

Requerimientos de instalación de SIMOS

Versión 2.0

Enrique Velasco Jiménez.
Jeobany Ramirez Escobar.

Control de cambios.

Versión	Responsable	Fecha
1.0	Jeobany Ramirez Escobar	17/01/2023
2.0	Jeobany Ramirez Escobar	24/07/2023

Tabla de contenido

Introducción	4
Requerimientos para la instalación	4
Requerimientos de software y dependencias	4
Requerimientos de código	4
Diagrama de flujo de alto nivel de la interacción entre dependencias	5
Instalación de SIMOS	5
1. Instalar las herramientas para desarrolladores	5
2. Instalación de Firewall	6
3. Instalación del servidor Apache HTTP	6
4. Instalación de PHP	8
5. Instalación de Tomcat.	9
6. Instalación de PostgreSQL	13
7. Instalación de MapServer	15
8. Configuración de la base de datos	17
9. Instalación de SIMOS core	23
9.1. Despliegue de servicios.	24
9.2. Despliegue de Apache Solr	25
9.3. Despliegue del cliente web.	28
9.4. Configuración básica de SIMOS	30
9.4.1. Configuración de UI y Mapas.	30
9.4.2. Configuración del origen de datos.	30
9.4.3. Configuración de las capas.	31
9.4.4. Configuración de Solr	34
9.4.5. Configuración de identificación y análisis espacial	38
10. Funcionalidades adicionales.	42
10.1. Visualización de capas tipo cluster y mapa de calor	42
10.1.1. Capas Cluster.	42
10.1.2. Capas Mapa de Calor.	44
10.2. Leyenda personalizada	46
10.3. Estadísticas y graficas con pop-up	50
10.4 Carga de capas	55
Referencias	56

Introducción.

El Sistema de Movilidad Integrado (SIMOS) es una aplicación de software libre (Open Source) potenciada por el core de MxSIG, una plataforma de código abierto para la web desarrollada para implementar soluciones geomáticas que facilitan el uso, integración, interpretación, publicación y análisis de la información geográfica y estadística.

SIMOS es una aplicación que fomenta el acceso libre y gratuito a los datos geoespaciales de los que dispone con el objetivo de informar, generar estadística básica y ayudar a la toma de decisiones.

La capacidad de georreferenciación con la que cuenta SIMOS es de gran importancia para la política pública, tanto en su formulación como en la implementación, monitoreo y evaluación de esta, incrementando la efectividad de la gestión de los gobiernos estatales y municipales al evidenciar y enfocar las necesidades en la agenda pública.

Requerimientos para la instalación.

De acuerdo con una capacitación proporcionada por INEGI, no se tiene con exactitud un registro de requerimientos mínimos que sean necesarios para la instalación de MxSIG, y en consecuencia, de SIMOS.

No obstante, con base en el despliegue actual disponible en <http://simos.col.gob.mx/mxsig/> y pruebas realizadas en máquinas virtuales, los requerimientos del servidor son los siguientes:

- Servidor con sistema operativo CentOS 7 x86
- Clonar el repositorio de SIMOS [1][2]
- 4 GB de ram (recomendado)
- 2 vCPU/core (recomendado)
- Mínimo 8 GB de almacenamiento (16 GB recomendados)

Cabe destacar que, con los requerimientos antes enlistados, SIMOS funciona de manera adecuada, aunque se han presentado situaciones extraordinarias en las que se recomienda un procesador con más de 2vCPU y 16GB de RAM, esto en función del uso concurrente que se le dé a la aplicación.

Requerimientos de software y dependencias.

En esta sección se enlistan los complementos de software que son necesarios para el funcionamiento de SIMOS.

- PgAdmin
- PostgreSQL 11
- PostGIS 25.11
- MapServer 7
- Firewall (firewalld)
- Servidor Apache HTTP
- PHP 7
- Java 8
- Apache Tomcat 8.5.68

Requerimientos de código.

Finalmente, en esta sección se enlistan los componentes que se requieren en código, todos los componentes se pueden encontrar en [2].

- Tomcat-Solr
- Archivos WAR
- Cliente web MxSIG

Diagrama de flujo de alto nivel de la interacción entre las dependencias

En la fig. 1 se ilustra de manera sencilla la forma en que los componentes están distribuidos dentro del servidor con CentOS 7.

Nótese que los archivos war mencionados en la sección anterior son los siguientes:

- Genera KML: Servicio encargado de tomar las capas seleccionadas en SIMOS y generar archivos KML.
- Mdmdownloadfile: Servicio encargado de descargar los datos de las capas seleccionadas en SIMOS, así como la descarga de las capas en impresiones en formato PDF.
- Mdmexport: Servicio encargado de exportar las capas de SIMOS.
- Tomcat Solr: Servicio encargado de la indexación y búsqueda de capas en SIMOS.
- mdmServices: Es el servicio que más tareas realiza, por ejemplo, se encarga del análisis espacial y las consultas de datos de puntos.

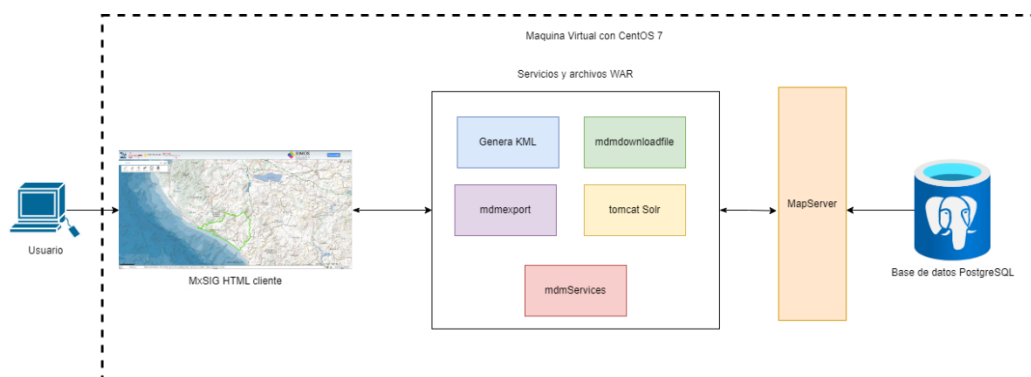


Figura 1: Distribución de componentes en el servidor.

Instalación de SIMOS:

Considerando que ya se tiene un servidor con sistema operativo CentOS 7 x86 y que se puede acceder a este a través de una conexión SSH y FTP, es posible instalar SIMOS siguiendo los siguientes pasos:

1. Instalar las herramientas para desarrolladores

A través de una conexión SSH con el servidor que tiene CentOS 7 instalado, se deben ejecutar los siguientes comandos para instalar las **herramientas de desarrollador**, esenciales para complementar algunas librerías de SIMOS, así como el comando **wget** para poder descargar archivos de la web y **nano**, un editor de texto que será de utilidad para configurar y editar archivos de SIMOS.

```
yum -y groupinstall "Development Tools"
yum -y install wget
yum -y install nano
```

Note que la bandera -y esta presente en los comandos para aceptar la instalación por default.

