTACIÓN DE LA ARQUITECTURA

4.1 INTRODUCCION

Como ya se mencionó anteriormente, la interfaz que se describirá a continuación fue desarrollada para el Área de Meteorología del Servicio Nacional de Estudios Territoriales.

Lo que dicha institución necesitaba, era que se creará una pequeña aplicación que cargará en pantalla el mapa de El Salvador, junto con las estaciones hidrométricas que ellos tienen alrededor de nuestro país. Dichas estaciones recogen información sobre los distintos parámetros que se miden.

Además de presentarlo en pantalla, se necesitaba que calculara algunos estadísticos básicos como promedios, valores máximos y valores mínimos de distintos parámetros, todo esto por medio de una selección en pantalla que el usuario realizaría a través de una herramienta de selección.

En dicha aplicación se utilizaron dos parámetros; el parámetro lluvia y el parámetro temperatura, aunque está abierta la opción de poder introducir otros parámetros.

Otra característica importante es que el usuario podrá escoger la fecha que desea consultar. Todos los cálculos que se realicen se podrán presentar por medio de un grafico estadístico como lo es el gráfico de barras.

La Unidad de Servicios Informáticos (USI) del Servicio Nacional de Estudios Territoriales nos proporcionó una Base de Datos, la cual se encontraba en formato Access (***Ver Anexo 3***) y contenía información sobre las diferentes estaciones hidrométricas, los diferentes parámetros que éstas medían así como también todas las mediciones tomadas por las estaciones.

Cabe aclarar que SNET solo nos proporcionó la información referente a las mediciones tomadas desde el mes de Enero del 2006 hasta el mes de Octubre del 2006, siendo así un aproximado de 5,000,000 de mediciones sobre los parámetros lluvia y temperatura.
Los datos obtenidos del parámetro lluvia son sobre la precipitación acumulada a lo largo del periodo o temporada lluviosa, y su unidad de medida es el mm.
Los datos obtenidos del parámetro temperatura, son la temperatura ambiente, la cual es medida en grados centígrados (ºC).

Para poder trasladar dicha Base de Datos de Access a PostgreSQL se utilizó un software llamado EMS PostgreSQL cuya función es la de importar Bases de Datos de Access a PostgreSQL.

Luego de haber explicado un poco sobre la aplicación que el Servicio Nacional de Estudios Territoriales necesitaba, describiremos los componentes y funcionalidades que dicha aplicación posee.

4.2 DISEÑO DE LA INTERFAZ

Dentro de la aplicación existe una pantalla principal que es con la que se realizan la mayoría de operaciones ***(Ver Figura 11).***

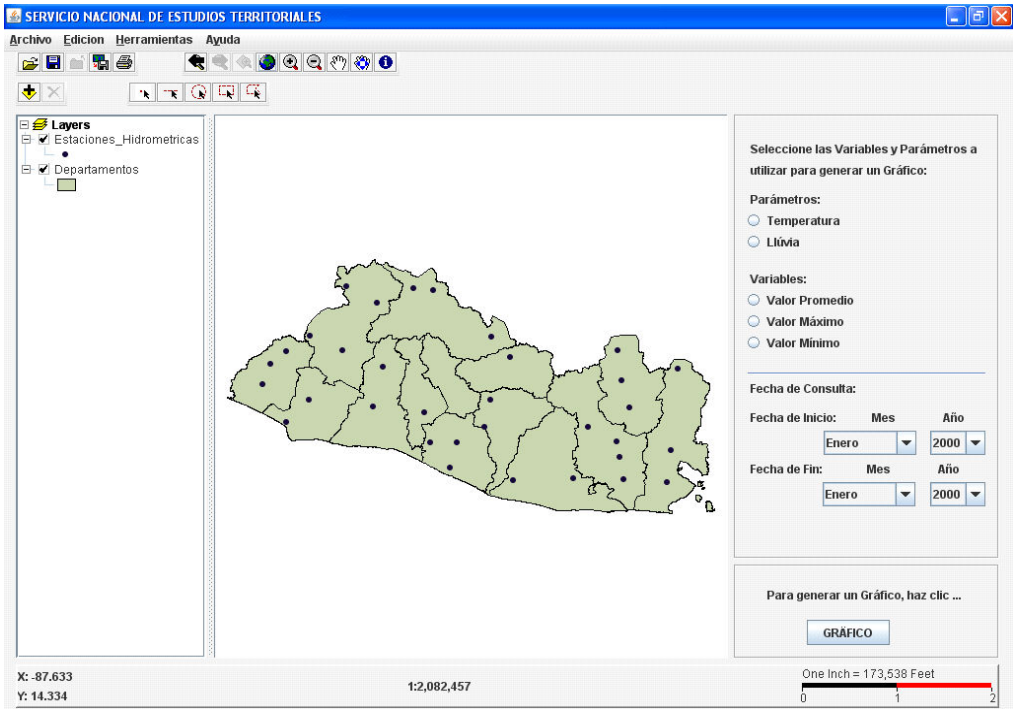


Figura 11: Pantalla Principal

La pantalla principal cuenta con una barra de menú en la que pueden colocarse distintos comandos y funciones, que se pueden definir para hacer una aplicación mas completa.
En la barra de menú hay opciones de archivo, edición, herramientas y ayuda.(***Figura 12***)

Figura 12: Barra de Menú

En el menú Archivo se colocaran funciones como: abrir un proyecto guardado, Cerrar la aplicación, entre otros.
El menú Edición tendrá funciones como copiar, pegar, cortar que son útiles y necesarias en una aplicación.
El menú Herramientas contendrá funciones para desplegar las barras de herramientas en el caso que se encuentren ocultas.
Por ultimo, el menú Ayuda, como su nombre lo dice, tendrá una pequeña ayuda acerca del uso de la aplicación, así como también tendrá el cuadro Acerca de, el cual guarda la información de los creadores y otro tipo de información.

Dentro de la misma interfaz, encontramos una serie de barras de herramientas, las cuales sirven para desplegar, manejar y consultar la información que se nos muestra en la pantalla

La primera barra es llamada ProjectToolBar (***Ver Figura 13***), es una Barra de Herramientas diseñada y desarrollada por MapObject, sus funciones son varias, entre las que se pueden mencionar:

- Abrir un proyecto ArcXML almacenándolo en su disco duro. (ArcXML es un lenguaje estándar de estructura e intercambio de documentos el cual sirve para comunicarse con los componentes de ArcIMS)
- Guardar un proyecto de trabajo para poder abrirlo posteriormente.
- Guardar el mapa desplegado en pantalla como una imagen jpg en el disco duro.
- Imprimir la imagen desplegada.

Figura 13: ProjectToolBar

Otra de las barras de herramientas se llama ZoomPanToolBar (***Figura 14***), la cual es de mucha utilidad, sus funciones son las siguientes:

- Funciones de Undo (Atrás) y Redo(Adelante).
- Función que muestra la extensión completa del mapa con que se está trabajando. (Zoom to FullExtent)
- Funciones de acercamiento (Zoom In) y alejamiento (Zoom Out) del mapa.
- Herramientas para arrastrar un mapa (Pan) dentro de nuestro contenedor.
- Funciones de despliegue de información de los atributos de un mapa determinado.

Figura 14: ZoomPanToolBar

La barra de herramientas llamada LayerToolBar nos sirve para poder cargar o quitar capas dentro de nuestro contenedor de mapas, la función de esta barra es muy importante para el manejo de mapas.(***Figura 15***)

Figura 15: LayerToolBar

Nuestra ultima Barra de Herramientas, es de gran utilidad, ya que su función es la de escoger los diferentes tipos de selección para poder realizar nuestra consulta. ***(Figura 16)***
Los diferentes tipos de selección son:

- Selección por medio de un punto.
- Selección por medio de una línea.
- Selección por medio de un círculo.
- Selección por medio de un rectángulo.
- Selección por medio de un polígono.

Figura 16: Barra de Selección

Siguiendo con la descripción de la interfaz podemos decir que en la parte central se encuentra un SplitPanel el cual esta compuesto por dos parte. En el lado izquierdo del SplitPanel se colocó una tabla de contenido el cual visualiza los Layers o capas existentes en un mapa. También en el lado derecho se utiliza para poder desplegar los diferentes mapas o capas. (***Figura 17***)

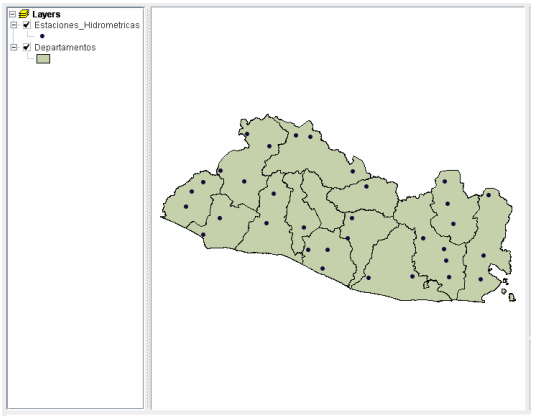


Figura 17: SplitPanel

Continuando con nuestra descripción, podemos ver en el lado derecho de la aplicación, dos paneles los cuales contienen la siguiente información ***(Figura 18):***

En el primer panel, se ha colocado, por medio de herramientas de selección, dos parámetros de medición. El parámetro Temperatura y el parámetro Lluvia. La selección dependerá de lo que el usuario necesite calcular.

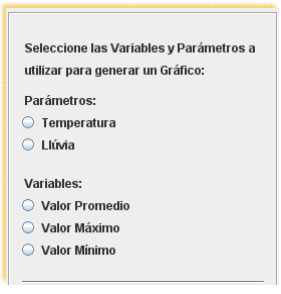


Figura 18: Selección de Parámetros y Variables

Mas abajo encontramos otro panel de selección, esta vez, lo que el usuario seleccionara será una de las variables a graficar (***Ver Figura 18***). En dicha selección tenemos tres opciones: graficar Valores Promedios, Valores Máximos o Valores Mínimos ya sea de cualquiera de los dos parámetros seleccionados anteriormente (Temperatura o lluvia).

