Tabla de Contenido

Lista de Figuras

Anexo A. PRACTICAS

Este Anexo contiene practicas de las bases tecnológicas utilizadas en el proyecto, donde se indica la configuración de herramientas utilizadas en el proyecto tales como el Servidor de Mapas Geoserver, agente SNMP en DEBIAN, utilización de herramientas como OpenLayers, PostGIS y PostgreSQL.

Como primera medida se va a abarcar las herramientas relacionadas con el desarrollo de la entidad gestora, después las referentes a la entidad gestionada, en este caso el contador.

1. PRACTICAS ENTIDAD GESTORA

Para Tratar el tema de Sistemas de Información Geográfica primero que todo se necesita conocer que los objetos GIS soportados por PostGIS, son de características simples definidas por OpenGIS. Actualmente PostGIS soporta las características y el API de representación de la especificación OpenGIS, pero no tiene varios de los operadores de comparación y convolución de esta especificación [1].

Ejemplos de la representación en modo texto:

- POINT(0 0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0))
- MULTIPOINT(0 0 0,1 2 1)
- MULTILINESTRING((0 0 0,1 1 0,1 2 1),(2 3 1,3 2 1,5 4 1))
- MULTIPOLYGON(((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0)),((-1 -1 0,-1 -2 0,-2 -2 0,-2 -1 0,-1 -1 0)))
- GEOMETRYCOLLECTION(POINT(2 3 9),LINESTRING((2 3 4,3 4 5))

Antes de realizar cualquier acción dentro de la base de datos hay que Instalar PostGIS para PostgreSQL. En la carpeta donde está instalado Postgres8.3 se abre la Aplicación Stack Builder [2]. Se le da click en Next (siguiente).

1.2. Instalación de PostGIS

Figura 1. Configuración Stack Builder1

Habiendo seleccionado dentro de la opción Spatial Extensions, la opción PostGIS, se da click en Next

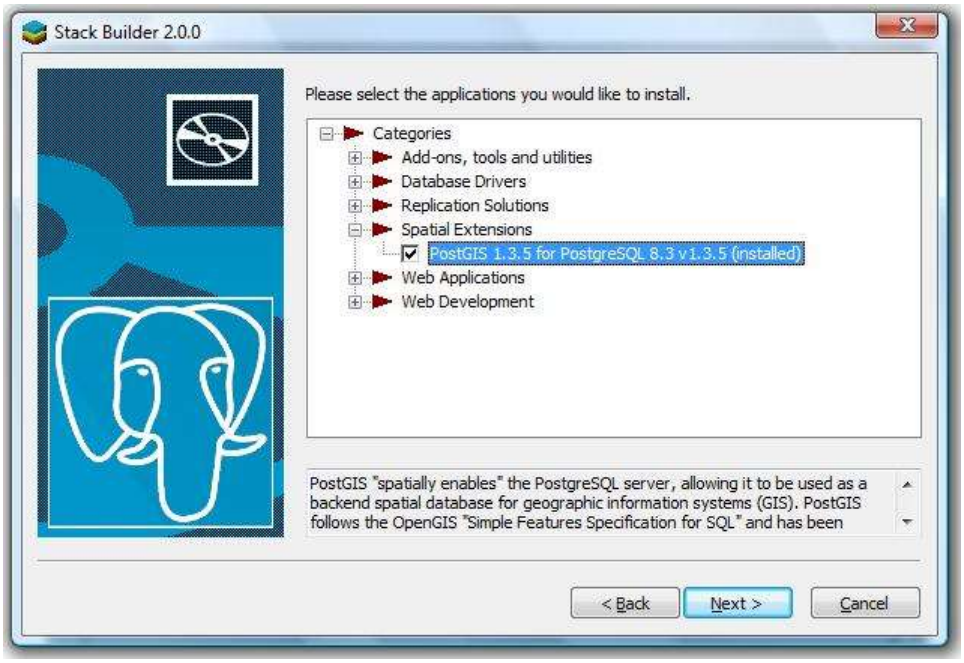

Figura 2. Configuración Stack Builder2

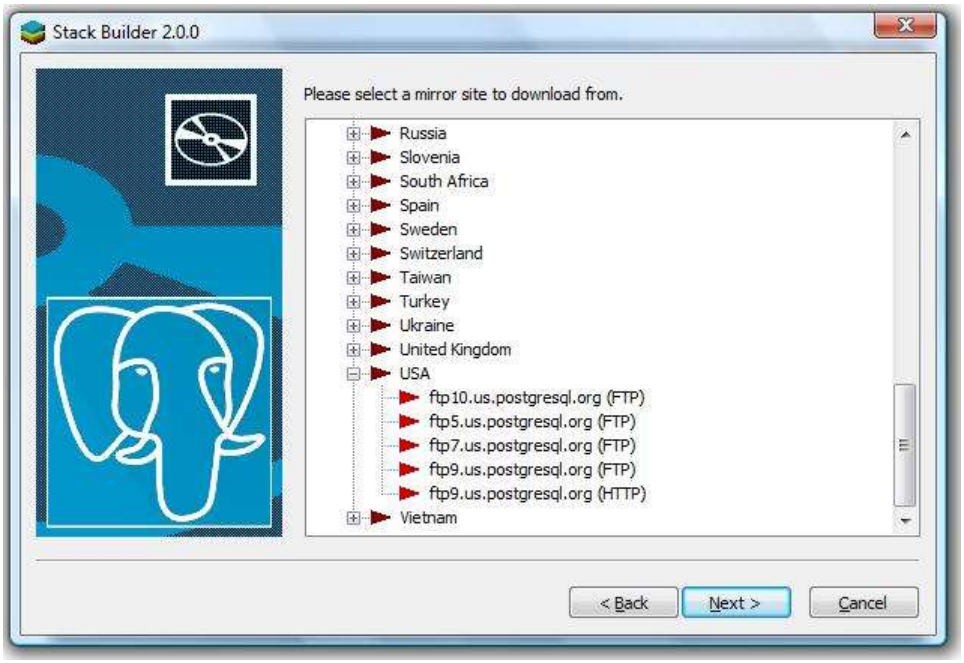

Figura 3. Configuración Stack Builder3

Se escoge un servidor para descargar PostGIS y se da click en Next, enseguida se inicia la descarga, una vez completada aparece la siguiente ventana.

Figura 4. Configuración Stack Builder4

Ahora se da click en Next y se procede a instalar PostGIS para PostgreSQL 8.3.

1.3. Creación de una Base de Datos Espacial

Una vez instalado se crea una base de datos espacial abriendo pgAdminIII el gestor de bases de datos de PostgreSQL

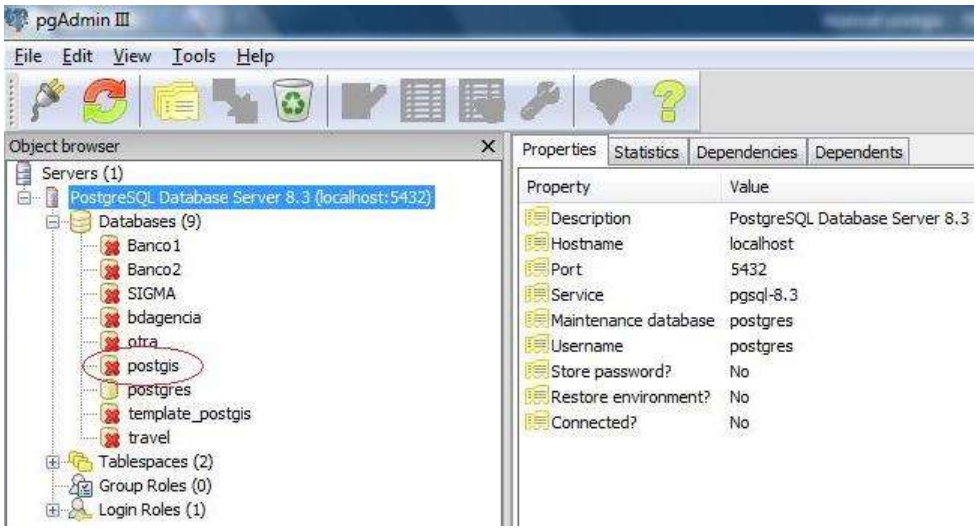

Figura 5. Configuración pgAdmin1

Se puede notar que hay una base de datos creada por defecto llamada postgis, esto indica que la instalación se ha realizado correctamente, ahora se pueden crear las bases de datos espaciales. Dando click derecho sobre “Databases” se crea una nueva base de datos.