2. Instalación de Firewall

Comúnmente el sistema operativo CentOS ya tiene un firewall instalado y ejecutándose. Para verificar el estado del firewall se puede utilizar el siguiente comando:

```
firewall-cmd --state
```

En caso de que el firewall no este instalado o este deshabilitado, se puede instalar y habilitar con los siguientes comandos:

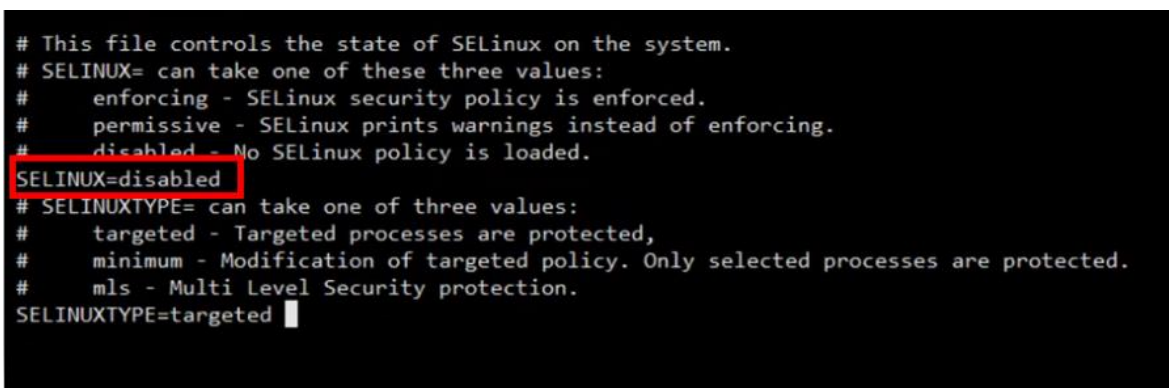
```
yum -y install firewalld  
systemctl enable firewalld
```

A continuación, se deben de editar los archivos de configuración de SELinux (Security-Enhance Linux) para deshabilitarlo. SELinux es código que define los controles de acceso a aplicaciones, procesos y archivos dentro del sistema. [3]

Para deshabilitar SELinux se debe modificar el archivo de configuración a través de nano con el siguiente comando:

```
nano /etc/selinux/config
```

Se debe buscar la línea de código que define la variable **SELINUX** y cambiar su valor a **disabled**, como se muestra en la figura 2.



```
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#   enforcing - SELinux security policy is enforced.  
#   permissive - SELinux prints warnings instead of enforcing.  
#   disabled - No SELinux policy is loaded.  
SELINUX=disabled  
# SELINUXTYPE= can take one of three values:  
#   targeted - Targeted processes are protected,  
#   minimum - Modification of targeted policy. Only selected processes are protected.  
#   mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

Figura 2: Archivo de configuración de Firewall.

Este cambio requiere que el servidor se reinicie, para ello se puede utilizar el comando **reboot**. Luego de reiniciar el servidor se recomienda confirmar que el firewall sigue en estado “running”.

3. Instalación del servidor Apache HTTP.

El servidor Apache HTTP es un proyecto de código abierto de un servidor web HTTP para sistemas operativos modernos basados en UNIX y Windows. Este servidor será el responsable de atender y responder las peticiones que los usuarios hagan a SIMOS, específicamente, Apache HTTP es el encargado de consultar la base de datos del proyecto. [4]

Para instalar Apache, primero se debe instalar algunos archivos de configuración y repositorios disponibles en el paquete EPEL (Extra Packages for Enterprise Linux). Para instalar la versión más reciente de los paquetes EPEL se debe ejecutar el siguiente comando: [5]

```
yum -y install epel-release
```

En seguida, para evitar conflictos derivados de la versión de IP utilizada para actualizar el comando YUM y los repositorios YUM del sistema operativo se debe modificar el archivo `/etc/yum.conf` para que se puedan resolver las peticiones utilizando una IPv4. Ejecutar el siguiente comando realiza esa modificación: [6]

```
echo "ip_resolve=4" >> etc/yum.conf
```

A continuación, se actualizan todos los paquetes instalados en el sistema operativo, incluyendo los paquetes de seguridad. Se puede utilizar el siguiente comando:

```
yum -y update
```

Después, se instala Apache HTTP con el siguiente comando

```
yum -y install httpd
```

También se debe instalar `httpd-devel` que contiene archivos binarios adicionales que son requeridos.

```
yum -y install httpd-devel
```

Para iniciar el servidor Apache HTTP se ejecuta el siguiente comando.

```
systemctl start httpd
```

Para habilitar el servidor y que se inicie en automático al iniciar el sistema operativo, se ejecuta el siguiente comando.

```
systemctl enable httpd
```

Para verificar que el servidor este ejecutándose apropiadamente se puede ejecutar el siguiente comando.

```
systemctl status httpd
```

Para evitar que el firewall impida el acceso al servidor desde la internet, se deben modificar las reglas del firewall con los siguientes comandos.

```
firewall-cmd --zone=public --permanent --add-service=http
```

```
firewall-cmd --zone=public --permanent --add-service=https
```

Luego, se reinicia el firewall con el comando.

```
firewall-cmd --reload
```

Finalmente, se puede verificar el acceso al servidor a través de un navegador de internet, colocando en la barra de navegación la IP pública del servidor y el puerto :80, como se muestra en la figura 3.

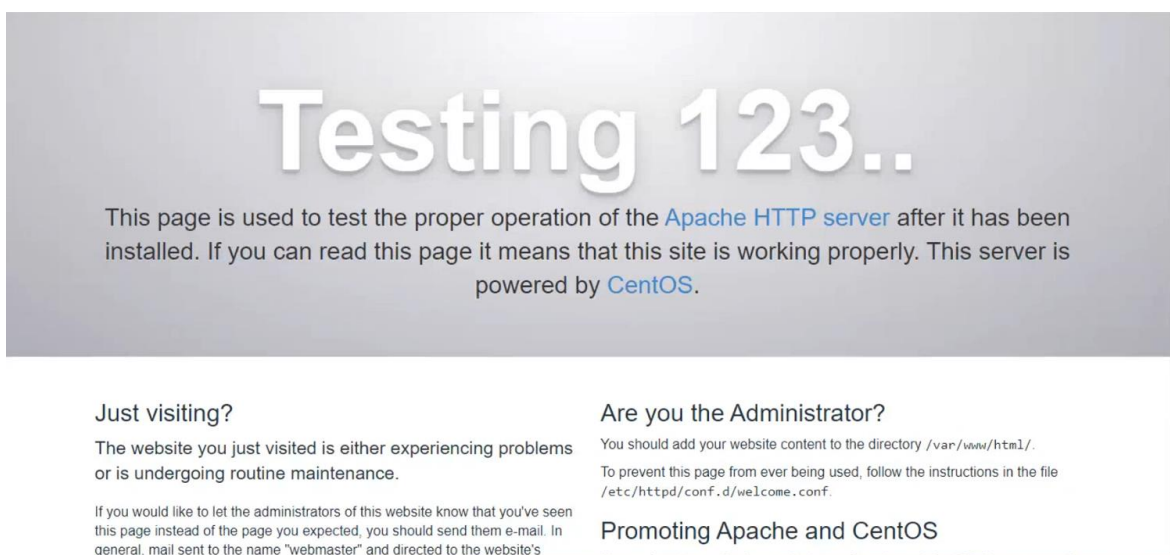


Figura 3: Página de inicio del servidor Apache.

4. Instalación de PHP

PHP es un lenguaje de programación de código abierto especialmente utilizado para desarrollo web y ser incrustado en páginas HTML. [7]

Instalar PHP no es estrictamente necesario para el funcionamiento de SIMOS, ni el de su core MxSIG. No obstante, es útil para desarrollar funcionalidad adicional al código de SIMOS.

Primero, se debe instalar el repositorio REMI, el cual es una colección de paquetes en donde se incluye PHP 7.4, así como diferentes librerías de PHP, por ejemplo, php-mysql para conexión con una base de datos de MySQL. El comando para instalar el repositorio REMI es el siguiente.

```
yum -y install http://rpms.remirepo.net/enterprise/remi-release-7.rpm
```

El repositorio REMI requiere una configuración para instalar PHP 7.4. Para lograr lo anterior, se deben instalar las utilidades de YUM y luego configurar el repositorio con los siguientes comandos.

```
yum -y install yum-utils
yum-config-manager --enable remi-php74
```

Después, se instala PHP y algunas librerías y complementos de utilidad con el siguiente comando.