Mas abajo, el siguiente panel servirá para poder seleccionar la fecha de inicio de nuestra consulta así como también poder seleccionar la fecha fin de la consulta. ***(Figura 19)***

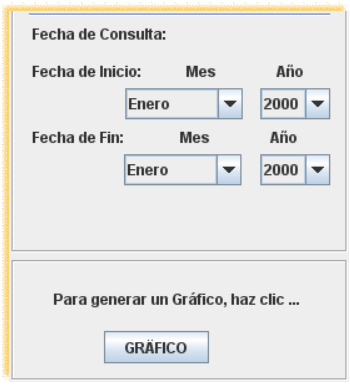


Figura 19: Selección de Fecha y Gráfico

Al final de este grupo de paneles se encuentra un botón (GRAFICO) el cual es el que se ejecutará para poder desplegar un grafico de barras con la información antes seleccionada.

Por ultimo, al final de la pantalla principal de nuestra interfaz, se encuentra la última barra de herramientas, dicha barra es un ScaleBar, ***(Figura 20)*** la cual nos servirá para poder saber posicionamiento del cursor en nuestro mapa así como tener una referencia o escala con respecto a nuestro mapa original.

Figura 20: ScaleBar

Otra de las pantallas que utiliza dicha aplicación es la pantalla de despliegue del grafico ***(Ver Figura 21)***, en dicha pantalla se desplegará el grafico generado a partir de las diferentes selecciones hechas en nuestra pantalla principal como lo es la selección del grafico, variable, parámetros y fechas de consulta.

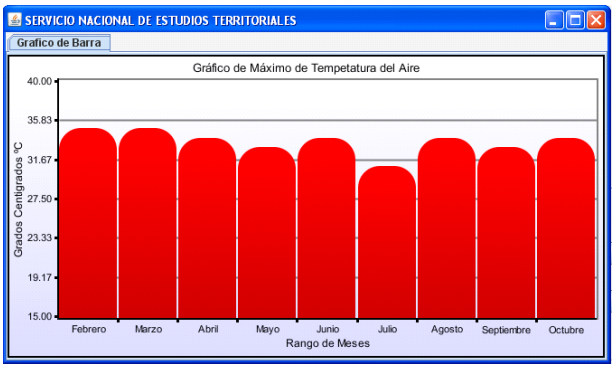


Figura 21: Gráfica de Barras

La interfaz desarrollada también contiene pantallas como pantalla de ayuda, acerca de, así como también pantallas de avisos o alertas.

4.3 DISEÑO DE DIAGRAMA DE CLASES

Para el desarrollo de la programación de la interfaz descrita anteriormente se utilizó, como ya se dijo, la plataforma de desarrollo integrado Netbeans, version 5.5

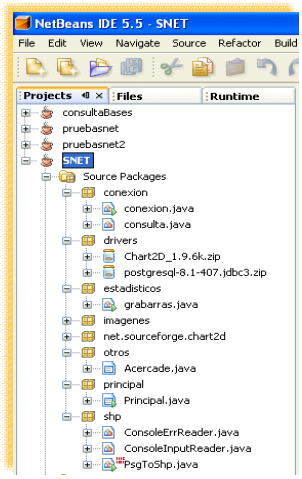


Figura 22: Librerías y Paquetes

En dicha plataforma se estructuró, en paquetes, el contenido de las distintas clases ***(Ver Figura 22)*** de la siguiente manera:

4.3.1 PAQUETE PRINCIPAL

Este paquete contiene el desarrollo de la clase llamada ***Principal.java*** y su función es la de crear un JFrame y desplegar en el todos los componentes como Barras de Menú, Barras de Herramientas, Paneles de Selección, Paneles Contenedores, así como también todos los eventos asociados a las diferentes clases creadas y utilizadas.

Dentro de la clase ***Principal.java*** se encuentran definidas otras clases como ***MySelect (Ver Figura 23)***, cuya función es la de crear el modo de selección de la barra de herramientas de selección.

```
  private MySelect select = new MySelect();

private BaseRubberCircle brcircle = new BaseRubberCircle();
private BaseRubberPoint brpoint = new BaseRubberPoint();
private BaseRubberPolygon brpolygon = new BaseRubberPolygon();
private BaseRubberPolyline brpolyline = new BaseRubberPolyline();
private BaseRubberEnvelope brenvelope = new BaseRubberEnvelope();


class MySelect extends Select {
    public void rubberbandCompleted() {
        super.rubberbandCompleted();

        SelectionSet ss = this.getSelection();
        com.esri.mo2.data.feat.Cursor c = null;
        Feature f = null;
        if (ss != null) {
            c = ss.getSelectedData(ss);
            while (c.hasMore()) {
                f = (Feature) c.next();
            }
        }

    }
}
```

Figura 23: Clase MySelect

En dicha clase se define las selecciones tipo punto, línea, circulo, rectángulo y polígono, que son utilizadas para la selección de áreas especificas de nuestro mapa a consultar.

Además de la definición de clases, la clase ***Principal.java*** define una serie de eventos por ratón, los cuales son responsables de ejecutar diferentes instrucciones y clases definidas dentro de nuestra aplicación.

4.3.2 PAQUETE CONEXIÓN

El paquete "***conexion"***, define dos clases: la clase ***conexion.java*** y la clase ***consulta.java.***

La clase ***conexion.java*** es la encargada de crear un objeto tipo consulta.

La clase ***consulta.java*** es la encargada de definir objetos tipo Connection y tipo Statement los cuales sirven para poder crear una conexión con nuestra Base de Datos en PostgreSQL, así como también crear sentencias de consulta SQL. ***(Ver Figura 24)***

```
public class consulta extends JFrame{
   static final String CONTROLADOR_JDBC = "org.postgresql.Driver";
   static final String URL_BASEDEDATOS = "jdbc:postgresql://localhost:5432/snet";
   String usr = "postgres";
   String pwd = "postgres";

   // Declaracion de objetos Connection y Statement para acceder a
   // la base de datos y realizar consultas
   private Connection conexion;
   private Statement instruccion;

   // el constructor se conecta a la base de datos, para realizar la consulta, procesa
   // los resultados y los muestra en la ventana
   public consulta ()
   {
      super( "Consulta a Base de Datos SNET" );
            // conectarse a la base de datos y consultar la base de datos
      try {

         // cargar clase de controlador de base de datos
         Class.forName( CONTROLADOR_JDBC );
         // establecer conexión a la base de datos
         conexion = DriverManager.getConnection( URL_BASEDEDATOS, usr, pwd );
         // crear objeto Statement para consultar la base de datos
         instruccion = conexion.createStatement();
         // consultar la base de datos
         ResultSet conjuntoResultados =
            instruccion.executeQuery( "SELECT * FROM mediciones" );
```

Figura 24: Clase Consulta

La clase ***consulta.java***, es la encargada de cargar el driver JDBC para PostgreSQL, el cual sirve para poder correr consultas y recuperar resultados de la Base de Datos.

4.3.3 PAQUETE SHP

Dentro del paquete ***"shp"***, ese encuentra definida la clase ***PsgToShp.java*** la cual nos sirve para poder leer desde nuestra Base de Datos la información necesaria para crear un archivo Shapefile y poder desplegarlo en la pantalla principal de la aplicación.

4.3.4 PAQUETE ESTADÍSTICOS

El paquete ***"Estadísticos"***, define la clase ***grabarras.java***, cuya función es muy importante, ya que es la encargada de crear un Gráfico de Barras dependiendo del parámetro y variable seleccionados en la pantalla principal de nuestra interfaz.

Para poder crear un Gráfico de Barras utilizamos la librería ***Chart2D versión 1.9.6*** la cual es una librería que es espacial para poder trabajar con Gráficos Estadístico; además, dicha librería es Open Source por lo que no hay inconvenientes en cuanto a su utilización.

En la clase ***grabarras.java*** se define todos los parámetros necesarios para poder dibujar un Gráfico de Barras.

4.3.5 PAQUETE DRIVER

En este paquete se almacena el driver de conexión JDBC para PostgreSQL el cual sirve para realizar una conexión a la Base de Datos.

Además, en el paquete ***"driver"*** se almacena también todas las clases utilizadas por la librería Chart2D para poder utilizarlas y realizar un Gráfico Estadístico.

4.3.6 PAQUETE IMÁGENES

Este paquete, no contiene clases, lo que contiene son todas las imágenes utilizadas en las distintas pantallas de nuestra interfaz, dichas imágenes pueden ser utilizadas en botones y barras de herramientas.

4.3.7 PAQUETE OTROS

El paquete ***"otros"*** contiene clases las cuales despliegan ventanas como ventana de ***Ayuda*** y ventana ***Acerca De.***

Dichas ventanas son importantes para la documentación de la aplicación.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Es posible desarrollar aplicaciones que en su mayoría son Open Source.

- El uso del estándar de OpenGis permite desarrollar aplicaciones que integre las fuentes de datos SIG y las Bases de Datos Relacionales.

- Se logró crear una pequeña aplicación para el área de Meteorología de SNET, la cual estaba orientada a usuarios finales.

- Se dejó claro la importancia de una estandarización, así como también el papel que juega OpenGis Consourtium dentro de esta estandarización.

- Este trabajo de graduación deja una puerta abierta para futuras investigaciones, y para poder seguir estandarizando. Por ejemplo, desarrollar los estándares de comunicación de información espacial.