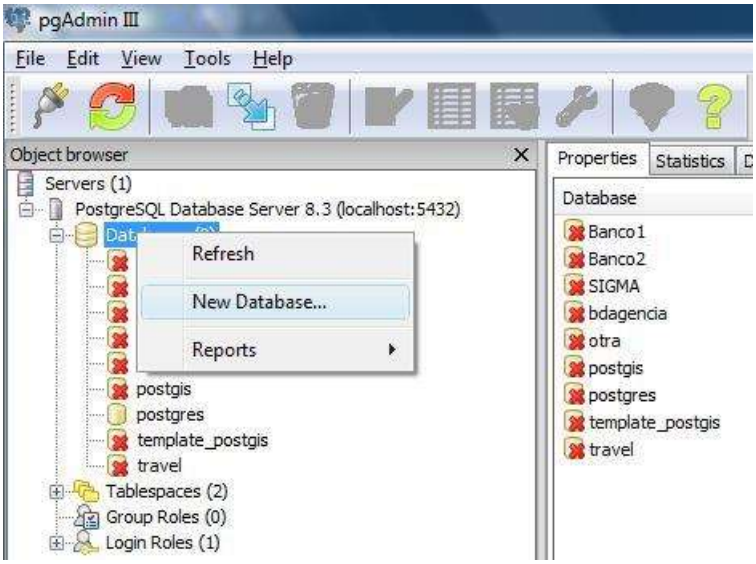

Figura 6. Configuración pgAdmin2

Ahora se llenan los datos necesarios, en este caso el nombre de la base de datos es “SIGMApostgis” y en el campo Template se escoge la opción postgis y se da click en ok.

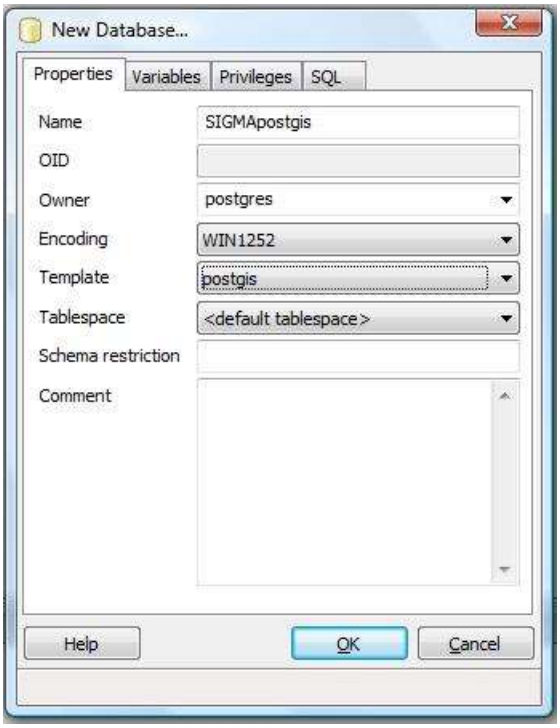

Figura 7. Configuración pgAdmin3

1.4. Crear tablas espaciales.

Para crear una tabla con datos espaciales se realizan dos pasos [3]:

1. Creación de una tabla no espacial.

   Ejemplo: CREATE TABLE contadores_suspendidos (matricula_predio)

2. Adición de una columna(campo) espacial a la tabla usando la función

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

   Ejemplo:

SELECT AddGeometryColumn('SIGMApostgis', 'contadores_suspendidos', 'suspendido_geom', 21891, 'POINT', 2)

- Creando una tabla a partir de otra [4].

  En este caso tenemos una tabla llamada predios y otra llamada cuentas, la tabla cuentas tiene un campo llamado estado que guarda el estado del servicio de energía eléctrica, por lo tanto vamos a crear una tabla con los identificadores de los predios correspondientes a las cuentas con el servicio suspendido.

  CREATE TABLE contadores_suspendidos AS SELECT predios.matricula_predio AS matricula_predio FROM predios, cuentas WHERE cuentas.estado ILIKE 'desconectado' AND cuentas.matricula_predio ILIKE predios.matricula_predio

  Ahora añadimos el campo espacial de nuestra nueva tabla, el cual va a contener los puntos georeferenciados para cada contador.

  SELECT AddGeometryColumn('SIGMApostgis', 'contadores_suspendidos', 'suspendido_geom', 21891, 'POINT', 2)

  Como creamos esta tabla a partir de otra que ya tiene registros tenemos una tabla que contiene dos columnas, una con las matrículas de los predios y otra con los puntos la cual se encuentra vacía, por lo que podemos llenarla actualizando la ubicación espacial de estos puntos con la siguiente sentencia.

  UPDATE contadores_suspendidos SET suspendido_geom = GeometryFromText('POINT((<COORDENADAX> <COORDENADAY>)', 21891) WHERE contadores_suspendidos.matricula_predio = '<MATRICULAPREDIO>'

  Donde <COORDENADAX> y <COORDENADAY> son las coordenadas geográficas del predio y <MATRICULAPREDIO> el número de la matrícula inmobiliaria.

1.5. Cargar cartografía en formato shape en la base de datos.

Para cargar datos que se encuentran en formato shape a la base de datos se utiliza el cargador de datos “shp2pgsql”, éste convierte archivos de figuras ESRI a SQL para su inserción en una base de datos PostGIS/PostgreSQL, que es nuestro caso.

Para mayor agilidad se copian los siguientes archivos ubicados en la carpeta bin de postgres a una carpeta que se crea en “C:/” llamada “pgutils”.

comerr32.dll	krb5_32.dll	libeay32.dll	libiconv-2.dll	libintl3.dll
libpq.dll	pgsql2shp.exe	psql.exe	pg_dump.exe	pg_restore.exe
shp2pgsql.exe	ssleay32.dll			

Se abre símbolo del sistema y se entra a la carpeta que contiene la cartografía para convertir los “shapefiles” que es el formato vectorial en el que está la cartografía a “sql”, para poder ser cargados en la base de datos, con la siguiente sentencia [5]:

c:\pgutils\shp2pgsql -s 21891 pop_urbano.shp pop_urbano > pop_urbano.sql

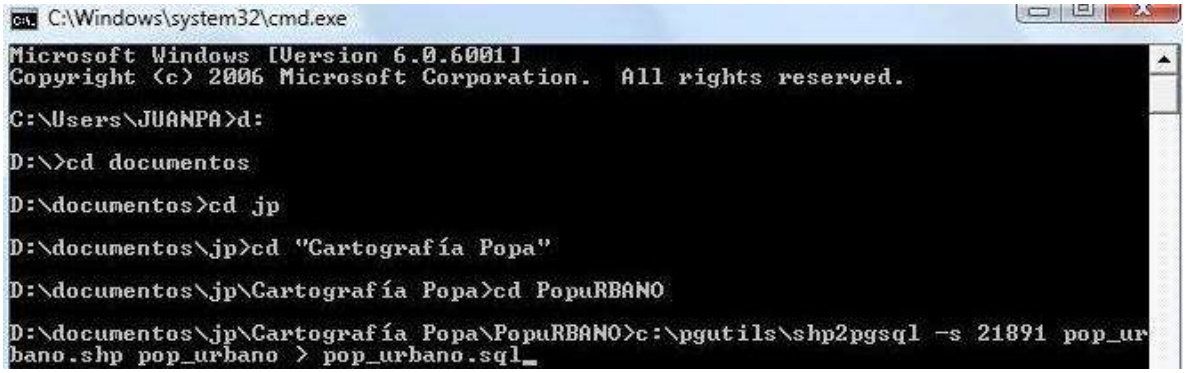

Figura 8. Cargar Cartografía

Debe aparecer:

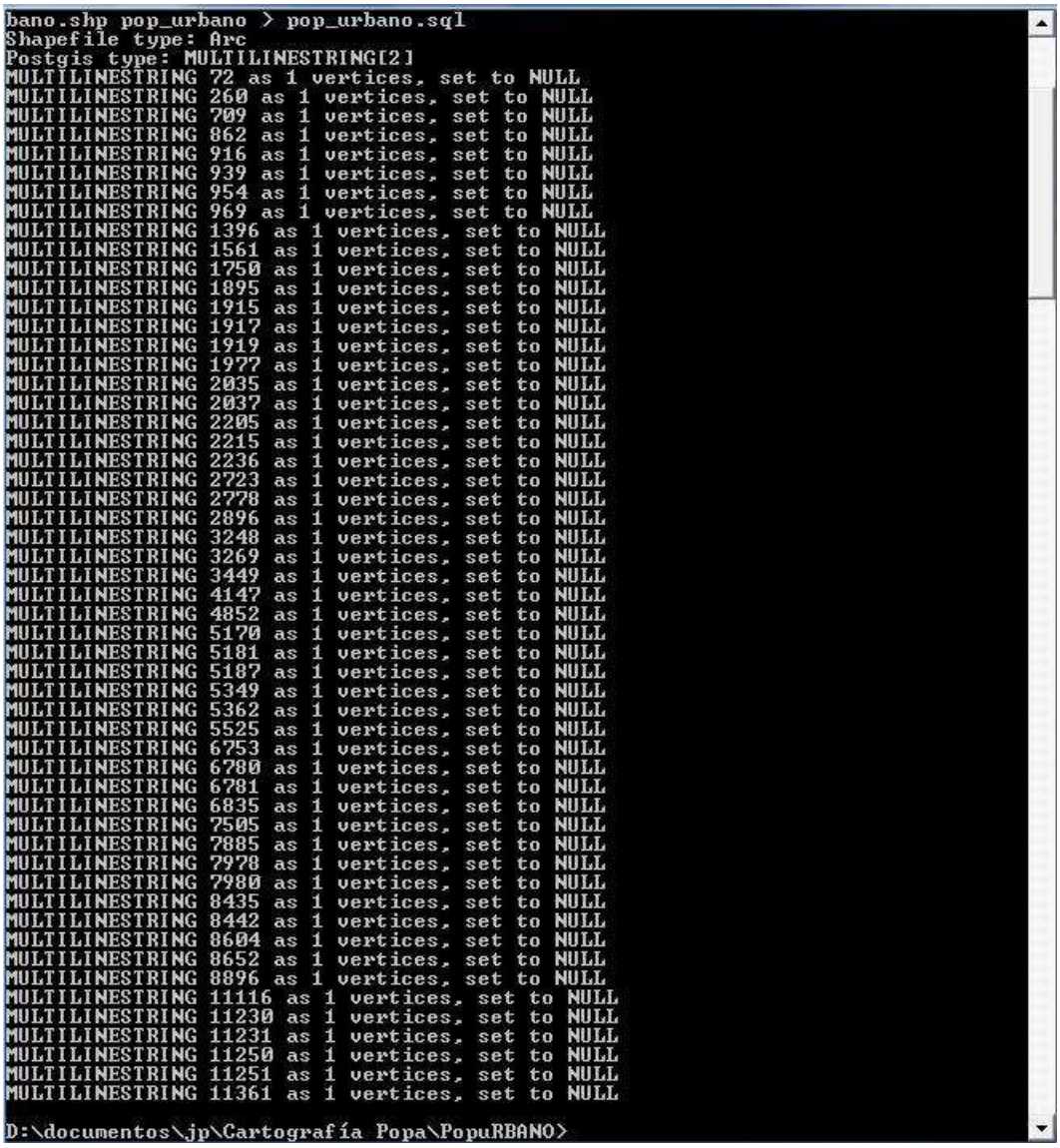

Figura 9. Resultados Cargar Cartografía

Se hace lo mismo con la capa de las comunas con la línea:

c:\pgutils\shp2pgsql -s 21891 popaCo.shp popaCo > popaCo.sql

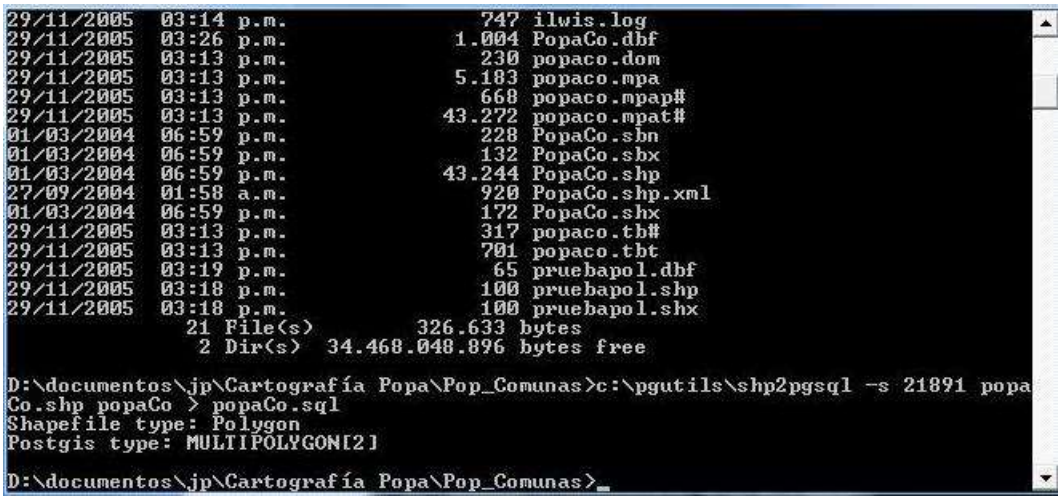

Figura 10. Cargar Capas SIG

Ahora con la cartografía convertida a formato “sql”, se ingresan a la base de datos espacial creada.

Se copian los archivos sql en la carpeta bin de postgreSQL y estando ubicados en la carpeta “bin” en símbolo de sistema se ejecuta la siguiente sentencia:

psql -d SIGMApostgis -h localhost -U postgres -f pop_urbano.sql

Figura 11. Ingresar a la Base de Datos

Se digita la clave para el usuario postgres, la cual fue definida en la instalación de PostgreSQL y se presiona la tecla enter [6].