```
yum -y install php php-gd php-pgsql php-mbstring php-xml php-mcrypt php-cli php-gd php-curl php-mysql php-
ldap php-zip php-fileinfo
```

Para comprobar que PHP fue instalado correctamente, se puede verificar la versión instalada con el siguiente comando.

```
php -v
```

Por último, se reinicia el servidor Apache HTTP.


```
systemctl restart httpd.service
```

5. Instalación de Tomcat.

Apache Tomcat es un contenedor de servlets de código abierto, comúnmente utilizado para compilar y ejecutar aplicaciones web desarrolladas con el lenguaje Java, en otras palabras, Tomcat es como un servidor para ejecutar aplicaciones web Java. Tomcat es necesario para ejecutar componentes esenciales del core de MxSIG. [8]

Primero, se debe instalar Java en el sistema operativo, la versión recomendada es la 1.8.0. Para instalar Java se recomienda utilizar el siguiente comando.

```
yum -y install java-1.8.0-openjdk.x86_64
```

Después, se deben descargar los archivos binarios para la instalación de Tomcat. Los archivos se pueden descargar en cualquier directorio dentro del sistema operativo, pero se recomienda crear un directorio específico para hacerlo, por ejemplo, creando el directorio “instalación” en la ruta /opt/tomcat, para esto se pueden utilizar los siguientes comandos.

```
cd /usr/local  
mkdir instalacion  
cd instalacion
```

En seguida, para descargar los archivos de Tomcat es necesario verificar la versión menor más actual de Tomcat 8 accediendo a la siguiente página.

<https://downloads.apache.org/tomcat/tomcat-8/>

Como se muestra en la figura 4, la versión menor más actual de Tomcat 8 es la v8.5.92, no obstante, esta puede cambiar.

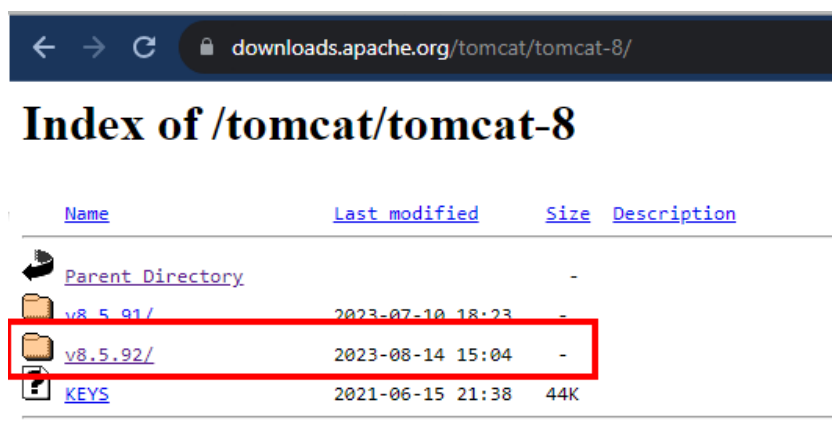
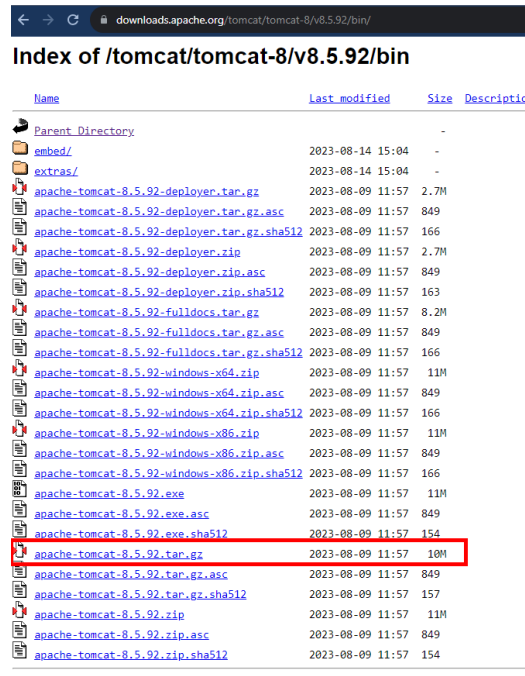


Figura 4: Versión recomendada de Tomcat 8.

Accediendo a esa versión, y después a la carpeta bin, se pueden encontrar los archivos de Tomcat comprimidos con extensión .tar.gz. Como se muestra en la figura 5.



Name	Last modified	Size	Description
Parent Directory		-	
embed/	2023-08-14 15:04	-	
extras/	2023-08-14 15:04	-	
apache-tomcat-8.5.92-deployer.tar.gz	2023-08-09 11:57	2.7M	
apache-tomcat-8.5.92-deployer.tar.gz.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92-deployer.tar.gz.sha512	2023-08-09 11:57	166	
apache-tomcat-8.5.92-deployer.zip	2023-08-09 11:57	2.7M	
apache-tomcat-8.5.92-deployer.zip.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92-deployer.zip.sha512	2023-08-09 11:57	163	
apache-tomcat-8.5.92-fulldocs.tar.gz	2023-08-09 11:57	8.2M	
apache-tomcat-8.5.92-fulldocs.tar.gz.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92-fulldocs.tar.gz.sha512	2023-08-09 11:57	166	
apache-tomcat-8.5.92-windows-x64.zip	2023-08-09 11:57	11M	
apache-tomcat-8.5.92-windows-x64.zip.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92-windows-x64.zip.sha512	2023-08-09 11:57	166	
apache-tomcat-8.5.92-windows-x86.zip	2023-08-09 11:57	11M	
apache-tomcat-8.5.92-windows-x86.zip.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92-windows-x86.zip.sha512	2023-08-09 11:57	166	
apache-tomcat-8.5.92.exe	2023-08-09 11:57	11M	
apache-tomcat-8.5.92.exe.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92.exe.sha512	2023-08-09 11:57	154	
apache-tomcat-8.5.92.tar.gz	2023-08-09 11:57	10M	
apache-tomcat-8.5.92.tar.gz.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92.tar.gz.sha512	2023-08-09 11:57	157	
apache-tomcat-8.5.92.zip	2023-08-09 11:57	11M	
apache-tomcat-8.5.92.zip.asc	2023-08-09 11:57	849	
apache-tomcat-8.5.92.zip.sha512	2023-08-09 11:57	154	

Figura 5: Archivos de descarga de Tomcat comprimido.

Nótese que la URL donde se encuentran los archivos de Tomcat de este ejemplo es la siguiente.

<https://downloads.apache.org/tomcat/tomcat-8/v8.5.92/bin/apache-tomcat-8.5.92.tar.gz>

Utilizando la URL de los archivos, se ejecuta el siguiente comando dentro de la maquina con CentOS.

```
wget https://downloads.apache.org/tomcat/tomcat-8/v8.5.92/bin/apache-tomcat-8.5.92.tar.gz
```

A continuación, para descomprimir los archivos se crea un nuevo sub-directorio dentro del directorio /opt con el siguiente comando.

```
mkdir /opt/tomcat
```

En seguida se descomprimen los archivos con el siguiente comando.

```
tar -zxvf apache-tomcat-8.5.68.tar.gz -C /opt/tomcat --strip-components=1
```

Nótese que el comando se ejecuta estando ubicado en la carpeta /usr/local/instalacion, que es donde se almacena el archivo comprimido.

El directorio donde se despliega Tomcat es /opt/tomcat. Se debe agregar el grupo y usuario tomcat y proporcionarles privilegios para leer, escribir y ejecutar sobre los directorios /opt/tomcat/conf/, /opt/tomcat/bin/ y el propio directorio /opt/tomcat/. Lo anterior se consigue con los siguientes comandos.

```
cd /opt/tomcat
groupadd tomcat
useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
chmod g+rw conf
chmod g+r conf/*
```

```
chmod g+rx bin
chmod g+r bin/*
chown -R tomcat:tomcat /opt/tomcat/
```

Posteriormente, se debe configurar el servicio de Tomcat creando el archivo tomcat.service en el directorio `/etc/systemd/system/` con el siguiente comando.