RECOMENDACIONES

Una vez concluída la Tesis, se considera interesante investigar sobre otros aspectos relacionados con los Estándares de OpenGIS, tales como:

- Investigar sobre los demás estándares que ofrece OpenGIS para poder completar la puerta que se ha dejado abierta en este trabajo de graduación, aquí se plantean las bases que son la estándarización del almacenamiento de la información espacial; se puede dar seguimiento con estándares de comunicación.

- Trabajar en el mejoramiento de la Interfaz Gráfica de usuario, en el sentido que pueda realizar operaciones estadísticas más complejas o de acuerdo a las necesidades de los usuarios.

- Como herramientas de despliegue y consulta para la presentación de Información Espacial, se utilizaron las librerías de MapObject Java Edition, pero se puede investigar acerca de otras herramientas libres, y esto daría como resultado una aplicación totalmente libre.

BIBLIOGRAFÍA

✓ ***Libros de texto***

- *Harvey M. Deitel, Paul J. Deitel, quinta edición "Como Programar en Java", Introducción con UML y los Patrones de Diseño JDBC, SERVLETS, JSP.*
- *Rick Decker, Stuart Hirshfield, segunda edición "Programación con Java"*
- *Gis by Esri, "MapObjects Java Edition", Getting Started with MapObjects Java.*
- *MapObjects Java Edition 2.0, Programmer's Reference Help.*
- *Gis by Esri "Understanding Map Projections".*
- *Open GIS Consourtium, Inc., "OpenGIS Simple Features Specification For SQL.*
- *Robert C. Martin, "UML para Programadores Java.*
- *Rebecca M. Riordan, "Aprenda Programación en SQL Server 2000"*

- ✓ ***Recurso de Internet***

- ➢ ***MySQL, Oracle, PostgreSql y PostGis***

http://weblogs.javahispano.org/page/pacopaco

http://www.postgresql.org/about/?lang=es

http://alvherre.atentus.cl/intro/

http://www.postgresql.cl/entrevistas/

http://www.netpecos.org/docs/mysql_postgres/index.html

http://postgis.refractions.net/docs/index.html

http://www.cartografia.cl/index.php?option=com_content&task=view&id=227&Itemid=49

- ➢ ***Java y .Net***

http://www.fing.edu.uy/inco/cursos/tsi/TSI1/teorico2006/Clase3_NetFramework.pdf

http://adrformacion.com/cursos/puntonet/puntonet.html#1

http://www.willydev.net/descargas/articulos/general/j2eenet.aspx

http://es.wikipedia.org/wiki/Visual_Basic.NET

http://www.monografias.com/trabajos/java/java.shtml

http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/Index.htm

http://www.dma.fi.upm.es/mabellanas/voronoi/java/java.htm

http://www.programacion.com/tutorial/swing/

http://www.koders.com/

http://javahispano.net/snippet/browse.php?by=lang&lang=8

- ***Bases de Datos***

http://www.itlp.edu.mx/publica/tutoriales/basedat1/temas1.htm
http://www.itlp.edu.mx/publica/tutors.htm
http://es.tldp.org/Tutoriales/NOTAS-CURSO-BBDD/notas-curso-BD/node1.html
http://www.monografias.com/trabajos35/comparativa-bases-datos/comparativa-bases-datos.shtml

- ***Herramientas de Despliegue y Consulta***

http://www.bluemarblegeo.com/products/geoobjects.php?op=details
http://www.bluemarblegeo.com/docs/brochures/geoobj.pdf
http://www.tatukgis.com/products/Dk/kernel.aspx
http://www.esri-es.com/index.asp?pagina=72
http://www.clubdelphi.com/foros/printthread.php?t=4056&pp=20

- ***Otros***

http://www.lawebdelprogramador.com/news/
http://www.gnu.org/philosophy/free-sw.html
http://www.juntadeandalucia.es/averroes/manuales/guadaconceptos.html#GlossG
http://es.wikipedia.org/wiki/Portada

ANEXOS

ANEXO Nº 1: FORMATO DE CUESTIONARIO.

CUESTIONARIO PARA REALIZAR ENTREVISTA DIRIGIDA A SNET

1. Cuáles son las necesidades que tiene la institución en el área de SIG?
2. ¿Cuándo se conformó SNET y con qué fin?
3. ¿A través de que medios recopilan la Información Geográfica?
4. ¿Cuales son los software SIG que utilizan para el manejo de la Información Geográfica?
5. ¿Cuál es el departamento encargado en el manejo de la Información Geográfica?
6. ¿Cuáles son los Sistemas Operativos con los que trabaja la Institución?
7. ¿Cuáles son las instituciones que le aportan a SNET información acerca de los fenómenos que se estudian en las diferentes áreas?
8. ¿Cuáles son las áreas de especialización de SNET?
9. ¿Qué tipo de información manipulan y cual es el tipo de almacenamiento?
10. ¿Hacia que usuarios o con que fin crea información SNET?
11. ¿Qué tipo de Institución es SNET, Publica, Privada?
12. ¿Cuál es la mayor necesidad que posee SNET con respecto a la manipulación de Información Geográfica?
13. ¿Existe algún tipo de estructura organizativa dentro de SNET?
14. ¿Las herramientas y software que utilizan para el manejo de la información son de licencia pagada, código abierto o de libre distribución?
15. ¿Existe algún tipo de equipo especializado para el manejo de Información Geográfica?

ANEXO Nº 2: FORMATO DE CUESTIONARIO.

CUESTIONARIO PARA REALIZAR ENTREVISTA DIRIGIDA A LAS DIFERENTES ORGANIZACIONES QUE MANEJAN INFORMACIÓN GEOGRÁFICA.

1- ¿Considera que es mejor manejar los datos geográficos en Bases de datos Relacionales o en Sistema de Archivos y porqué?

2- ¿Considera que es importante desarrollar aplicaciones multiplataforma?

3- ¿Qué tipo de software SIG utilizan para el manejo de la información?

4- ¿Cuáles son las plataformas con las que trabajan?

5- ¿Desarrollan ustedes sus propias aplicaciones SIG?

6- ¿Sabe de la existencia de los estándares OpenGIS?

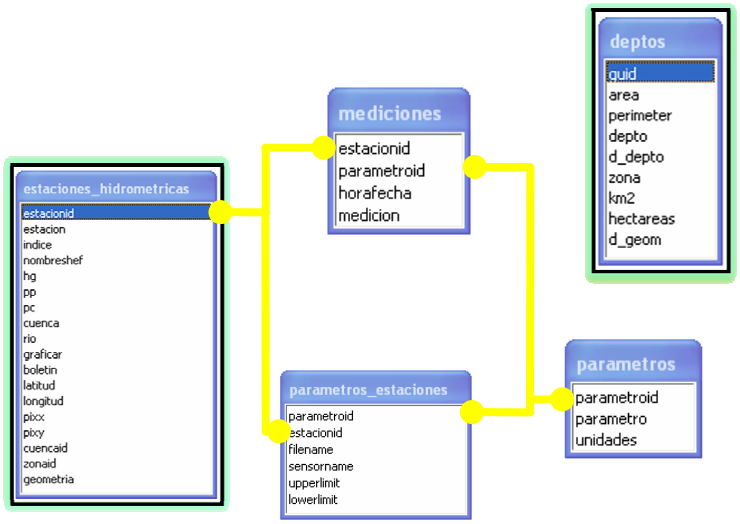
7- ¿Qué beneficio trae el uso de estándares de OpenGIS en los SIG?

8- ¿Considera usted que traería algún beneficio el desarrollar interfaces de SIG bajo estándares a las empresas que desarrollan o trabajan con algún SIG en el país?

9- ¿Qué ventajas piensa que tiene el desarrollo de éstas aplicaciones orientadas a usuarios finales frente a los software SIG comerciales?

10- ¿De existir dicha arquitectura implementarían ustedes interfaces basadas en esa arquitectura?

ANEXO Nº 3: ESQUEMA DE BASE DE DATOS PROPORCIONADA POR SNET.



ANEXO Nº 4: SENTENCIAS PARA LA CREACIÓN DE COLUMNA ESPACIAL, Y FORMA DE INSERTAR DATOS.

Las siguientes sentencias se utilizan para la creación de la columna geométrica en la tabla estaciones hidrométricas, que se deseaba estandarizar, en la Base de Datos, y la forma de insertar un dato en la columna, pero si la cantidad de datos a insertar es muy grande puede hacerse utilizando las sentencias de PostgreSQL.

```
psql to 'postgres'
snet =# Select AddGeometryColumn (
'snet', 'estaciones_hidrometricas',
 'geometria', -1, 'POINT', 2 );
```

```
psql to 'postgres'
snet =#  UPDATE estaciones_hidrometricas SET
geometría= GeometryFromText ('POINT (
-89. 2126007080078  14.3709001541138)',
-1 )  WHERE  estacionid = 13;
```

pgAdmin III Edit Data - PostgreSQL Database Server 8.1 (localhost:5432) - snet2 - estaciones_hidrometricas

	estacionid [PK] int8	estacion varchar	latitud float8	longitud float8	geometria geometry
1	2	El Zapotillo	14.1752996444702	-89.4130020141602	0101000000030000A06E5A56C0F8FFFFDFC0592C40
2	3	Belen Gualcho	14.4823999404907	-88.7942962646484	0101000000FDFFFFBFD53256C0F3FFFF1FFDF62C40
3	4	Ilopango	13.6983003616333	-89.1183013916016	0101000000030000040924756C0000000A087652B40
4	5	La Ceibita	14.4945001602173	-89.8788986206055	0101000000020000E03F7856C0080000202FFD2C40
5	6	Las Cruces	14.3168001174927	-89.6167984008789	0101000000000000A0796756C00E0000A033A22C40
6	7	La Lechuza	14.352499961853	-89.7133026123047	0101000000010000C0A66D56C0F1FFFFDF7AB42C40
7	8	Chapeltique	13.6423997879028	-88.2608032226563	0101000000040000000B11056C0EEFFFFF9FE8482B40
8	9	Osicala	13.8332004547119	-88.1500015258789	0101000000000000A0990956C0F8FFFF3F99AA2B40
9	10	San Marcos Lempa	13.4235000610352	-88.6968994140625	0101000000000000009A2C56C019000000D5D82A40
10	11	Las Flores	14.043399810791	-88.8089981079102	0101000000030000A0C63356C0F7FFFF7F38162C40
11	12	Tamarindo	14.0468997955322	-89.2525024414062	0101000000FCFFFFFF285056C0F1FFFF3F03182C40
12	13	Citala	14.3709001541138	-89.2126007080078	0101000000FFFFFF3F9B4D56C0110000A0E6BD2C40
13	14	La Esperanza	14.2938003540039	-88.1727981567383	0101000000010000200F0B56C0FCFFFFFF6C962C40
14	15	San Gregorio	13.9316997528076	-88.507698059082	0101000000FEFFFF1F7E2056C0F6FFFFBF07DD2B40