Debe aparecer lo siguiente:

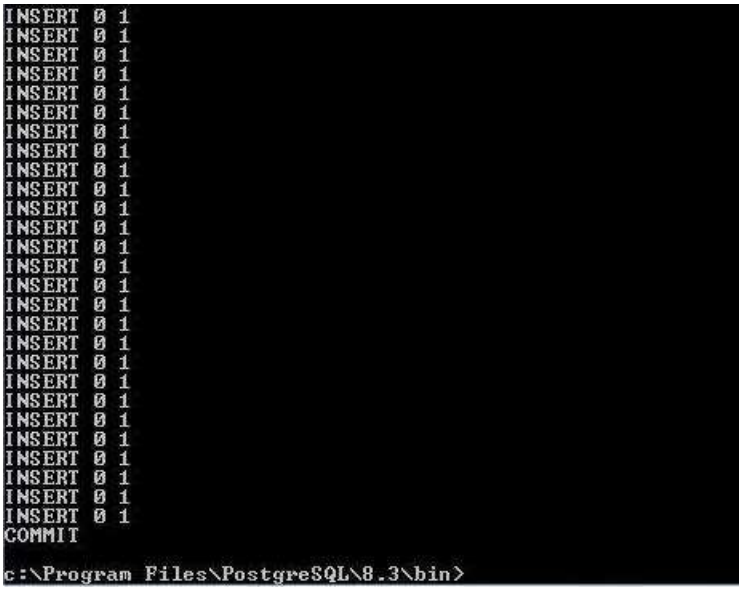

Figura 12Resultados Ingreso de datos a la BD

Hacemos lo mismo para popaCo.sql

psql -d SIGMApostgis -h localhost -U postgres -f popaCo.sql

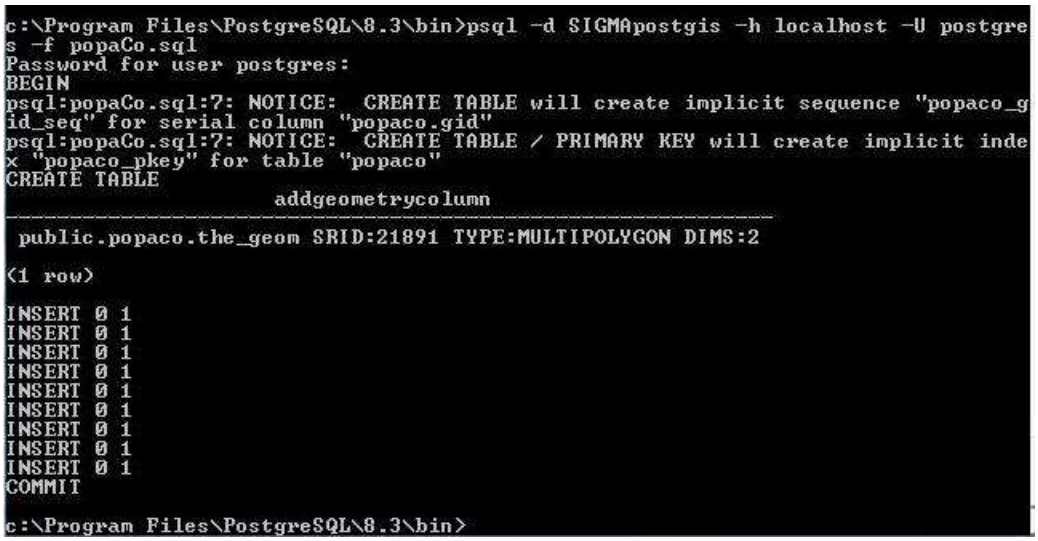

Figura 13. Ingresar Datos BD

1.6. Crear Almacen de Datos y Entidades para la publicación en el Servidor de Mapas.

Para esta práctica debemos instalar GeoServer, el cual se puede descargar desde la página: http://sourceforge.net/project/showfiles.php?group_id=25086 su instalación es muy intuitiva por lo que se va a detallar este procedimiento. Ahora, una vez instaldo se ejecuta Start GeoServer con el cual se inicia Geoserver.

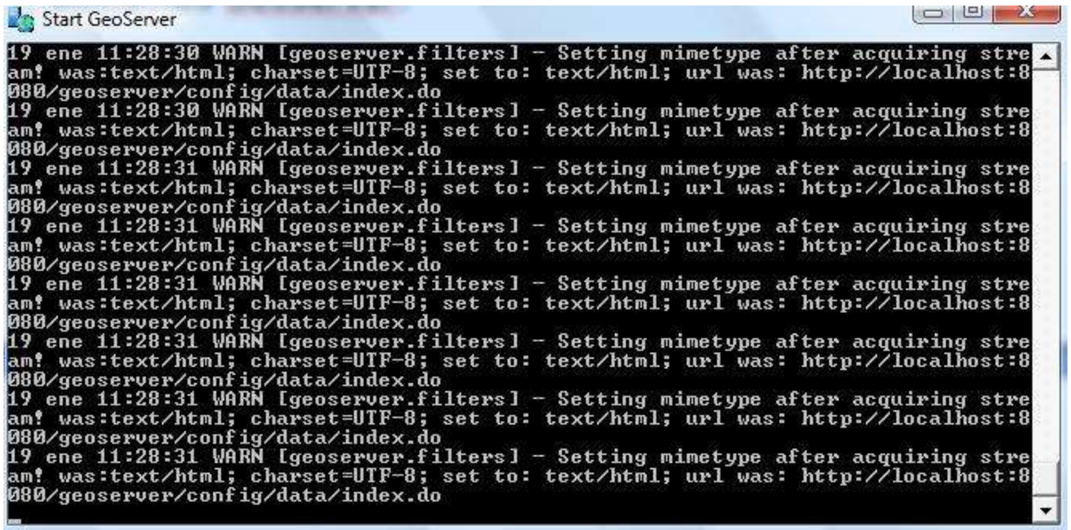

Figura 14. Iniciar GeoServer

En un navegador escribimos: http://localhost:8080/geoserver/

Enseguida debe aparecer la página de Bienvenida de GeoServer, ahora se da click en "Iniciar Sesión".

Figura 15. Bienvenido a GeoServer

Para iniciar sesión, el nombre de usuario por defecto es “admin” y la contraseña “geoserver”.

Figura 16. GeoServidor de GeoServer

Ahora se puede usar Geoserver, debe crearse un Almacén de Datos para la cartografía, se necesita decirle a Geoserver qué datos tenemos y donde están almacenados.

Se va a “Configuración”, “Datos”, “Almacenes”.

Figura 17. Configuración GeoServer1

Figura 18. Configuración GeoServer2

Figura 19. Configuración GeoServer3

Ahora se crea el almacén dando click en “nuevo”

Figura 20. Configuración almacén Datos GeoServer1

En la descripción de los datos se elige la opción Postgis y para el identificador del almacén se un nombre, en este caso “popa”. Se da click en “Nuevo” para ingresar la información referente al nuevo almacén [7].

Figura 21. Nuevo Almacén de Datos GeoServer

- En el campo “host” va la dirección ip donde se encuentra la base de datos Postgis, en este caso está ubicada localmente, es decir “localhost”.
- En “port”, el puerto por defecto en el que trabaja la base de datos postgres que es “5432”.
- En “database”, va el nombre de la base de datos en donde está almacenada la cartografía, que es “SIGMApostgis”.

- “user”, El usuario de la base de datos, que es “postgres”.
- “passwd”, contraseña del usuario postgres.

Se da click en el botón Enviar.

Figura 22. Editor almacén Datos GeoServer

A continuación se configuran la información, se da click “Datos”

Figura 23. Configuración almacén Datos GeoServer2

Figura 24. Configuración de Datos GeoServer

Se selecciona la opción “Entidades”

Figura 25. Configuración Entidades

Se Crea una nueva entidad, nótese que aparece la cartografía que se había cargado a la base de datos postgis dentro del almacén de datos que se creó llamado “popa” en un paso anterior.

para la capa pop_urbano se tiene

Figura 26. Crear nueva entidad en GeoServer

Figura 27. Crear nueva entidad en GeoServer2

Cuando se da click en nuevo aparece la página que me permite personalizar la información que se va a mostrar en el mapa.