```
nano /etc/systemd/system/tomcat.service
```

El comando anterior crea el archivo `tomcat.service` en el directorio indicado y abre un editor de texto. Dentro de la edición del archivo se debe colocar la siguiente configuración del servidor.

```
[Unit]
Description=Apache Tomcat Web Application Container
After=syslog.target network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/jre
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -
XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom -Djava.net.preferIPv4Stack=true -
Djava.net.preferIPv4Addresses=true'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/bin/kill -15 $MAINPID

User=tomcat
Group=tomcat

[Install]
WantedBy=multi-user.target
```

Además, para que el archivo de configuración de Tomcat sea tomado en cuenta, se deben reiniciar los servicios con el siguiente comando.

```
systemctl daemon-reload
```

Tomcat permite ser administrado a través de una aplicación de nombre "Manager". Se debe editar el archivo `tomcat-users.xml` en el directorio `/opt/tomcat/conf/`. Lo anterior se puede realizar con el siguiente comando.

```
nano /opt/tomcat/conf/tomcat-users.xml
```

El comando anterior, se abre un editor de texto donde se debe colocar la configuración del usuario que será utilizado para acceder a la aplicación Manager.

En el fragmento de código mostrado en seguida, se muestra que la etiqueta <user> debe ir dentro de <tomcat-users>. Nótese que valor de los parámetros username y password pueden ser modificados a voluntad.

```
<tomcat-users>

                                <user username="admin" password="admin" roles="admin, manager,
manager-gui,admin-gui"/>
                                </tomcat-users>
```

Adicionalmente, debido a las políticas de seguridad de Tomcat, este puede ser accedido únicamente a través de la propia maquina donde esta desplegado. Para modificar esto y permitir que sea accedido desde cualquier maquina se debe editar el archivo context.xml utilizando el siguiente comando.

/opt/tomcat/webapps/manager/META-INF/context.xml

Dentro del archivo context.xml se deben comentar las etiquetas <Valve> y <Manager> como se muestra en el siguiente fragmento de código.

```
<Context antiResourceLocking="false" privileged="true" >

    <!--
        <Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+::1|0:0:0:0:0:0:1" />
        <Manager
sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|string)|org\.apache\.ca
talina\.filters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
    -->

</Context>
```

Nótese que la única modificación al archivo context.xml es que se comentaron las líneas de código utilizando las etiquetas "<!-- -->".

Usualmente, Tomcat se ejecuta en el puerto 8080 de la maquina donde se instala/despliega, es por ello que el firewall debe permitir el acceso a este puerto. Para conseguir lo anterior se puede utilizar el siguiente comando.

firewall-cmd --zone=public --permanent --add-port=8080/tcp

En seguida se debe reiniciar el firewall con el siguiente comando.

firewall-cmd --reload

Finalmente, para configurar que Tomcat se ejecute y que se ejecute en cuanto se inicie el sistema operativo de la maquina se utilizan los siguientes comandos.

systemctl enable tomcat
systemctl start tomcat

Para corroborar que el despliegue fue exitoso, se puede acceder a Tomcat utilizando el puerto 8080 desde un navegador de internet. Como se muestra en la figura 6.

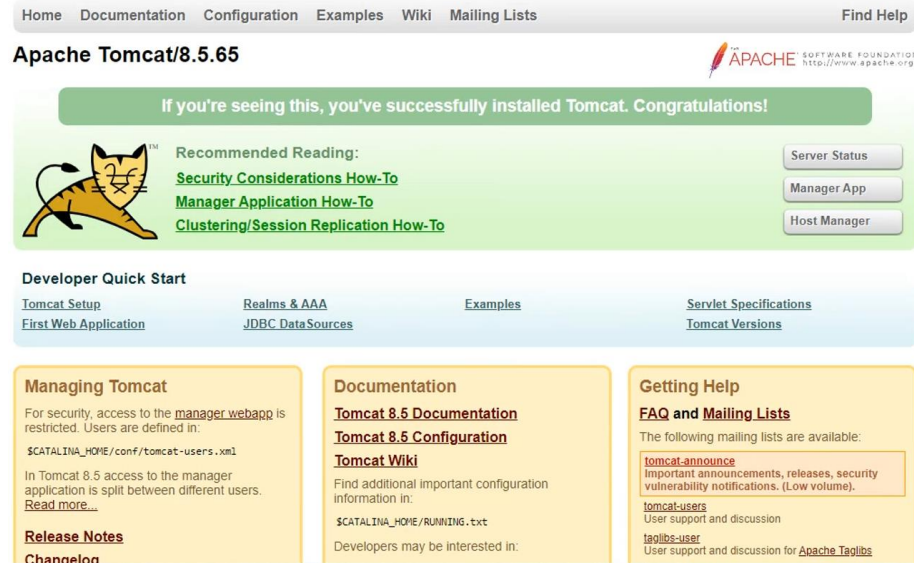


Figura 6: Página de inicio del servidor Tomcat.

6. Instalación de PostgreSQL

PostgreSQL es un sistema de bases de datos relacionales de código abierto que es confiable, robusto y tiene un buen rendimiento. PostgreSQL 11 es el sistema gestor de bases de datos recomendado para almacenar los datos de SIMOS, las capas geográficas, proyecciones y estadísticas. [9]

Para instar PostgreSQL en la maquina con CentOS 7 se tiene que instalar un repositorio externo, similar a lo ocurrido con Tomcat, en este caso el repositorio Red Hat Enterprise Linux (RHEL) contiene todo lo necesario para la instalación de PostgreSQL y algunas librerías adicionales. [10]

Para instalar RHEL se utiliza el siguiente comando. Nótese que en enlace de donde se obtiene el repositorio puede cambiar, se recomienda verificar su disponibilidad.

```
rpm -Uvh https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-7-x86_64/pgdg-redhat-repo-42.0-32.noarch.rpm
```

En seguida, se instala PostgreSQL 11 con el comando.

```
yum -y install postgresql11-server
```

Después, para iniciar la base de datos PostgreSQL, el servicio de PostgreSQL y que este se ejecute al arrancar el sistema operativo, se utilizan los siguientes comandos.

```
/usr/pgsql-11/bin/postgresql-11-setup initdb  
systemctl enable postgresql-11.service  
systemctl start postgresql-11.service
```

Para corroborar que Postgres se este ejecutando adecuadamente, se utiliza el siguiente comando.

```
systemctl status postgresql-11.service
```

A continuación, para configurar el acceso a Postgres, se debe agregar un usuario y una contraseña. Primero se ejecuta el siguiente comando, el cual abrirá una línea de comandos de Postgres en modo super usuario.

```
su - postgres -c "psql"
```

Después, se ejecuta el comando `/password postgres` para añadir la contraseña al usuario postgres. Finalmente, el comando `/q` cierra la línea de comandos de Postgres. Lo anterior se muestra en los siguientes comandos.

```
\password postgres
\q
```

Además, se debe añadir al firewall el puerto 5432, donde se ejecuta Postgres con los comandos siguientes.