Tabla: Estaciones Hidrométricas con su Columna Geométrica

ANEXO Nº 5: TABLA SPATIAL_REF_SYS Y TABLA GEOMETRY_COLUMNS DE LA BASE DE DATOS SNET.

pgAdmin III Edit Data - PostgreSQL Database Server 8.1 (localhost:5432) - snet2 - spatial_ref_sys

	srid [PK] int4	auth_name varchar	auth_srid int4	srtext varchar	proj4text varchar
1	2000	EPSG	2000	PROJCS["Anguilla 1957 / British West Indies Grid",GEOGCS["	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
2	2001	EPSG	2001	PROJCS["Antigua 1943 / British West Indies Grid",GEOGCS["	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
3	2002	EPSG	2002	PROJCS["Dominica 1945 / British West Indies Grid",GEOGCS[	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
4	2003	EPSG	2003	PROJCS["Grenada 1953 / British West Indies Grid",GEOGCS[	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
5	2004	EPSG	2004	PROJCS["Montserrat 58 / British West Indies Grid",GEOGCS[	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
6	2005	EPSG	2005	PROJCS["St Kitts 1955 / British West Indies Grid",GEOGCS["	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
7	2006	EPSG	2006	PROJCS["St Lucia 1955 / British West Indies Grid",GEOGCS["	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
8	2007	EPSG	2007	PROJCS["St Vincent 45 / British West Indies Grid",GEOGCS["	+proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=
9	2008	EPSG	2008	PROJCS["NAD27(CGQ77) / SCoPQ zone 2",GEOGCS["NAD27	+proj=tmerc +lat_0=0 +lon_0=-55.5 +k=0.999900 +x_
10	2009	EPSG	2009	PROJCS["NAD27(CGQ77) / SCoPQ zone 3",GEOGCS["NAD27	+proj=tmerc +lat_0=0 +lon_0=-58.5 +k=0.999900 +x_
11	2010	EPSG	2010	PROJCS["NAD27(CGQ77) / SCoPQ zone 4",GEOGCS["NAD27	+proj=tmerc +lat_0=0 +lon_0=-61.5 +k=0.999900 +x_

pgAdmin III Edit Data - PostgreSQL Database Server 8.1 (localhost:5432) - snet2 - geometry_columns

	oid	f_table_name varchar	f_geometry_column varchar	coord_dimension int4	srid int4	type varchar
1	27127	estaciones_hidrometricas	geometria	2	-1	POINT
2	27145	deptos	the_geom	2	-1	MULTIPOLYGON
3	27178	basepais	the_geom	2	-1	MULTIPOLYGON
4	27310	municipios	the_geom	2	-1	MULTIPOLYGON
*						

ANEXO Nº 6: CLASES UTILIZADAS PARA LA CREACIÓN DE LA INTERFAZ GRÁFICA DE USUARIO.