Se asigna un “Alias” a la entidad, para este caso va a ser “pop_urbano” igual al archivo shape cargado en la base de datos.

Para el “Estilo” se selecciona line y damos click en la opción “Crear nuevo SLD”

Figura 28. Crear nuevo SLD para Entidad

Para esta capa, se escogen las características que deseemos y se da click en “Apply Style” y después en “Finished”.

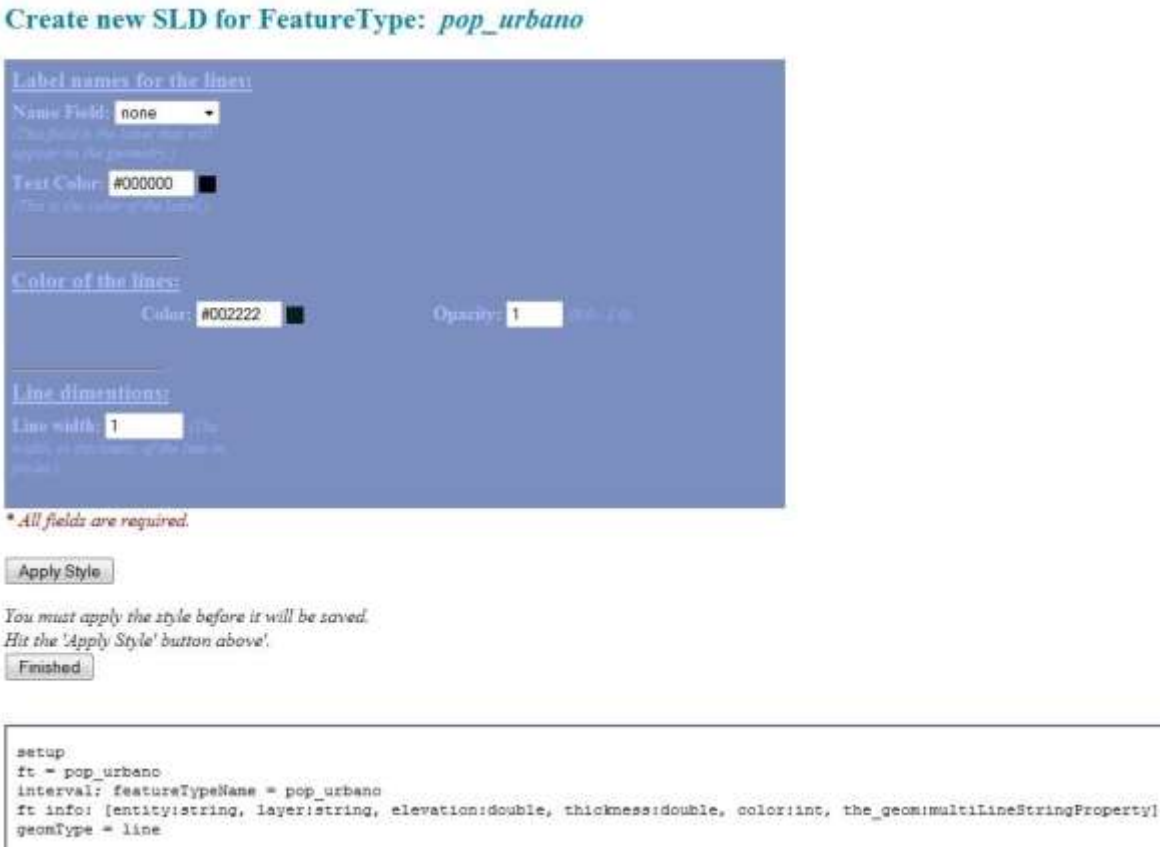

Figura 29. Configuración Estilo de Capa

En el campo SRS el valor es 21891 que es el Sistema de Referencia espacial para la zona que estamos trabajando.

Figura 30. Características de la Entidad

En el campo “Encuadre”, simplemente se genera automáticamente presionando el botón “Generar”, es para delimitar la zona donde se encuentra la cartografía cargada con el fin de realizar un despliegue adecuado.

Figura 31. Características de la Entidad 2

Ahora le damos click en enviar

Figura 32. Características de la Entidad 3

Finalmente se aplica y Guarda los cambios realizados

Figura 33. Guardar Cambios

Para verificar que la capa quedó bien simplemente en “Demostración” podremos ver la cartografía entrando a “Vista Preliminar de Mapas”.

Figura 34. Vista Preliminar de Mapas

En el listado aparece la entidad que se cargó. Se selecciona.

Figura 35. Entidades Habilitadas

Y tendremos una vista previa de la capa seleccionada.

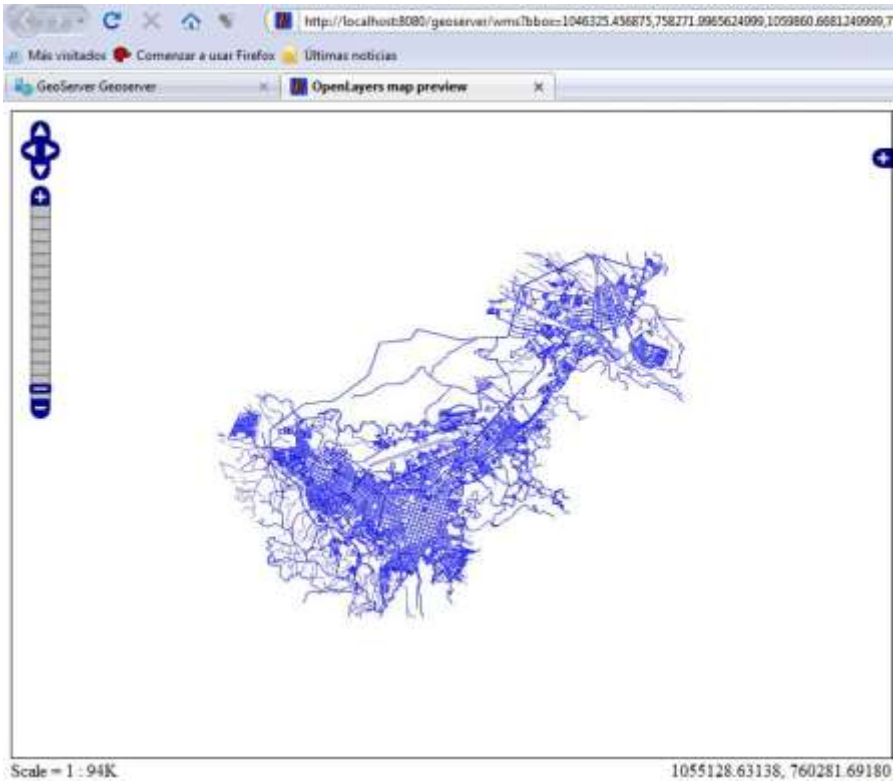

Figura 36. Vista Preliminar de la Entidad configurada

1.7. Despliegue de Cartografía con OpenLayers.

Para poder realizar esta práctica es indispensable haber realizado las practicas 2 y 3. Una vez se tiene disponible la cartografía en la base de datos PostgreSQL, se ha creado el almacén de datos referente a nuestra base de datos y por último referenciado las entidades con las tablas que se desean desplegar con OpenLayers, proseguimos a hacer lo siguiente:

Tomando como base.

El siguiente código presente en la documentación de Geoserver[8] y analizando cómo se desarrollaban otros ejemplos como [9], [10], se llegó a lo siguiente:

En la cabecera (head) se define la función init() que más adelante en el cuerpo (body) se llama.