```
firewall-cmd --permanent --add-port=5432/tcp
firewall-cmd --reload
```

Adicionalmente, se debe instalar PostGIS, un complemento que extiende la funcionalidad de PostgreSQL permitiéndole almacenar, indexar y consultar datos geográficos. [11]

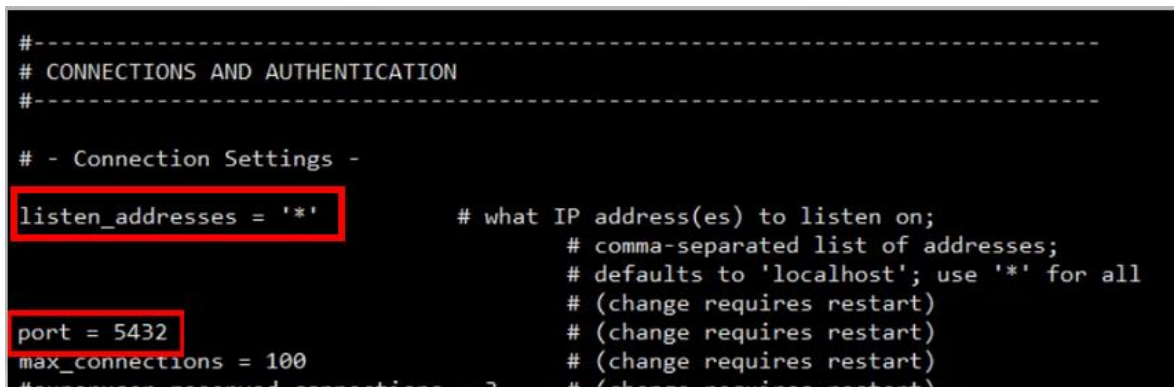
PostGIS para Postgres 11 se instala ejecutando los siguientes comandos.

```
yum -y install --skip-broken postgresql11-devel
yum -y install postgis25_11
rpm -qi postgis25_11
```

Para que un usuario pueda conectarse y autenticarse en Postgres, se deben configurar los puertos de conexión del archivo `postgresql.conf` en el directorio `/var/lib/pgsql/11/data/`. Lo anterior se consigue con el siguiente comando.

```
nano /var/lib/pgsql/11/data/postgresql.conf
```

Dentro de la edición del archivo `postgresql.conf` se deben des-comentar las líneas de **listen_addresses** y **port**. Además, el valor de `listen_addresses` debe modificarse a `*`. Por otro lado, se recomienda mantener el valor de `port` en **5432**, como se ilustra en la figura 7.



```
#-----
# CONNECTIONS AND AUTHENTICATION
#-----

# - Connection Settings -
listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 5432                    # (change requires restart)
max_connections = 100          # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
```

Figura 7: Archivo de configuración de PostgreSQL.

Posteriormente, se configuran las IP que pueden conectarse a Postgres en el archivo `pg_hba.conf` del directorio `/var/lib/pgsql/11/data/` con el siguiente comando.

```
nano /var/lib/pgsql/11/data/pg_hba.conf
```

Con fines prácticos, en el ejemplo de la figura 8 se muestra que se permiten todas las conexiones por IPv4, representadas por un valor de 0.0.0.0/0. Note que el valor del método es modificado a MD5.

#	TYPE	DATABASE	USER	ADDRESS	METHOD
# "local" is for Unix domain socket connections only					
local	all		all		peer
# IPv4 local connections:					
host	all		all	0.0.0.0/0	md5
# IPv6 local connections:					
host	all		all	:::1/128	ident
# Allow replication connections from localhost, by a user with the					
# replication privilege.					
local	replication		all		peer
host	replication		all	127.0.0.1/32	ident
host	replication		all	:::1/128	ident

Figura 8: Configuración de IP permitidas para conectarse a PostgreSQL.

Finalmente, se reinicia el servicio de Postgres para que los cambios surtan efecto.

```
systemctl restart postgresql-11
```

7. Instalación de MapServer

MapServer es una plataforma de código abierto para desarrollar aplicaciones web con mapas interactivos y publicación de datos espaciales. MapServer se encarga de mostrar el mapa y los datos de SIMOS. [12]

Antes de iniciar con la instalación de Map Server se requiere ejecutar los siguientes comandos, los cuales instalan librerías utilizadas por MapServer. Nótese que puede haber librerías que ya estén instaladas por defecto en el sistema operativo.

```
yum install -y gdal
yum install -y gdal-devel
yum install -y libxml2-devel
yum install -y geos-devel
yum install -y proj-epsg
yum install -y libjpeg-turbo-devel
yum install -y dejavu-sans-fonts
yum install -y fribidi
yum install -y fcgi
yum install -y librsvg2
yum install -y fcgi-devel spawn-fcgi
```

Por otro lado, se debe agregar la referencia espacial EPSG:900913 en el archivo epsg del directorio /usr/share/proj/. EPSG:900913 o *Web Mercator projection* es la proyección espacial utilizada por el core MxSIG y SIMOS. Para agregar la referencia espacial se ejecuta el siguiente comando.

```
nano /usr/share/proj/epsg
```


Después, se agrega la siguiente línea de código.

```
<900913> +proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0
+units=m +nadgrids=@null +no_defs <>
```

A continuación, para comenzar con la instalación de MapServer, se debe copiar el archivo **mapserver-7.0.0-1.el7.centos.x86_64.rpm**, ubicado en la carpeta MapServer del repositorio de SIMOS, al directorio /usr/local/instalacion, como se muestra en la figura 9. [1]

```
[root@mxsig instalacion]# pwd
/usr/local/instalacion
[root@mxsig instalacion]# ls -la
total 29456
drwxr-xr-x  3 root root   130 jul  4 2021 .
drwxr-xr-x 14 root root   166 ago 23 12:26 ..
-rw-r--r--  1 root root 10559131 jun 11 2021 apache-tomcat-8.5.68.tar.gz
-rw-r--r--  1 root root 1201236 abr  5 2016 mapserver-7.0.0-1.el7.centos.x86_64.rpm
```

Figura 9: Archivo de MapServer en la carpeta de instalación.

Se recomiendan dos maneras de copiar el archivo **mapserver-7.0.0-1.el7.centos.x86_64.rpm** al servidor, la primera es clonando el repositorio de GitHub y moviendo el archivo con comandos. La segunda forma, y la más recomendada, es a través de una ventana de SFTP.

Con la terminal ubicada en el directorio /usr/local/instalacion, se ejecutan los siguientes comandos para instalar MapServer.

```
rpm -ivh mapserver-7.0.0-1.el7.centos.x86_64.rpm
cp /usr/libexec/mapserv /var/www/cgi-bin
```

Finalmente, para corroborar la instalación de MapServer se puede ejecutar el siguiente comando para mostrar la versión instalada.