CLASE PRINCIPAL

```
/*
 * Principal.java
 */

package principal;

import com.esri.mo2.data.feat.Feature;
import com.esri.mo2.data.feat.SelectionSet;
import com.esri.mo2.map.dpy.FeatureLayer;
import com.esri.mo2.ui.bean.BaseRubberCircle;
import com.esri.mo2.ui.bean.BaseRubberEnvelope;
import com.esri.mo2.ui.bean.BaseRubberPoint;
import com.esri.mo2.ui.bean.BaseRubberPolygon;
import com.esri.mo2.ui.bean.BaseRubberPolyline;
import com.esri.mo2.ui.bean.Select;
import conexion.consulta;
import estadisticos.grabarras;
import java.awt.Color;
import javax.swing.JOptionPane;
import otros.Acercade;
import otros.Ayuda;
```

```
public class Principal extends javax.swing.JFrame {


    private MySelect select = new MySelect();

    private BaseRubberCircle brcircle = new BaseRubberCircle();
    private BaseRubberPoint brpoint = new BaseRubberPoint();
    private BaseRubberPolygon brpolygon = new BaseRubberPolygon();
    private BaseRubberPolyline brpolyline = new BaseRubberPolyline();
    private BaseRubberEnvelope brenvelope = new BaseRubberEnvelope();


    class MySelect extends Select {
        public void rubberbandCompleted() {
            super.rubberbandCompleted();

            SelectionSet ss = this.getSelection();
            com.esri.mo2.data.feat.Cursor c = null;
            Feature f = null;
            if (ss != null) {
                c = ss.getSelectedData(ss);
                while (c.hasMore()) {
                    f = (Feature) c.next();
                }

            }

        }

    }



    /** Creación del Nuevo Formulario Principal */

        public Principal() {
        initComponents();
    }
```

```
/** Este es el método que es llamado por el constructor para
 * poder inicializar el formulario.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    buttonGroup2 = new javax.swing.ButtonGroup();
    jPanel1 = new javax.swing.JPanel();
    layerToolBar1 = new com.esri.mo2.ui.tb.LayerToolBar();
    zoomPanToolBar1 = new com.esri.mo2.ui.tb.ZoomPanToolBar();
    projectToolBar1 = new com.esri.mo2.ui.tb.ProjectToolBar();

    jToolBar1 = new javax.swing.JToolBar();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    jButton4 = new javax.swing.JButton();
    jButton5 = new javax.swing.JButton();

    jPanel2 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();

    jRadioButton1 = new javax.swing.JRadioButton();
    jRadioButton2 = new javax.swing.JRadioButton();
    jRadioButton3 = new javax.swing.JRadioButton();

    jLabel5 = new javax.swing.JLabel();

    jRadioButton4 = new javax.swing.JRadioButton();
    jRadioButton5 = new javax.swing.JRadioButton();

    jSeparator2 = new javax.swing.JSeparator();

    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();

    jComboBox1 = new javax.swing.JComboBox();
    jComboBox2 = new javax.swing.JComboBox();

    jLabel8 = new javax.swing.JLabel();
```

```
jComboBox5 = new javax.swing.JComboBox();
jComboBox3 = new javax.swing.JComboBox();

jPanel3 = new javax.swing.JPanel();

jLabel9 = new javax.swing.JLabel();

jButton6 = new javax.swing.JButton();

jSplitPane1 = new javax.swing.JSplitPane();
jScrollPane1 = new javax.swing.JScrollPane();

treeToc1 = new com.esri.mo2.ui.toc.TreeToc();
map1 = new com.esri.mo2.ui.bean.Map();
layer1 = new com.esri.mo2.ui.bean.Layer();
scaleBar1 = new com.esri.mo2.ui.bean.ScaleBar();

jMenuBar1 = new javax.swing.JMenuBar();
Archivo = new javax.swing.JMenu();
Abrir = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JSeparator();
Cerrar = new javax.swing.JMenuItem();

Edicion = new javax.swing.JMenu();
Copiar = new javax.swing.JMenuItem();
Cortar = new javax.swing.JMenuItem();
Pegar = new javax.swing.JMenuItem();

Herramientas = new javax.swing.JMenu();
Barraherram = new javax.swing.JMenuItem();

Ayuda = new javax.swing.JMenu();
Ayudas = new javax.swing.JMenuItem();
Acerca = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("SERVICIO NACIONAL DE ESTUDIOS TERRITORIALES");
setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

layerToolBar1.setFloatable(false);
layerToolBar1.setMap(map1);
```

```
    zoomPanToolBar1.setFloatable(false);
    zoomPanToolBar1.setMap(map1);

    projectToolBar1.setFloatable(false);
    projectToolBar1.setMap(map1);

    jToolBar1.setFloatable(false);
    jButton1.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/imagenes/select_point.GIF")
   )
);

    jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
       public void mousePressed(java.awt.event.MouseEvent evt) {
          jButton1MousePressed(evt);
       }
    }
);


    jToolBar1.add(jButton1);

    jButton2.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/imagenes/select_line.gif")
     )
);


    jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
       public void mousePressed(java.awt.event.MouseEvent evt) {
          jButton2MousePressed(evt);
       }
    }
);


    jToolBar1.add(jButton2);

    jButton3.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/imagenes/select_circle.gif")
     )
);
```

```
    jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mousePressed(java.awt.event.MouseEvent evt) {
            jButton3MousePressed(evt);
        }
    }
);

    jToolBar1.add(jButton3);


    jButton4.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/imagenes/select_rect.gif")
    )
);


    jButton4.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mousePressed(java.awt.event.MouseEvent evt) {
            jButton4MousePressed(evt);
        }
    }
);


    jToolBar1.add(jButton4);

    jButton5.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/imagenes/select_poly.gif")
    )
);


    jButton5.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mousePressed(java.awt.event.MouseEvent evt) {
            jButton5MousePressed(evt);
        }
    }
);


    jToolBar1.add(jButton5);
```

```
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(jPanel1Layout.createSequentialGroup()

   .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)

   .addGroup(jPanel1Layout.createSequentialGroup()
   .addComponent(layerToolBar1, javax.swing.GroupLayout.PREFERRED_SIZE,

       javax.swing.GroupLayout.DEFAULT_SIZE,
       javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(61, 61, 61)
            .addComponent(jToolBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
             javax.swing.GroupLayout.DEFAULT_SIZE,
             javax.swing.GroupLayout.PREFERRED_SIZE)
           )

     .addGroup(jPanel1Layout.createSequentialGroup()
            .addComponent(projectToolBar1,
             javax.swing.GroupLayout.PREFERRED_SIZE,
             javax.swing.GroupLayout.DEFAULT_SIZE,
             javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(45, 45, 45)
            .addComponent(zoomPanToolBar1,
             javax.swing.GroupLayout.PREFERRED_SIZE,
             javax.swing.GroupLayout.DEFAULT_SIZE,
             javax.swing.GroupLayout.PREFERRED_SIZE)
           )
       )

      .addContainerGap(137, Short.MAX_VALUE)
   )
);


jPanel1Layout.setVerticalGroup(
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
             Alignment.LEADING)

               .addComponent(projectToolBar1,
                javax.swing.GroupLayout.PREFERRED_SIZE, 25,
                javax.swing.GroupLayout.PREFERRED_SIZE)
               .addComponent(zoomPanToolBar1,
                javax.swing.GroupLayout.PREFERRED_SIZE, 25,
                javax.swing.GroupLayout.PREFERRED_SIZE)
              )

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
             Alignment.LEADING)
               .addComponent(layerToolBar1,
                javax.swing.GroupLayout.PREFERRED_SIZE, 25,
                javax.swing.GroupLayout.PREFERRED_SIZE)
               .addComponent(jToolBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
                25, javax.swing.GroupLayout.PREFERRED_SIZE)
              )
            .addContainerGap()
         )
      );

      jPanel2.setBorder(javax.swing.BorderFactory.createEtchedBorder()
       );

      jLabel2.setText(" Seleccione las Variables y Par\u00e1metros a");

      jLabel3.setText(" utilizar para generar un Gr\u00e1fico:");

      jLabel4.setText(" Variables:");

      buttonGroup2.add(jRadioButton1);
      jRadioButton1.setText(" Valor Promedio");
      jRadioButton1.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0)
      );

      jRadioButton1.setMargin(new java.awt.Insets(0, 0, 0, 0)
      );
```

```
        buttonGroup2.add(jRadioButton2);
        jRadioButton2.setText(" Valor M\u00e1ximo");
        jRadioButton2.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0)
        );

        jRadioButton2.setMargin(new java.awt.Insets(0, 0, 0, 0)
        );

        buttonGroup2.add(jRadioButton3);
        jRadioButton3.setText(" Valor M\u00ednimo");
        jRadioButton3.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0)
        );

        jRadioButton3.setMargin(new java.awt.Insets(0, 0, 0, 0)
        );

        jLabel5.setText(" Par\u00e1metros:");

        buttonGroup1.add(jRadioButton4);
        jRadioButton4.setText(" Temperatura");
        jRadioButton4.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0)
        );

        jRadioButton4.setMargin(new java.awt.Insets(0, 0, 0, 0));

        buttonGroup1.add(jRadioButton5);
        jRadioButton5.setText(" Ll\u00favia");
        jRadioButton5.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0)
        );

        jRadioButton5.setMargin(new java.awt.Insets(0, 0, 0, 0)
        );

        jLabel6.setText(" Fecha de Consulta:");

        jLabel7.setText(" Fecha de Inicio:          Mes          A\u00f1o");

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre",
"Octubre", "Noviembre", "Diciembre" }
        )
    );
```

```
        jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007" }
        )
    );

        jLabel8.setText(" Fecha de Fin:              Mes              A\u00f1o ");


        jComboBox5.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre",
"Octubre", "Noviembre", "Diciembre" }
        )
    );

        jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007" }
        )
    );

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

                jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
                Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
            jPanel2Layout.createSequentialGroup()
                .addContainerGap()

                    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
                     Alignment.LEADING)

                    .addComponent(jSeparator2, javax.swing.GroupLayout.
                     DEFAULT_SIZE, 219, Short.MAX_VALUE)

                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
                Alignment.TRAILING, false)

                    .addComponent(jLabel2, javax.swing.GroupLayout.
                     Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel5, javax.swing.GroupLayout.Alignment.LEADING)
```

```
                .addComponent(jRadioButton4,
                 javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jRadioButton5,

                 javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jRadioButton1,

                 javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jRadioButton2,

                 javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jRadioButton3,

                 javax.swing.GroupLayout.Alignment.LEADING)


                .addComponent(jLabel6, javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel7, javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel8, javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
                 jPanel2Layout.createSequentialGroup()
                   .addGap(76, 76, 76)

                  .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
                  Alignment.TRAILING)

.addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(jComboBox5,
                         javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE,
                         javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addGap(15, 15, 15)
                        .addComponent(jComboBox3,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(jPanel2Layout.createSequentialGroup()
```

```
                    .addComponent(jComboBox1,
                     javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addGap(14, 14, 14)
                    .addComponent(jComboBox2,
                     javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.PREFERRED_SIZE)
                   )
                )
             )
          )
       )
       .addContainerGap()
      )
);


jPanel2Layout.setVerticalGroup(
   jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
   Alignment.LEADING)

   .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
   jPanel2Layout.createSequentialGroup()
      .addContainerGap(32, Short.MAX_VALUE)
      .addComponent(jLabel2)
      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jLabel3)
      .addGap(14, 14, 14)
      .addComponent(jLabel5)

      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jRadioButton4)
      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jRadioButton5)
      .addGap(22, 22, 22)
      .addComponent(jLabel4)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jRadioButton1)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jRadioButton2)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jRadioButton3)
.addGap(23, 23, 23)

.addComponent(jSeparator2, javax.swing.GroupLayout.
PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jLabel6)
.addGap(14, 14, 14)
.addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)

    .addComponent(jComboBox2, javax.swing.GroupLayout.
     PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
     javax.swing.GroupLayout.PREFERRED_SIZE)

    .addComponent(jComboBox1, javax.swing.GroupLayout.
    PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
   )

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jLabel8)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
 Alignment.BASELINE)

    .addComponent(jComboBox3, javax.swing.GroupLayout.
    PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)

    .addComponent(jComboBox5, javax.swing.GroupLayout.
    PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
       )
```

```
            .addGap(52, 52, 52)