Para mostrar una sola capa:

```
function init(){

	var map = new OpenLayers.Map('map');

	var wms = new OpenLayers.Layer.WMS("Urbano",
		"http://5.115.55.156:8080/geoserver/wms",
		{layers: 'topp:pop_urbano',
		}
		);

	map.addLayer(wms);
	var mapaBounds = new OpenLayers.Bounds(
		-76.65055896987614,	2.4184016930064898,	-76.5499556561985,
		2.4976740340573524
	);

	map.zoomToExtent(mapaBounds);

}
```

Cuando instanciamos una capa con openlayers tenemos los siguientes parámetros:

```
Var wms = new OpenLayers.Layer.WMS("nombre de la capa",
	"dirección ip del servidor",
	{layers: 'nombre de la capa en el servidor',
	}
	);
```

Para desplegar varias capas tenemos:

```
function init(){
	var map = new OpenLayers.Map('map');

	var wms1 = new OpenLayers.Layer.WMS("Urbano",
		"http://5.115.55.156:8080/geoserver/wms",
		{layers: 'topp:pop_urbano',
		}
		);

	var wms2 = new OpenLayers.Layer.WMS("Comunas",
		"http://5.115.55.156:8080/geoserver/wms",
		{layers: 'topp:popaco',
		}
		);

	var wms3 = new OpenLayers.Layer.WMS("Contadores",
		"http://5.115.55.156:8080/geoserver/wms",
		{layers: 'topp:SIGMApuntos',
		transparent: true
		}
		);

	map.addLayers([wms1,wms2,wms3]);
```

```
        map.addControl(new OpenLayers.Control.LayerSwitcher());
        var mapaBounds = new OpenLayers.Bounds(
                -76.65055896987614,    2.4184016930064898,    -76.5499556561985,
                2.4976740340573524
                );
        map.zoomToExtent(mapaBounds);
}
```

La diferencia es que al desplegar varias capas tenemos la función map.addLayers, cuando solo se despliega una se invoca la función map.addLayer.

2. PRACTICAS ENTIDAD GESTIONADA

Esta sección tiene que ver con el uso del protocolo SNMP, se trata en primera instancia de aprender a hacer uso de las variables del árbol de la MIB las cuales son muy útiles cuando una entidad externa que gestiona todo un conjunto de elementos y datos solicita información través de un agente en este caso el agente SNMP, información que es almacenada en determinadas variables identificadas con un único número (O.I.D).

Para probar esta función se utilizará un programa que aleatoriamente genera un dato numérico este se almacenará automáticamente en una variable, el árbol de la MIB está compuesto por varios grupos que a su vez se componen de variables, entre ellos el grupo System algunas de las variables que lo componen son sysDescr (Descripción completa del sistema (version, HW, OS)), sysObjectID (Identificación que da el distribuidor al objeto), sysUpTime (Tiempo desde la última reinicialización), sysName (Nombre del Equipo), entre otros, cada una con un único identificador (O.I.D) por ejemplo: 1.3.6.1.2.1.1.1 – sysDescr, 1.3.6.1.2.1.1.2 – sysObjectID, 1.3.6.1.2.1.1.3 – sysUpTime y 1.3.6.1.2.1.1.5 – sysName, la variable en la cual se almacenará este dato es la sysName del grupo System.

También se utilizará una variable de la sección extensible, la ideas es que al hacer una operación GET sobre esta se ejecute un programa que hace un cambio entre dos posibles estados por ej. Conectado/Desconectado de una carga eléctrica conectada al puerto paralelo, además consultar el estado en el cual se encuentra esta carga para lo cual se hace uso de otra variable de la sección extensible donde automáticamente se va almacenando ese dato.

Aprender a utilizar estas variables para manipular ciertos datos es muy útil ya que la finalidad de este protocolo es poder gestionar datos o dispositivos de manera remota, la lógica que maneja el protocolo SNMP es Gestor-Agente, esta práctica se trata de cómo configurar el agente para que quede listo a responder las solicitudes del gestor.

Primero que todo se debe contar con un computador que tenga la capacidad suficiente para que le sea instalado el sistema operativo Linux y algunos paquetes más como el Snmp y Snmpd de acuerdo a lo que se vaya a ejecutar en el equipo, es necesario aclarar

que esta práctica se diseño de acuerdo a la experiencia adquirida en el desarrollo de un sistema de gestión para lo cual se generaron dos programas uno el lenguaje Java encargado de generar un dato numérico aleatorio sumarlo a otro y almacenarlo en una variable de la MIB, por lo tanto se instalo el entorno de desarrollo Eclipse, el otro programa en lenguaje C contiene la lógica para que al consultar una varible se ejecute un cambio de estado en una salida del puerto paralelo al cual se conecta una carga, para esto se instalo un compilador de C y el paquete Shell.

2.1. Configuración agente SNMP

El primer paso luego de tener claro que se necesita es configurar el agente SNMP habiendo instalado previamente el sistema operativo Linux, en este caso se hizo con la versión Debian.

Es necesario contar con los siguientes componentes: Server (snmpd) y el Cliente (snmp), para lo cual se debe instalar el paquete Snmpd.

Procedimiento:

Ingresar como administrador al equipo bajo el sistema operativo Linux Debian e instalar el paquete snmp y snmpd, en Linux existen herramientas para efectuar este proceso está el gestor de paquetes Synaptic el cual permite hacerlo de manera gráfica, está el tradicional apt el cual permite hacerlo por consola tecleando "#apt – get install snmpd snmp" ó atraves del aptitude que es similar al apt solo que un poco más avanzado.

Por defecto el archivo de configuración queda localizado en el directorio "/etc/snmp" este contiene los siguientes archivos:

"snmpd.conf" y "snmptrapd.conf"

"/etc/snmp/snmpd.conf" – archivo de configuración del agente SNMP.
"/etc/snmp/snmptrapd.conf" – archivo de configuración del demonio trap SNMP.

El archivo en el cual se realizan los cambios y/o adiciones para configurar de manera que el agente cumpla con lo requerido es: snmpd.conf.
Antes de hacer cualquier cambio en "/etc/snmp/snmpd.conf" se recomienda hacer una copia del original utilizando el siguiente comando:

#cp /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.orig

Lo siguiente es crear las listas de control de acceso correspondientes en el archivo "/etc/snmp/snmpd.conf", estas listas permiten definir quién tendrá acceso hacia el servicio

snmpd. A una de estas listas se le otorgará permiso de acceso de lectura y escritura para lo que sea necesario y a la otra de solo lectura.

Por razones de seguridad solo la interfaz 127.0.0.1 será la de lectura escritura. Se otorgará permiso de acceso de solo lectura a una red o bien a una IP en la otra lista de control de acceso.

Se requiere que el agente pueda recibir órdenes desde ubicaciones remotas es decir desde otras maquinas ubicadas en la misma red, para realizar esto se agrega rangos de direcciones IP en las lista de control de acceso como se indica a continuación: abrimos el archivo “/etc/snmp/snmpd.conf”.

Este es el código que aparece por defecto al momento de la instalación, se modifica,

```
# sec.name            source  community
  com2sec     paranoid        default public
# com2sec     readonly        default public
# com2sec     readwrite default private
```

por:

```
# sec.name            source                  community
  com2sec     local   localhost       public
  com2sec     localNet        192.168.0.0/24 public
# com2sec     readwrite default               private
```

En lo anterior la segunda línea significa que habrá una lista de control de acceso denominada «local» y que corresponderá solo a 127.0.0.1/32 (localhost), asignando public como clave de acceso. En la tercera línea localNet es el nombre del grupo, 192.168.0.0/24 es el identificador de la red que va a ser monitoreada y public el nombre de la comunidad.

Se puede definir lo que se guste mientras no sea la clave de root, esto porque dicha clave se transmite a través de la red en forma de texto simple (es decir, sin cifrar).