```
/var/www/cgi-bin/mapserv -v
```

También se puede consultar MapServer a través de un navegador de internet en el puerto a través del puerto 80 y el subdominio /cgi-bin/mapserv, como se muestra en la figura 10. El mensaje de la figura 10 no es un error, representa que MapServer responde, pero no se le consulto nada.

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

Figura 10: Consulta a MapServer vacía.

8. Configuración de la base de datos

Para comenzar con la configuración de la base de datos, se recomienda utilizar un gestor de base de datos para establecer conexión con la versión de PostgreSQL instalada en CentOS. El gestor recomendado es pgAdmin 4, disponible en [13].

La interfaz gráfica de pgAdmin es similar a la que se muestra en la figura 11.

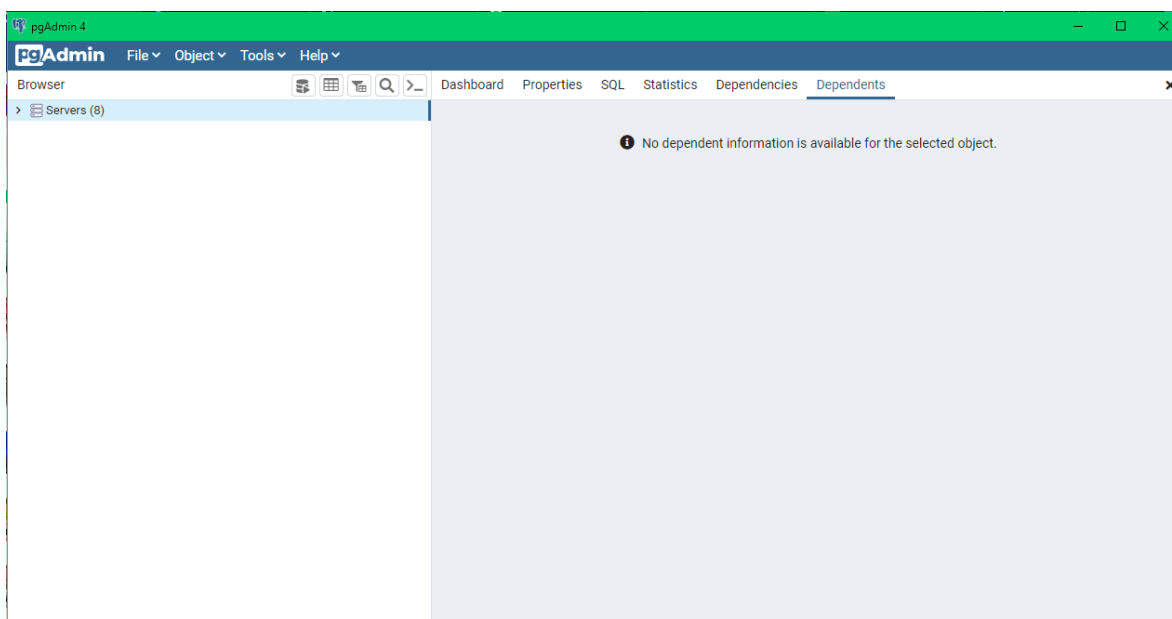


Figura 11: Interfaz gráfica de pgAdmin.

Para conectarse con el servidor, se debe dar click derecho en la opción de Servers y luego dar click en Create -> Server..., como se muestra en la figura 12.

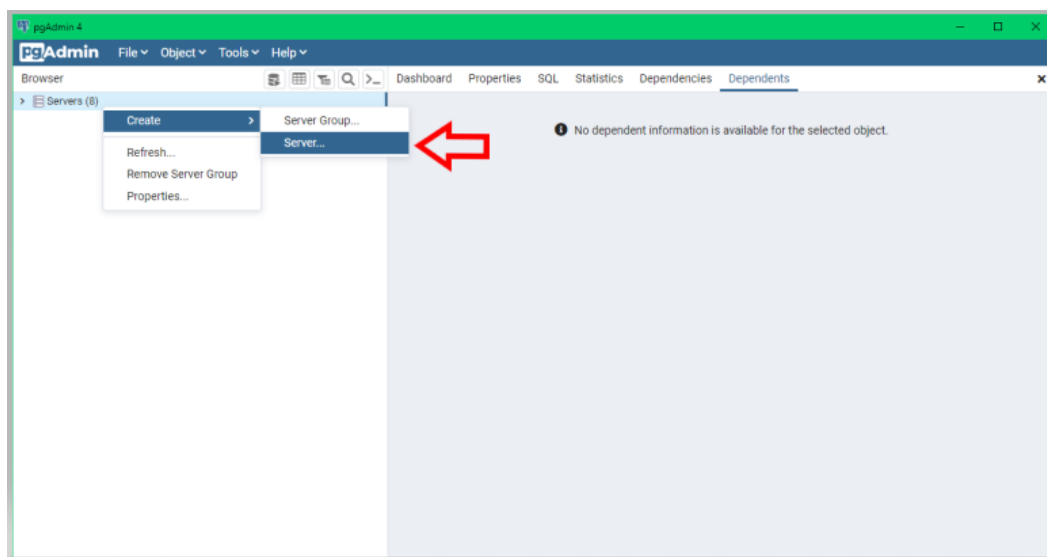


Figura 12: Selección de creación de server.

Entonces, aparecerá una ventana emergente. En la pestaña de “General” se coloca un nombre del servidor, como en la figura 13. Después, en la pestaña de “Connection” se coloca la dirección IP del servidor, el puerto de la base de datos (5432) y las credenciales del usuario postgres que se configuraron previamente, una vez realizado se da click en “Save”, como se ilustra en la figura 14.

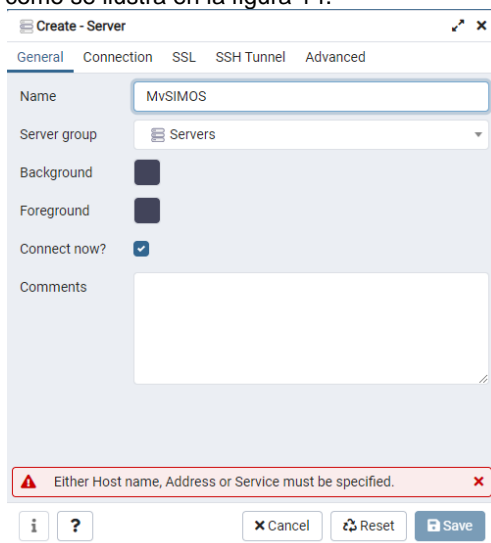


Figura 13: Opciones de creación de un servidor.

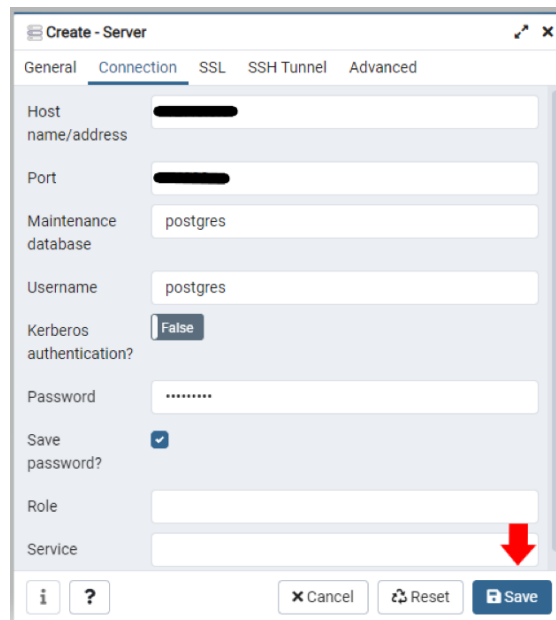


Figura 14: Opciones de conexión al servidor de PostgreSQL.

Una vez establecida la conexión, se selecciona el servidor en el navegador de pgAdmin. Dando click derecho sobre el servidor, se selecciona la opción Create -> Database..., como se muestra en la figura 15.