        )
    );

    jPanel3.setBorder(javax.swing.BorderFactory.createEtchedBorder()
    );
    jLabel9.setText(" Para generar un Gr\u00e1fico, haz clic ...");

    jButton6.setText("GR\u00c4FICO");

    jButton6.addMouseListener(new java.awt.event.MouseAdapter() {
       public void mousePressed(java.awt.event.MouseEvent evt) {
          jButton6MousePressed(evt);
       }
    }
);



    javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
    jPanel3.setLayout(jPanel3Layout);
    jPanel3Layout.setHorizontalGroup(

    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
       .addGroup(jPanel3Layout.createSequentialGroup()

    .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)
              .addGroup(jPanel3Layout.createSequentialGroup()
                 .addGap(29, 29, 29)
                 .addComponent(jLabel9)
                )

              .addGroup(jPanel3Layout.createSequentialGroup()
                 .addGap(72, 72, 72)
                 .addComponent(jButton6)
                )
        )
         .addContainerGap(36, Short.MAX_VALUE)
       )
    );
```

```
 jPanel3Layout.setVerticalGroup(

         jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.
         Alignment.LEADING)
         .addGroup(jPanel3Layout.createSequentialGroup()
            .addGap(21, 21, 21)
            .addComponent(jLabel9)

            .addGap(18, 18, 18)
            .addComponent(jButton6)

            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
             Short.MAX_VALUE)
          )
       );

      jSplitPane1.setDividerLocation(150);
      jSplitPane1.setDividerSize(10);
      treeToc1.setMap(map1);

      treeToc1.setMaximumSize(new java.awt.Dimension(74, 44)
       );

      jScrollPane1.setViewportView(treeToc1);

      jSplitPane1.setLeftComponent(jScrollPane1);

      map1.addMouseListener(new java.awt.event.MouseAdapter() {
         public void mouseClicked(java.awt.event.MouseEvent evt) {
            map1MouseClicked(evt);
         }
      }
  );

layer1.setDataset("com.esri.mo2.src.file.FileSystemConnection!
                        C:/Estaciones_Hidrometricas.shp!");
      javax.swing.GroupLayout layer1Layout = new javax.swing.GroupLayout(layer1);
      layer1.setLayout(layer1Layout);
      layer1Layout.setHorizontalGroup(

layer1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGap(0, 170, Short.MAX_VALUE)
      );
```

```
        layer1Layout.setVerticalGroup(

            layer1Layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
           .addGap(0, 230, Short.MAX_VALUE)
        );
        layer1.setBounds(10, 20, 170, 230);
        map1.add(layer1, javax.swing.JLayeredPane.DEFAULT_LAYER);

        jSplitPane1.setRightComponent(map1);

        scaleBar1.setMap(map1);

        Archivo.setMnemonic('A');
        Archivo.setText("Archivo");
        Abrir.setMnemonic('A');
        Abrir.setText("Abrir");
        Archivo.add(Abrir);

        Archivo.add(jSeparator1);

        Cerrar.setMnemonic('C');
        Cerrar.setText("Cerrar");
        Cerrar.addMouseListener(new java.awt.event.MouseAdapter() {
           public void mousePressed(java.awt.event.MouseEvent evt) {
              CerrarMousePressed(evt);
           }
        }
);

        Archivo.add(Cerrar);

        jMenuBar1.add(Archivo);

        Edicion.setMnemonic('E');
        Edicion.setText("Edicion");
        Copiar.setMnemonic('C');
        Copiar.setText("Copiar");
        Edicion.add(Copiar);

        Cortar.setMnemonic('t');
        Cortar.setText("Cortar");
        Edicion.add(Cortar);
```

```
        Pegar.setMnemonic('P');
        Pegar.setText("Pegar");
        Edicion.add(Pegar);

        jMenuBar1.add(Edicion);

        Herramientas.setMnemonic('H');
        Herramientas.setText("Herramientas");
        Barraherram.setText("Barra de Herramientas");
        Herramientas.add(Barraherram);

        jMenuBar1.add(Herramientas);

        Ayuda.setMnemonic('y');
        Ayuda.setText("Ayuda");
        Ayudas.setMnemonic('A');
        Ayudas.setText("Ayuda");
        Ayudas.addActionListener(new java.awt.event.ActionListener() {
           public void actionPerformed(java.awt.event.ActionEvent evt) {
              Ayudaayuda(evt);
           }
        }
    );

        Ayuda.add(Ayudas);

        Acerca.setMnemonic('c');
        Acerca.setText("Acerca de ...");
        Acerca.addActionListener(new java.awt.event.ActionListener() {
           public void actionPerformed(java.awt.event.ActionEvent evt) {
              Acercaacerca(evt);
           }
        }
);


        Ayuda.add(Acerca);

        jMenuBar1.add(Ayuda);

        setJMenuBar(jMenuBar1);
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane()
);

getContentPane().setLayout(layout);
layout.setHorizontalGroup(
   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(layout.createSequentialGroup()
      .addContainerGap()

      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
       Alignment.LEADING)
         .addComponent(scaleBar1, javax.swing.GroupLayout.
          DEFAULT_SIZE, 755, Short.MAX_VALUE)

         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
         layout.createSequentialGroup()

            .addComponent(jSplitPane1, javax.swing.GroupLayout.
             DEFAULT_SIZE, 506, Short.MAX_VALUE)

             .addPreferredGap(javax.swing.LayoutStyle.
             ComponentPlacement.RELATED)

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING, false)
               .addComponent(jPanel3,
               javax.swing.GroupLayout.Alignment.TRAILING,
               javax.swing.GroupLayout.DEFAULT_SIZE,
               javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.
               .addComponent(jPanel2,
               javax.swing.GroupLayout.Alignment.TRAILING,
               javax.swing.GroupLayout.DEFAULT_SIZE,
               javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            )
       )

         .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
         javax.swing.GroupLayout.DEFAULT_SIZE,
         javax.swing.GroupLayout.PREFERRED_SIZE)
      )
      .addContainerGap()
   )
);
```

```
layout.setVerticalGroup(
   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

   .addGroup(layout.createSequentialGroup()
      .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
       javax.swing.GroupLayout.PREFERRED_SIZE)
      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
      Alignment.LEADING)

         .addGroup(layout.createSequentialGroup()

            .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.
             ComponentPlacement.RELATED)

            .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
             javax.swing.GroupLayout.DEFAULT_SIZE,
             javax.swing.GroupLayout.PREFERRED_SIZE)
          )

         .addComponent(jSplitPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
         533, Short.MAX_VALUE)
        )

      .addGap(6, 6, 6)

      .addComponent(scaleBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
      .addContainerGap()
    )
);


pack();


java.awt.Dimension screenSize =
 java.awt.Toolkit.getDefaultToolkit().getScreenSize();
java.awt.Dimension dialogSize = getSize();
```

```
    setLocation((screenSize.width-dialogSize.width)/2,
                    (screenSize.height-dialogSize.height)/2);
                }// </editor-fold>

private void Ayudaayuda(java.awt.event.ActionEvent evt) {
    Ayuda ay = new Ayuda();
    ay.setVisible(true);
}

private void map1MouseClicked(java.awt.event.MouseEvent evt) {


}

private grabarras barras;


private void jButton6MousePressed(java.awt.event.MouseEvent evt) {

    String tempProm="Gráfico de Promedio de Temperatura del Aire";
    String tempMax="Gráfico de Máximo de Tempetatura del Aire";
    String tempMin="Gráfico de Mínimo de Temperatura del Aire";
    String lluvProm="Gráfico de Promedio de Llúvia Acumulada";
    String lluvMax="Gráfico de Máximo de Llúvia Acumulada";
    String lluvMin="Gráfico de Mínimo de Llúvia Acumulada";
    int n=9;


    String medTemp="Grados Centigrados ºC";
    String medLluv="Precipitación Acumulada  mm";
    String titulo="";
    String medida="";
    float val1 = 29, val2=30, val3=31, val4=32, val5=33, val6=34, val7=33, val8=32,
          val9=31, val10=31, val11=30, val12=29;
    float limsup=40, liminf=15;


    if (jRadioButton4.isSelected()==false && jRadioButton5.isSelected()==false){
    JOptionPane.showMessageDialog(null, "   Seleccione un Parametro", "Atencion",
    JOptionPane.ERROR_MESSAGE);
    barras.destroy();
    }
```

```
if (jRadioButton1.isSelected()==false && jRadioButton2.isSelected()==false &&
    jRadioButton3.isSelected()==false){
JOptionPane.showMessageDialog(null, "   Seleccione una Variable", "Atencion",
JOptionPane.ERROR_MESSAGE);
barras.destroy();
}


if (jRadioButton4.isSelected()==true && jRadioButton1.isSelected()==true){
medida= medTemp;
titulo= tempProm;
limsup=35;
liminf=15;
val1=31; val2=34; val3=31; val4=33; val5=30; val6=30; val7=29; val8=31; val9=32;
}


if (jRadioButton4.isSelected()==true && jRadioButton2.isSelected()==true){
medida= medTemp;
titulo= tempMax;
limsup=40;
liminf=15;
val1=35; val2=35; val3=34; val4=33; val5=34; val6=31; val7=34; val8=33; val9=34;
}


if (jRadioButton4.isSelected()==true && jRadioButton3.isSelected()==true){
medida= medTemp;
titulo= tempMin;
limsup=30;
liminf=10;
val1=26; val2=27; val3=26; val4=25; val5=24; val6=27; val7=23; val8=22; val9=24;
}


if (jRadioButton5.isSelected()==true && jRadioButton1.isSelected()==true){
medida= medLluv;
titulo= lluvProm;
limsup=1400;
liminf=0;
val1=40; val2=60; val3=305; val4=512; val5=537; val6=658; val7=841; val8=1050;
val9=1359;
}
```

```
if (jRadioButton5.isSelected()==true && jRadioButton2.isSelected()==true){
medida= medLluv;
titulo= lluvMax;
limsup=2000;
liminf=0;
val1=60; val2=105; val3=391; val4=723; val5=899; val6=1023; val7=1357;
val8=1725; val9=1946;
}


if (jRadioButton5.isSelected()==true && jRadioButton3.isSelected()==true){
medida= medLluv;
titulo= lluvMin;
limsup=1000;
liminf=0;
val1=5; val2=36; val3=99; val4=254; val5=324; val6=458; val7=621; val8=734;
val9=953;
}


barras = new grabarras ();
barras.parametro = titulo;
barras.medida = medida;
barras.n=n;
barras.val1=val1;
barras.val2=val2;
barras.val3=val3;
barras.val4=val4;
barras.val5=val5;
barras.val6=val6;
barras.val7=val7;
barras.val8=val8;
barras.val9=val9;


barras.limsup=limsup;
barras.liminf=liminf;

barras.init();
barras.start();

//consulta con1 = new consulta ();

}
```

```
private void jButton5MousePressed(java.awt.event.MouseEvent evt) {

    //manda la seleccion tipo poligono
    select.setRubberBand(brpolygon);

    FeatureLayer flayer = (FeatureLayer) map1.getLayer(0);
    flayer.setSelectionColor(Color.MAGENTA);
    select.setLayer(flayer);
    map1.setSelectedTool(select);

}

private void jButton4MousePressed(java.awt.event.MouseEvent evt) {

    //manda la seleccion tipo rectangulo
    select.setRubberBand(brenvelope);

    FeatureLayer flayer = (FeatureLayer) map1.getLayer(0);
    flayer.setSelectionColor(Color.MAGENTA);
    select.setLayer(flayer);
    map1.setSelectedTool(select);

}

private void jButton3MousePressed(java.awt.event.MouseEvent evt) {

    //manda la seleccion tipo circulo
    select.setRubberBand(brcircle);

    FeatureLayer flayer = (FeatureLayer) map1.getLayer(0);
    flayer.setSelectionColor(Color.MAGENTA);
    select.setLayer(flayer);
    map1.setSelectedTool(select);

}
```

```
    private void jButton2MousePressed(java.awt.event.MouseEvent evt) {

        //manda la seleccion tipo linea
        select.setRubberBand(brpolyline);

        FeatureLayer flayer = (FeatureLayer) map1.getLayer(0);
        flayer.setSelectionColor(Color.MAGENTA);
        select.setLayer(flayer);
        map1.setSelectedTool(select);

    }

    private void jButton1MousePressed(java.awt.event.MouseEvent evt) {

        //manda la seleccion tipo punto
        select.setRubberBand(brpoint);

        FeatureLayer flayer = (FeatureLayer) map1.getLayer(0);
        flayer.setSelectionColor(Color.MAGENTA);
        select.setLayer(flayer);
        map1.setSelectedTool(select);

    }

    private void Acercaacerca(java.awt.event.ActionEvent evt) {
        Acercade a = new Acercade();
        a.setVisible(true);

    }

    private void CerrarMousePressed(java.awt.event.MouseEvent evt) {
// Cierra la aplicación existente
        System.exit(0);
    }



    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Principal().setVisible(true);
            }
        }
      );
    }
```

```
    // Variables declaration
private javax.swing.JMenuItem Abrir;
private javax.swing.JMenuItem Acerca;
private javax.swing.JMenu Archivo;
private javax.swing.JMenu Ayuda;
private javax.swing.JMenuItem Ayudas;

private javax.swing.JMenuItem Barraherram;

private javax.swing.JMenuItem Cerrar;
private javax.swing.JMenuItem Copiar;
private javax.swing.JMenuItem Cortar;

private javax.swing.JMenu Edicion;

private javax.swing.JMenu Herramientas;

private javax.swing.JMenuItem Pegar;

private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;

private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;

private javax.swing.JComboBox jComboBox1;
private javax.swing.JComboBox jComboBox2;
private javax.swing.JComboBox jComboBox3;
private javax.swing.JComboBox jComboBox5;

private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
```

```
    private javax.swing.JMenuBar jMenuBar1;

    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;

    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JRadioButton jRadioButton3;
    private javax.swing.JRadioButton jRadioButton4;
    private javax.swing.JRadioButton jRadioButton5;

    private javax.swing.JScrollPane jScrollPane1;

    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;

    private javax.swing.JSplitPane jSplitPane1;

    private javax.swing.JToolBar jToolBar1;

    private com.esri.mo2.ui.bean.Layer layer1;
    private com.esri.mo2.ui.tb.LayerToolBar layerToolBar1;

    private com.esri.mo2.ui.bean.Map map1;
    private com.esri.mo2.ui.tb.ProjectToolBar projectToolBar1;

    private com.esri.mo2.ui.bean.ScaleBar scaleBar1;
    private com.esri.mo2.ui.toc.TreeToc treeToc1;
    private com.esri.mo2.ui.tb.ZoomPanToolBar zoomPanToolBar1;

    // End of variables declaration

}
```