La clave pública permite a los gestores realizar peticiones de valores de variables, mientras que la clave privada permite realizar peticiones de escritura. A estas palabras clave se les llama en SNMP communities. Cada dispositivo conectado con una red gestionada con SNMP, ha de tener configuradas estas dos communities.

Es muy común tener asignando por defecto el valor "public" al community público, y "private" al privado. Por lo que es muy importante cambiar estos valores para proteger la seguridad de la red. [1].

2.2. Definición de grupos.

Se crean dos grupos: MyRWGroup y MyROGroup. El primero será un grupo al que se asignarán más adelante permisos de lectura escritura y el segundo es un grupo al que posteriormente se asignará permisos de solo lectura. Por cada grupo se asignan tres líneas que especifican el tipo de acceso que se permitirá en un momento dado a un grupo en particular. Es decir, MyRWGroup se asocia a local y MyROGroup a localNet.

Procedimiento:

Se busca la siguiente linea: sec.model sec.name y debajo de esta se encuentran las líneas que van a ser modificadas como se indica a continuación:

```
#group          MyROSystem v1 paranoid
#group          MyROSystem v2c paranoid
#group          MyROSystem usm paranoid
group  MyROGroup v1 readonly
group  MyROGroup v2c readonly
group  MyROGroup usm readonly
group  MyRWGroup v1 readwrite
group  MyRWGroup v2c readwrite
group  MyRWGroup usm readwrite
```

y cambiarlas por:

```
group  MyROGroup v1 localNet
group  MyROGroup v2c localNet
group  MyROGroup usm localNet
group  MyRWGroup v1 local
group  MyRWGroup v2c local
group  MyRWGroup usm local
```

Tercer Paso: ramas.

Se especifican las ramas de la MIB que se van a permitir ver a través del servicio. Lo más común, es lo siguiente:

```
#          incl/excl subtree                    mask
view all    included  .1                      80
view system included  .iso.org.dod.internet.mgmt.mib-2.system
```

Cuarto Paso: Asignación de permisos a los grupos.

Se debe especificar que permisos tendrán los grupos MyROGroup y MyRWGroup, son de especial interés las últimas columnas.

```
context sec.model       sec.level match read     write  notif

#access MyROSystem ""      any     noauth  exact   system          none
access MyRWGroup ""        any     noauth          exact   all             all     all
access MyRWGroup ""        any     noauth          exact   all             all     all
```

2.3. Quitar la dirección de loopback para que el equipo pueda monitorear otras maquinas en la red.

Ya se tendría configurado el servidor SNMP, ahora nos aseguraremos de que el servidor este escuchando para cualquier máquina, el loopback es el sistema de trabajo en red en modo local por lo tanto se debe quitar la dirección loopback para que más equipos tengan acceso, para ello se sustituye en el archivo "/etc/default/snmpd" la línea,

SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid 127.0.0.1′

Por:

SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid'

Ahora sólo quedaría pendiente reiniciar el servicio y comprobar que todo ha funcionado correctamente. Reiniciaremos el servicio con el siguiente comando:

#/etc/init.d/snmpd restart

Considerando, como ejemplo, que se asignó como clave de acceso public en un sistema cuya dirección IP es 192.168.1.110, para probar si la configuración funciona, solo hay que ejecutar las dos siguientes líneas a fin verificar que devuelvan información acerca del sistema consultado.

#snmpwalk -v 1 192.168.1.110 -c public system
#snmpwalk -v 1 192.168.1.110 -c public interfaces

Para probar la configuración con el localhost se ejecuta desde consola la sentencia:

#snmpwalk localhost -c public -v1 [2] [3].

Con el procedimiento descrito anteriormente se inicio el Agente SNMP para que el gestor pueda monitorear el sistema, ahora se implementaran algunas funciones.

2.4. Creación de funciones

Una de las funciones que puede hacer el agente como se nombro al inicio es ejecutar ordenes que una entidad gestora emita, por ejemplo atender una solicitud a través de la operación GET, el agente responde enviando el dato solicitado que está almacenado en una variable de la MIB se puede desarrollar un programa en java que genere automáticamente números de manera aleatoria y lo sume a otro, este dato numérico a media que cambia se va almacenando en la variable sysName cuyo identificador (O.I.D) es 1.3.6.1.2.1.1.5.0.

Se solicita de manera local con la operación GET así,

#snmpget –v1 -c public localhost .1.3.6.1.2.1.1.5.0

y se obtiene la siguiente respuesta:

SNMPv2 –MIB:: SysName.0=STRING=350.0, esto quiere decir que el valor que se está registrando al momento de la consulta es 350, a medida que el valor que se está registrando cambia se altera también el valor almacenado en la variable, se hacen nuevas consultas para ver este cambio así:

#snmpget –v1 -c public localhost .1.3.6.1.2.1.1.5.0

y se obtiene la siguiente respuesta:

SNMPv2 –MIB:: SysName.0=STRING=381.0, esto quiere decir que el valor se incremento en 31.

La variable sysName es donde generalmente se almacena el nombre del equipo como se puede ver en la siguiente línea esta orden: "# snmpget -c public -v 1 localhost system.0" arroja lo siguiente: SNMPv2-MIB::sysName.0 = STRING: debian, la versión del protocolo, el nombre de la variable y por último el nombre del equipo en este caso "debian", sin embargo aquí se utiliza con otro fin, almacenar un registro numérico.

Para lograr esto de manera más sencilla la recomendación es usar una API como la que tiene Adventnet al tener este paquete se puede disponer rápidamente de una aplicación que ejecute una solicitud Snmp es decir Snmpget, esta API cuenta con las librerías necesarias para ser ejecutada en Java, es importante descargar el paquete RFC1213.mib y guardarlo en la carpeta donde estará guardado el proyecto.

Para más claridad los pasos nombrados se ilustran a continuación:

1. Luego de tener instalado el paquete Snmp, se configura el agente en el archivo se denomina snmpd.conf.

2. Se cambia el loopback para que no solo pueda ser accesado desde el localhost sino desde otras ubicaciones.
3. Se descarga una API generalmente basada en java que contenga aplicaciones como Snmpget que puedan utilizadas rápidamente, SNMPAPI de Adventnet es una opción con base en esto se puede generar una aplicación que genere un dato numérico.
4. Crear una carpeta con el nombre del proyecto y guardar allí la aplicación java Snmpget con adiciones y/o modificaciones como por ejemplo generar el dato numérico aleatorio que se almacenara en una variable. igualmente guardar aquí mismo la carpeta RFC1213.mib la cual es requerida por la aplicación.

A continuación se enseña un código el cual genera un número aleatorio lo suma a otro llamado anterior y por último da el resultado, este código muestra como guardar este dato y como localizarlo, se tomo como base el código Snmpget del SNMPAPI de Adventnet.