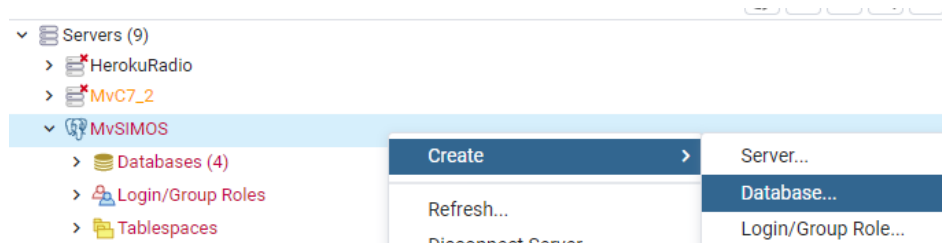
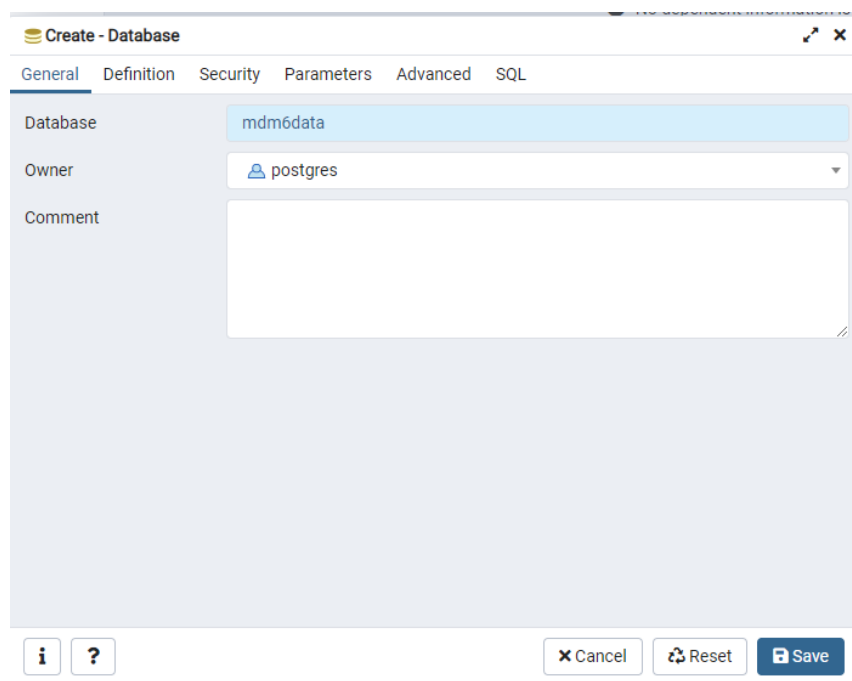


Figura 15: Servidor añadido y creación de una base de datos.

Se abre una ventana emergente y en el campo de texto de "Database" se coloca el nombre **mdm6data**. Este es el nombre de base de datos con el que SIMOS está configurado, como en la figura 16.



Create - Database

General Definition Security Parameters Advanced SQL

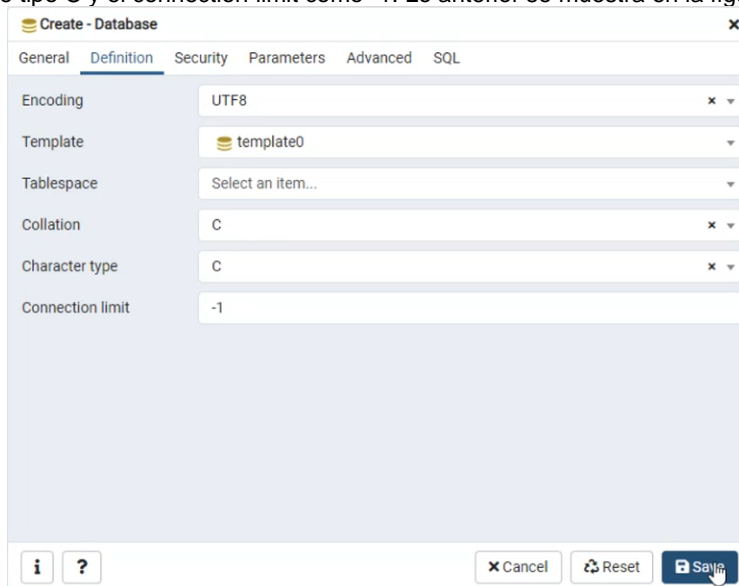
Database:

Owner:

Comment:

Figura 16: Nombre de la base de datos de SIMOS, mdm6data.

Después, en la pestaña de “Definition”, se selecciona la codificación UTF8, el template template0, Collation y Character type como tipo C y el connection limit como -1. Lo anterior se muestra en la figura 17.



Create - Database

General Definition Security Parameters Advanced SQL

Encoding:

Template:

Tablespace:

Collation:

Character type:

Connection limit:

Figura 17: Configuración de la base de datos mdm6data.

La nueva base de datos se mostrará debajo del servidor de Postgres en el navegador. Dando click derecho sobre la base de datos mdm6data, se selecciona la opción "Restore...", como se muestra en la figura 18.

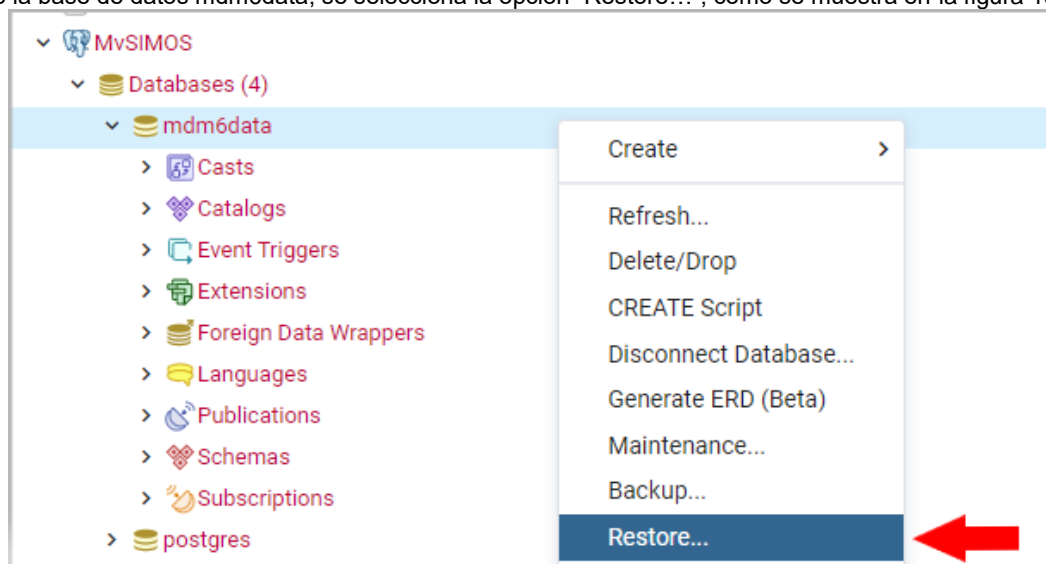


Figura 18: Selección de restauración de base de datos.

Se selecciona la dirección donde se encuentra el archivo **simos_tar** que se encuentra en el repositorio en la carpeta **DB**. Luego se da click en restore, como se muestra en la figura 19. Nótese que en el repositorio el archivo tar esta comprimido dentro de un archivo rar, primero debe extraerse de ahí.[1]

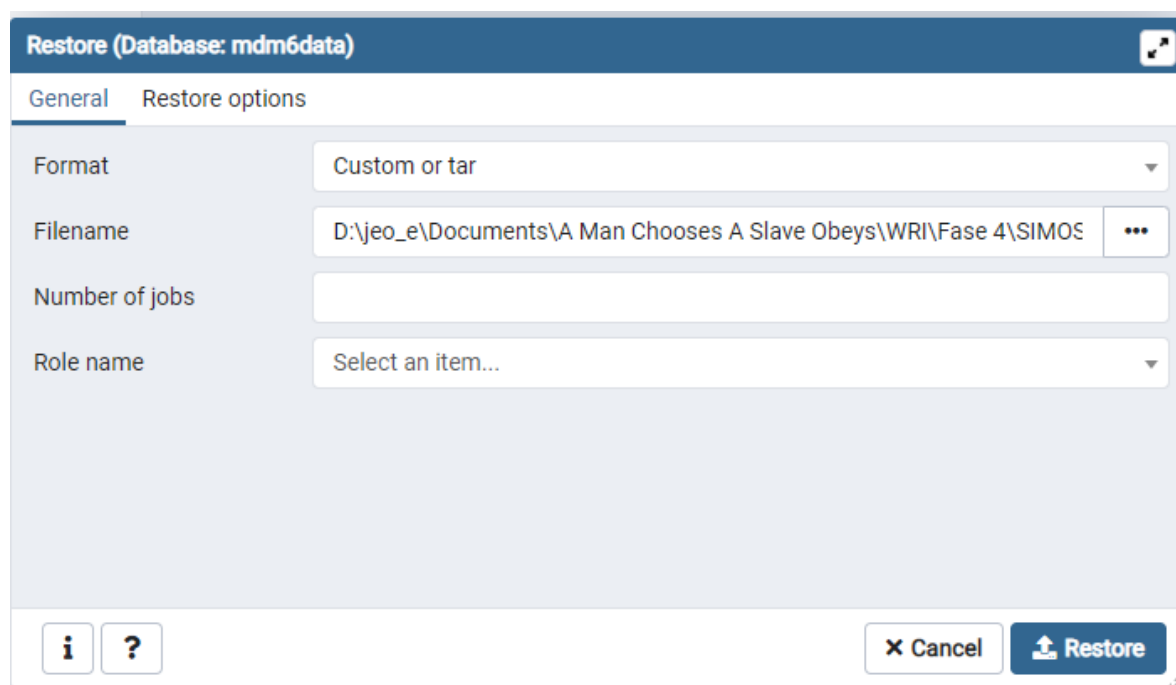


Figura 19: Restauración de la base de datos.

Una vez concluido el proceso de restauración de la base de datos mdm6data, de lado izquierdo de la interfaz gráfica de pgAdmin aparecerán una serie de esquemas debajo de la base de datos, como se muestra en la figura 20.