CLASE CONEXIÓN

```
package conexion;

import javax.swing.*;

public class conexion {

   public conexion () {
   }

     public static void main(String[] args) {
     consulta con = new consulta ();
     con.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      }
}
```

CLASE CONSULTA

```
package conexion;

import java.awt.*;
import java.sql.*;
import java.util.*;
import javax.swing.*;


public class consulta extends JFrame{

  static final String CONTROLADOR_JDBC = "org.postgresql.Driver";
  static final String URL_BASEDEDATOS = "jdbc:postgresql://localhost:5432/snet";
  String usr = "postgres";
  String pwd = "postgres";

  // Declaracion de objetos Connection y Statement para acceder a
  // la base de datos y realizar consultas

  private Connection conexion;
  private Statement instruccion;

  // el constructor se conecta a la base de datos, para realizar la consulta, procesa
  // los resultados y los muestra en la ventana

  public consulta ()
  {
    super( "Conexión a Base de Datos" );


    // conectarse a la base de datos y consultar la base de datos

    try {

      // cargar clase de controlador de base de datos

      Class.forName( CONTROLADOR_JDBC );
```

```
// establecer conexión a la base de datos

conexion = DriverManager.getConnection( URL_BASEDEDATOS, usr, pwd );

// crear objeto Statement para consultar la base de datos

instruccion = conexion.createStatement();

// consultar la base de datos

ResultSet conjuntoResultados =
  instruccion.executeQuery( "SELECT sum (medicion) from mediciones where
  estacionid=33 and parametroid='AT'" );



// procesar los resultados de la consulta

StringBuffer resultados = new StringBuffer();
ResultSetMetaData metaDatos = conjuntoResultados.getMetaData();
int numeroDeColumnas = metaDatos.getColumnCount();

for ( int i = 1; i <= numeroDeColumnas; i++ )
  resultados.append( metaDatos.getColumnName( i ) + "\t" );

resultados.append( "\n" );

while ( conjuntoResultados.next() ) {

  for ( int i = 1; i <= numeroDeColumnas; i++ )
    resultados.append( conjuntoResultados.getObject( i ) + "\t" );

  resultados.append( "\n" );
}

// configurar GUI y ventana para mostrar el resultado

JTextArea areaTexto = new JTextArea( resultados.toString()
 );
Container contenedor = getContentPane();

contenedor.add( new JScrollPane( areaTexto )
 );
```

```
      setSize( 320, 130 );  // establecer el tamaño de la ventana
      setVisible( true );   // mostrar la ventana

    }  // fin de bloque try

    // detectar posibles problemas al interactuar con la base de datos

    catch ( SQLException excepcionSql ) {
      JOptionPane.showMessageDialog( null, excepcionSql.getMessage(),
        "Error en base de datos", JOptionPane.ERROR_MESSAGE );

      System.exit( 1 );
    }
       // detectar posibles problemas al cargar el controlador de la base de datos

    catch ( ClassNotFoundException claseNoEncontrada ) {
      JOptionPane.showMessageDialog( null, claseNoEncontrada.getMessage(),
        "No se encontró el controlador", JOptionPane.ERROR_MESSAGE );

      System.exit( 1 );
    }

    // asegurar que instruccion y conexion se cierren correctamente

    finally {

      try {
        instruccion.close();
        conexion.close();
      }

      // manejar posibles excepciones al cerrar instruccion y conexión

      catch ( SQLException excepcionSql ) {
        JOptionPane.showMessageDialog( null,
          excepcionSql.getMessage(), "Error en base de datos",
          JOptionPane.ERROR_MESSAGE );

        System.exit( 1 );
      }
    }

  }  // fin del constructor consulta

}  // fin de la clase consulta
```

CLASE GRABARRAS

```
/*
 * grabarras.java
 * Con este codigo se genera un grafico de barras
 * y ademas lo genera en un applet.
  */

package estadisticos;

import net.sourceforge.chart2d.Chart2D;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JApplet;
import javax.swing.JFrame;
import javax.swing.JTabbedPane;
import java.awt.Color;
import java.util.Random;

import net.sourceforge.chart2d.Chart2DProperties;
import net.sourceforge.chart2d.Dataset;
import net.sourceforge.chart2d.GraphChart2DProperties;
import net.sourceforge.chart2d.GraphProperties;
import net.sourceforge.chart2d.LBChart2D;
import net.sourceforge.chart2d.LegendProperties;
import net.sourceforge.chart2d.MultiColorsProperties;
import net.sourceforge.chart2d.Object2DProperties;


public class grabarras extends javax.swing.JApplet {


    private JFrame frame1 = null;
    private static boolean isApplet = true;
```

```
  Color c= new Color(77,64,238);

  public String parametro;
  public String medida;
  public int n;

  public float val1, val2, val3, val4, val5, val6, val7, val8, val9, val10, val11, val12;
  public float limsup, liminf;

  public static void main (String[] args) {
  isApplet = false;
  grabarras barras = new grabarras();
  barras.init();
  barras.start();

  }

public void init() {

  JTabbedPane panel = new JTabbedPane();
  panel.addTab ("Grafico de Barra", getChart2D()
 );

   boolean dynamicSizeCalc = false;
   if (dynamicSizeCalc) {
     int maxWidth = 0;
     int maxHeight = 0;

     for (int i = 0; i < panel.getTabCount(); ++i) {

       Chart2D chart2D = (Chart2D)panel.getComponentAt (i);
       chart2D.pack();
       Dimension size = chart2D.getSize();
       maxWidth = maxWidth > size.width ? maxWidth : size.width;
       maxHeight = maxHeight > size.height ? maxHeight : size.height;
      }

     Dimension maxSize = new Dimension (maxWidth, maxHeight);
     System.out.println (maxSize);
```

```
      for (int i = 0; i < panel.getTabCount(); ++i) {
        Chart2D chart2D = (Chart2D)panel.getComponentAt (i);
        chart2D.setSize (maxSize);
        chart2D.setPreferredSize (maxSize);
      }

      System.out.println (panel.getPreferredSize()
       );
    }

    else {
       Dimension maxSize = new Dimension (575, 225);
      for (int i = 0; i < panel.getTabCount(); ++i) {
        Chart2D chart2D = (Chart2D)panel.getComponentAt (i);
        chart2D.setSize (maxSize);
        chart2D.setPreferredSize (maxSize);
      }

      panel.setPreferredSize (new Dimension (566 + 5, 280 + 5)
      ); //+ 5 slop
    }

    frame1 = new JFrame();
    frame1.getContentPane().add (panel);
    frame1.setTitle ("SERVICIO NACIONAL DE ESTUDIOS TERRITORIALES");
    frame1.addWindowListener (
      new WindowAdapter() {
        public void windowClosing (WindowEvent e) {
          destroy();
        }
      }
  );

    frame1.pack();

    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

    frame1.setLocation (
      (screenSize.width - frame1.getSize().width) / 2,
      (screenSize.height - frame1.getSize().height) / 2);
  }
```

```
public void start() {
   frame1.setVisible(true);
 }

 public void destroy() {

   if (frame1 != null) frame1.dispose();
   if (!isApplet) System.exit (0);
 }


  public Chart2D getChart2D() {


   // Comienza la configuracion del grafico de barras

   //Configuracion de las propiedades del objeto

   Object2DProperties object2DProps = new Object2DProperties();
   object2DProps.setObjectTitleText (parametro);

   //Configuracion de las propiedades del Grafico

   Chart2DProperties chart2DProps = new Chart2DProperties();
   chart2DProps.setChartDataLabelsPrecision (-2);

   //Configuracion de las leyendas del Grafico

   LegendProperties legendProps = new LegendProperties();

   //sirve para eliminar una leyenda dentro de un grafico

   legendProps.setLegendExistence(false);

   // Sirve para crear leyendas dentro del grafico
  // String [] legendLabels = {"Temperatura"};
   //legendProps.setLegendLabelsTexts (legendLabels);

   //Configuracion de las propiedades del Grafico de Barras

   GraphChart2DProperties graphChart2DProps = new GraphChart2DProperties();
   String[] labelsAxisLabels = {"Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio",
   "Agosto", "Septiembre", "Octubre"};
```

```
graphChart2DProps.setLabelsAxisLabelsTexts (labelsAxisLabels);
graphChart2DProps.setLabelsAxisTitleText ("Rango de Meses");
graphChart2DProps.setNumbersAxisTitleText (medida);

graphChart2DProps.setChartDatasetCustomizeGreatestValue (true);
graphChart2DProps.setChartDatasetCustomGreatestValue (limsup);
graphChart2DProps.setChartDatasetCustomizeLeastValue (true);

graphChart2DProps.setChartDatasetCustomLeastValue (liminf);
graphChart2DProps.setGraphComponentsColoringByCat (false);
graphChart2DProps.setGraphComponentsColorsByCat (new MultiColorsProperties());

//Configuracion de las propiedades Graficas

GraphProperties graphProps = new GraphProperties();
graphProps.setGraphComponentsAlphaComposite(graphProps.
ALPHA_COMPOSITE_NONE);

//Configuracion del conjunto de datos a graficar

Dataset dataset = new Dataset (1, n, 1);
dataset.set (0, 0, 0, val1);
dataset.set (0, 1, 0, val2);
dataset.set (0, 2, 0, val3);
dataset.set (0, 3, 0, val4);
dataset.set (0, 4, 0, val5);
dataset.set (0, 5, 0, val6);
dataset.set (0, 6, 0, val7);
dataset.set (0, 7, 0, val8);
dataset.set (0, 8, 0, val9);


//Configuracion de los colores del Grafico

MultiColorsProperties multiColorsProps = new MultiColorsProperties();
multiColorsProps.setColorsCustomize (true);
multiColorsProps.setColorsCustom (new Color[] {c});

GraphProperties graphPropsTrend = new GraphProperties();
graphPropsTrend.setGraphBarsExistence (false);
graphPropsTrend.setGraphLinesExistence (true);
```

```
    //Configuracion completa del Grafico de Barras

    LBChart2D chart2D = new LBChart2D();
    chart2D.setObject2DProperties (object2DProps);
    chart2D.setChart2DProperties (chart2DProps);
    chart2D.setLegendProperties (legendProps);

    chart2D.setGraphChart2DProperties (graphChart2DProps);

    chart2D.addGraphProperties (graphProps);
    chart2D.addDataset (dataset);
    chart2D.addMultiColorsProperties (multiColorsProps);

    //Validacion opcional:  Prints debug messages if invalid only.

    if (!chart2D.validate (false)) chart2D.validate (true);


    return chart2D;
      }


}
```