//1.Definir Librerias las trae por defecto el ejemplo Snmpget del SNMPAPI de adventnet

```
import com.adventnet.*;
import com.adventnet.snmp.beans.*;
import com.adventnet.snmp.snmp2.SnmpOID;
import com.adventnet.snmp.snmp2.SnmpVar;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

//2.Nombre de la Clase y definición de Variables Un thread es un hilo de ejecución en un programa

```
public class Consumo extends Thread {
private double consumo;
private double consumoSuma;
private int tiempo;
private SnmpTarget objSnmp;
private String ip;
```

//3.Definición del método Consumo que almacena un dato en la variable de la MIB identificada con .1.3.6.1.2.1.1.5.0, este dato se consulta de una ubicación determinada que se identifica con una dirección ip//

```
    public Consumo (String nombre, String ip_1){
        super (nombre);
        ip=ip_1;
        consumo = 0;
        tiempo = 0;
        objSnmp = new SnmpTarget();
        objSnmp.setTargetHost(ip);
        objSnmp.setObjectID(".1.3.6.1.2.1.1.5.0");
    }
```

//4. Definición del metodo getConsumoSuma que retorna un valor llamado consumoSuma

```
	public double getConsumoSuma() {
		return consumoSuma;
	}
//5. fijar el valor a través del método setValue y cargar la MIB RFC1213

		public void setValue(double in){
		try{
			objSnmp.setTargetHost(ip);
			objSnmp.setObjectID(".1.3.6.1.2.1.1.5.0");
			try{
			objSnmp.loadMibs("RFC1213-MIB");


				}
				catch (Exception ex){
				System.err.println("Error loading MIBs: "+ex);
					}

			objSnmp.snmpSet(""+in);
		}

				catch(DataException ex){		ex.printStackTrace();

		}
	}
//6. generación de un número aleatorio
	public double generarnumero(int numPosibilidades){

		double aleat = Math.random() * numPosibilidades;
		aleat = Math.floor(aleat);
		return aleat;
	}
//7. El número generado  se suma a un acumulado

	public float sumar(){
		objSnmp.setTargetHost(ip);
		String s=objSnmp.snmpGet(new SnmpOID(".1.3.6.1.2.1.1.5.0")).toString();
		float f=Float.parseFloat(s);

		return f;
	}
//8. Definición de parámetros bajo los cuales se ejecuta el método, consumo es el resultado de
sumar dos datos un “anterior”que se ha generado previamente denominado más un incremento //

	public void run(){
		tiempo = 0;
		while (tiempo <= 300 && consumo <= 500 ){
			double incremento = generarnumero(10);
			System.out.println("incremento "+incremento);
```

```
                        double anterior = sumar();
                        System.out.println("anterior "+anterior);
                        consumo = anterior + incremento;
                        setValue(consumo);
                        System.out.println("consumo "+consumo);

                        try{
                                tiempo = tiempo+1;
                                sleep(50);
                        }
                        catch(InterruptedException e){
                                        System.out.println("no se puede ejecutar");
                        }
                }
        }
}
```

El anterior es solo un ejemplo del uso de Snmpget del SNMPAPI de Adventnet, se muestra claramente el uso de la variable de la MIb para almacenar un dato.

El árbol de la MIB cuenta con una sección especial denominada “Sección Extensible”, ha sido creada con el propósito de que el sistema tenga la capacidad de ampliar las aplicaciones a través de adición de objetos, esta es una opción cuando los objetos que se quieren adicionar son pocos.
Mientras que para almacenar un registro numérico se probó con una variable del grupo system, la sysName: .1.3.6.1.2.1.1.5.0, para almacenar un estado de una carga se puede optar por una variable de la parte extensible de la MIB como la: entreprises.ucdavis.59.101.1 equivalente cuyo O.I.D es .1.3.6.1.4.1.2021.59.101.1, se puede usar otra variable para almacenar un código que es el ejecutable de un programa que hace el cambio de estado de un dato del puerto paralelo: entreprises.ucdavis.50.101.1 equivalente a .1.3.6.1.4.1.2021.50.101.1, al hacer una operación **get** sobre esta variable se ejecuta el programa que conmuta el estado de una carga, este programa se guarda en una carpeta puede ser en la ubicación /etc en el sistema de archivos de Linux.

La aplicación que contiene algún tipo de operación a realizarse sobre el puerto puede estar escrita en lenguaje C, si es así se generaran unos archivos ejecutables “.sh” estos archivos son muy importantes ya de ellos depende que la operación tenga éxito, la ubicación de estos se coloca junto a la línea que los ejecuta en el archivo de configuración del protocolo SNMP es decir el snmpd.conf, tal como se indica a continuación:

```
# Extensible sections.

extend .1.3.6.1.4.1.2021.50 shelltest /bin/bash /snmpc/cr1
extend .1.3.6.1.4.1.2021.59 shelltest /bin/bash /snmpc/consultar estado
```

#exec .1.3.6.1.4.1.2021.8 shelltest /bin/sh /testsnmp/shtest

% snmpwalk -v 1 -c public localhost .1.3.6.1.4.1.2021.50

entreprises.ucdavis.50.101.1 = "/bin/bash /snmpc/cr1"
#enterprises.ucdavis.50.1.1 = 1
#enterprises.ucdavis.50.2.1 = "shelltest"
#enterprises.ucdavis.50.3.1 = "/bin/sh /tmp/shtest"
#enterprises.ucdavis.50.100.1 = 35
#enterprises.ucdavis.50.101.1 = "hello world."

#enterprises.ucdavis.50.101.2 = "hi there."
#enterprises.ucdavis.50.102.1 = 0

La O.I.D .1.3.6.2.4.1.2021.50 es el identificador de objeto almacenado en la base de información de gestión (MIB), en este caso este objeto en particular representa una carga identificada como la carga cr1.

Como se dijo al principio se debe instalar un entorno de desarrollo integrado de código abierto multiplataforma en este caso Eclipse para correr el método Snmpget, para que este funcione y se actualice se debe descargar el modulo MIB del estándar RFC1213 y guardarlo en la carpeta donde está el programa en java generalmente en el workspace de Eclipse.
Para probar el funcionamiento del sistema se ejecutan de manera local las siguientes líneas:

Para consultar el valor que está almacenado en la variable sysName del grupo System cuya O.I.D es .1.3.6.1.2.1.1.5.0, se ejecuta:

snmpget -v 1 -c public localhost .1.3.6.1.2.1.1.5.0

Para hacer la conmutación de un valor en un dato del puerto que implica el cambio en el estado de una carga (conectado/desconectado) se utiliza la variable entreprises.ucdavis.50.101.1 de la sección extensible cuya O.I.D es .1.3.6.1.4.1.2021.50, se ejecuta:

snmpget -v 1 -c public localhost .1.3.6.1.4.1.2021.50

Para hacer la consulta del estado de esa carga (conectada/desconectada) se consulta la variable entreprises.ucdavis.59.101.1 de la sección extensible cuya O.I.D es .1.3.6.1.4.1.2021.59, se ejecuta:

snmpget -v 1 -c public localhost .1.3.6.1.4.1.2021.59

REFERENCIAS BIBLIOGRÁFICAS

[1] Manual PostGIS, Manuel Martín Martín, disponible en: http://postgis.refractions.net/documentation/postgis-spanish.pdf

[2] Install PostgreSQL 8.3.0, ArcSDE 9.3, and PostGIS 1.3.2 on Windows, ESRI Support Center, disponible en: http://support.esri.com/index.cfm?fa=knowledgebase.techArticles.articleShow&d=35128

[3] Paul Ramsey, PostGIS Manual, 2002, disponible en: http://www.geoconnections.org/developersCorner/devCorner_devNetwork/meetings/2002.05.30/postgis.pdf

[4] Paul Ramsey, Example queries – a compendium, 2007, disponible en: http://postgis.refractions.net/pipermail/postgis-users/2007-July/016358.html

[5] Mark Leslie, Geospatial Architect, LISAsoft, Introduction to PostGIS Spatial Extensions for PostgreSQL , 2009, disponible en: http://pugs.postgresql.org/files/Introduction_to_PostGIS_v1.0.pdf

[6] What Is PostGIS?. Boston Geographic Information Systems, Disponible en: http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgis_tut01

[7] GeoServer. User Tutorial Shapefile, disponible en: http://geoserver.org/display/GEOSDOC/User+Tutorial+Shapefile

[8] OpenLayers , GeoServer Documentation, disponible en http://geoserver.org/display/GEOSDOC/OpenLayers

[9] Getting Started, OpenLayers Library Documentation, disponible en: http://docs.openlayers.org/library/introduction.html

[10]Layers, OpenLayers Library Documentation, disponible en: http://docs.openlayers.org/library/layers.html