CLASE ACERCADE

```
package otros;


public class Acercade extends javax.swing.JDialog {

    /** Creates new form Acercade */
    public Acercade (){

    initComponents();

    }

    public Acercade(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">

    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Acerca  De ...");
        setResizable(false);
        jLabel1.setFont(new java.awt.Font("Comic Sans MS", 1, 12)
         );

        jLabel1.setText("Herramienta de Consulta Geogr\u00e1fica");

        jLabel2.setFont(new java.awt.Font("Comic Sans MS", 0, 12)
         );
```

```
        jLabel2.setText("Desarrollado por :");

        jLabel3.setFont(new java.awt.Font("Comic Sans MS", 0, 12)
          );

        jLabel3.setText("Alba Elizabeth Flores Villalobos");

        jLabel4.setFont(new java.awt.Font("Comic Sans MS", 0, 12)
          );

        jLabel4.setText("Mario Alejandro Sanchez Garcia");

        jLabel5.setFont(new java.awt.Font("Comic Sans MS", 0, 12)
           );

        jLabel5.setText("2007, Derechos Reservados");

        jButton1.setText("Aceptar");

        jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButton1MouseClicked(evt);
            }
        }
    );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane()
         );

        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(33, 33, 33)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addGap(10, 10, 10)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(layout.createSequentialGroup()
                                        .addGap(34, 34, 34)
```

```
                                .addComponent(jLabel2)
                                 )

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jLabel4)
                                .addComponent(jLabel3))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(12, 12, 12)
                                .addComponent(jLabel5)
                                 )
                            )
                        )

                        .addComponent(jLabel1)
                         )
                    )

                .addGroup(layout.createSequentialGroup()
                    .addGap(97, 97, 97)
                    .addComponent(jButton1)
                     )
                )

            .addContainerGap(26, Short.MAX_VALUE)
          )
    );

    layout.setVerticalGroup(
       layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
       .addGroup(layout.createSequentialGroup()

            .addContainerGap()
            .addComponent(jLabel1)
            .addGap(15, 15, 15)

            .addComponent(jLabel2)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel3)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel4)
            .addGap(27, 27, 27)
```

```
                .addComponent(jLabel5)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                 17, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addContainerGap()
               )
        );

        pack();
        java.awt.Dimension screenSize =
        java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        java.awt.Dimension dialogSize = getSize();
        setLocation((screenSize.width-dialogSize.width)/2,(screenSize.height-
        dialogSize.height)/2);
    }// </editor-fold>

    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
        dispose();

    }


    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    // End of variables declaration

}
```

CLASE AYUDA

```
package otros;


public class Ayuda extends javax.swing.JDialog {

    /** Creates new form Ayuda */
    public Ayuda (){

    initComponents();

    }

    public Ayuda(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">

    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Ayuda de Aplicaci\u00f3n");
        setResizable(false);

        jLabel1.setText("Esta es la ventana de ayuda aqui, se colocar\u00e1n");

        jLabel2.setText("todas las instrucciones necesarias para que el ");
```

```
        jLabel3.setText("usuario final pueda utilizar debidamente dicha");

        jLabel4.setText("aplicacion, esta vnetana, sirve para aclarar ");

        jLabel5.setText("dudas que cualquier usuario pueda tener en");

        jLabel6.setText("cuanto al uso y manejo de la aplicaci\u00f3n");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                 Alignment.TRAILING, false)

                    .addComponent(jLabel6, javax.swing.GroupLayout.Alignment.LEADING,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(jLabel5, javax.swing.GroupLayout.Alignment.LEADING,
                     javax.swing.GroupLayout.DEFAULT_SIZE, 236, Short.MAX_VALUE)

                    .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.LEADING,
                     javax.swing.GroupLayout.DEFAULT_SIZE, 236, Short.MAX_VALUE)

                    .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.LEADING,
                     javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.LEADING,
                     javax.swing.GroupLayout.DEFAULT_SIZE, 236, Short.MAX_VALUE)

                    .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.LEADING,
                     javax.swing.GroupLayout.DEFAULT_SIZE, 236, Short.MAX_VALUE)
                     )

                .addGap(10, 10, 10)
            )
        );
```

```
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel3)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel4)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel5)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel6)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                 Short.MAX_VALUE)
             )
        );

        jButton1.setText("OK");
        jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButton1MouseClicked(evt);
            }
        }
);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane()
    );
        getContentPane().setLayout(layout);

        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
             layout.createSequentialGroup()
                .addContainerGap(23, Short.MAX_VALUE)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE,
                 javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap()
                 )
```

```
            .addGroup(layout.createSequentialGroup()
                .addGap(99, 99, 99)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
                 javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(119, Short.MAX_VALUE)
             )
        );

        layout.setVerticalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE,
                 javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
                .addContainerGap(41, Short.MAX_VALUE)
             )
        );

        pack();

    }// </editor-fold>

    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {

        dispose();

    }


    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration

}
```

CLASE PSGTOSHP

```
//Este es el codigo para crear un shapefile desde una bd
//utiliza ConsoleInputReader y ConsoleErrReader

package shp;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;

public class PsgToShp {

        public static void main(String[] args){


   try {

      //No es necesario crear las otras dos clases definidas!!!
      //ConsoleErrReader y ConsoleInputReader!!!

      //Aqui se ejecuta el comando pgsql2shp para crear el shapefile
      //que carga la pantalla principal.

      Process process =
      Runtime.getRuntime().exec(
      "C:/WINDOWS/system32/cmd.exe /c pgsql2shp -f
        c:/Estaciones_Hidrometricas -u postgres snet public.estaciones_hidrometricas");

      //Esto se puede ejecutar si en caso da problema:

      //InputStream inputStream = process.getInputStream();
      //InputStream errStream = process.getErrorStream();

      //ConsoleErrReader err = new ConsoleErrReader(errStream);
      //ConsoleInputReader input = new ConsoleInputReader(inputStream);

      //Thread th = new Thread(err);
      //Thread th1 = new Thread(input);
      //th.start();
      //th1.start();
        }
```

```
        catch(Exception e)   {
           e.printStackTrace();
          }

      }
}
```

CLASE CONSOLE INPUTREADER

```
package shp;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;



class ConsoleInputReader implements Runnable
{
	InputStream in;
	ConsoleInputReader(InputStream in)
	{
	   this.in=in;
	}
	public void run() {
	   try
	   {
	      BufferedReader bufferreader = new BufferedReader(new
	            InputStreamReader(in));
	      String line = bufferreader.readLine();
	      while(line!=null)
	      {
	         System.out.println(line);
	         line=bufferreader.readLine();
	      }
	   }
	   catch(Exception e)
	   {
	      e.printStackTrace();
	   }
	}
}
```

CLASE CONSOLE ERRREADER

```
package shp;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;

class ConsoleErrReader implements Runnable
   {
      InputStream in;
      ConsoleErrReader(InputStream in)
      {
         this.in=in;
      }

      public void run() {
         try {
            System.out.println("err reader run");
            BufferedReader bufferreader = new BufferedReader(new
                  InputStreamReader(in));
            String line = bufferreader.readLine();
            while (line != null) {
               System.out.println(line);
               line = bufferreader.readLine();
            }
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
   }
```