Figura 20: Esquemas de la base de datos.

Desplegando el esquema con nombre “prueba”, se encuentran las tablas de datos disponibles, estas tablas representan también capas que pueden ser mostradas por SIMOS. Se recomienda que cualquier nueva capa que se quiera integrar a SIMOS se almacene bajo este esquema, como en la figura 21.

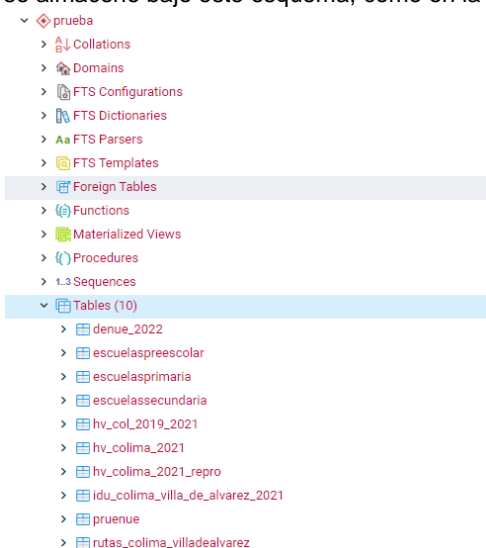


Figura 21: Tablas restauradas.

9. Instalación de SIMOS core.

Accediendo desde un navegador web al servidor Tomcat instalado en el paso 5. Instalación de Tomcat, se da click en el botón de Manager App, para abrir el Manager. Lo anterior se ilustra en la figura 22.

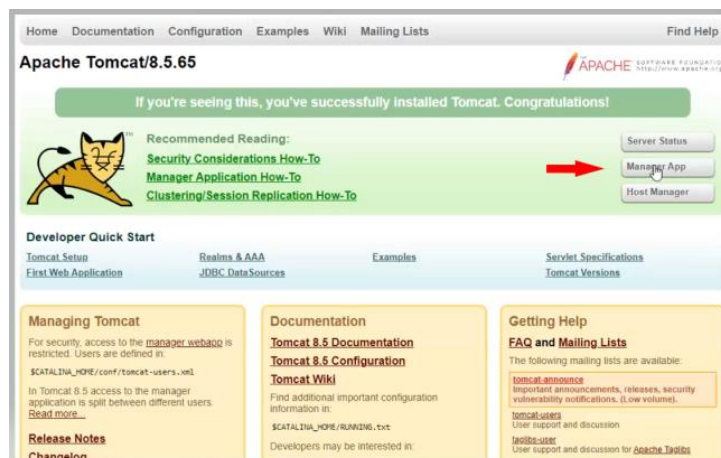


Figura 22: Abrir manager de Tomcat.

En seguida, aparece una ventana emergente solicitando las credenciales que se configuraron en el archivo `/tomcat-users.xml`. Después de colocar el usuario y contraseña se da click en el botón de “Iniciar sesión”, como se muestra en la figura 23.

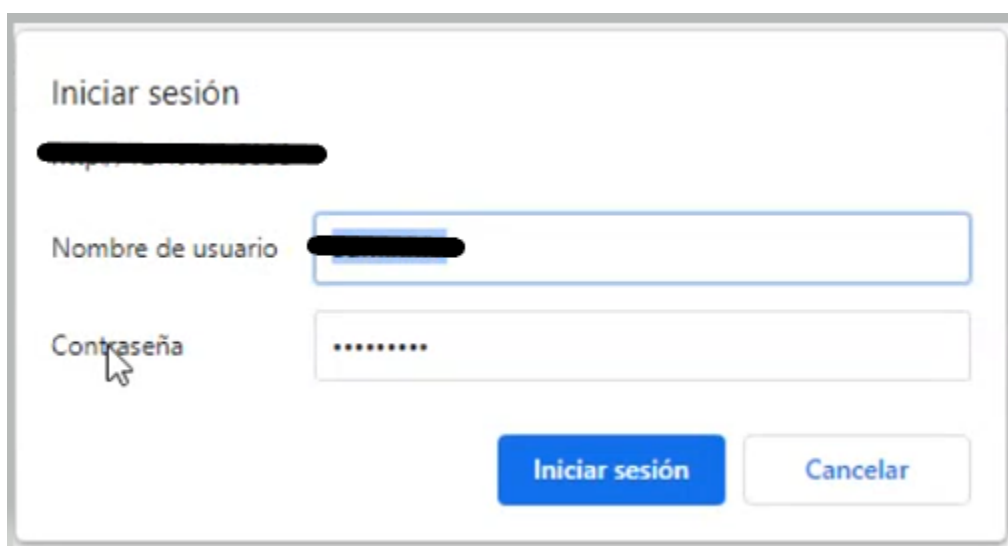


Figura 23: Inicio de sesión en el manager.

Entonces, se abre el Manager de Tomcat, en el cual se muestran todas las aplicaciones desplegadas en el servidor, como se ilustra en la figura 24.

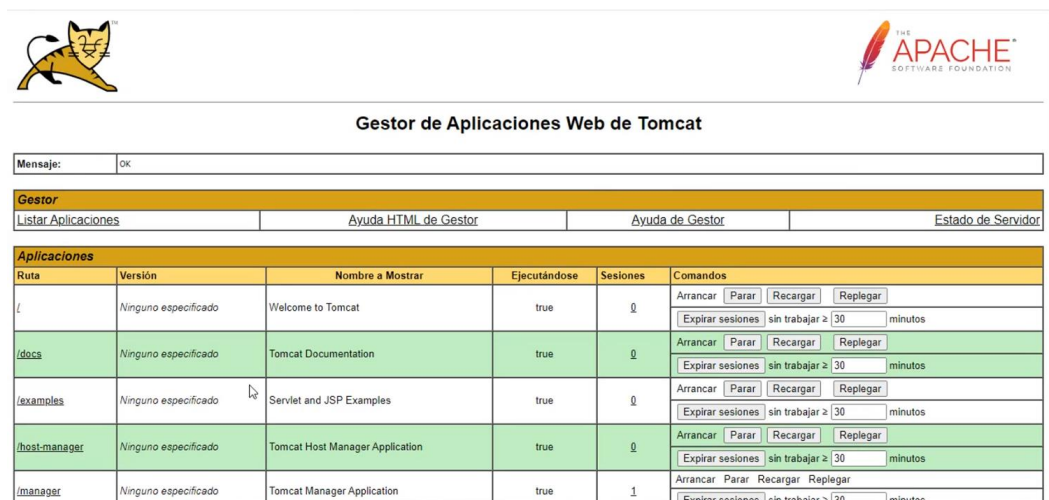


Figura 24: Manager de Tomcat

9.1. Despliegue de servicios.

El despliegue de aplicaciones se puede hacer a través del Manager de Tomcat, no obstante, se recomienda establecer una conexión FTP con la maquina con CentOS para realizar los despliegues.

Luego de establecer una conexión FTP con CentOS, se navega al directorio:

`/opt/tomcat/webapps`

Ese es el directorio donde se despliegan las aplicaciones de Tomcat, en la figura 25 se muestra el directorio de la instalación por default de Tomcat.

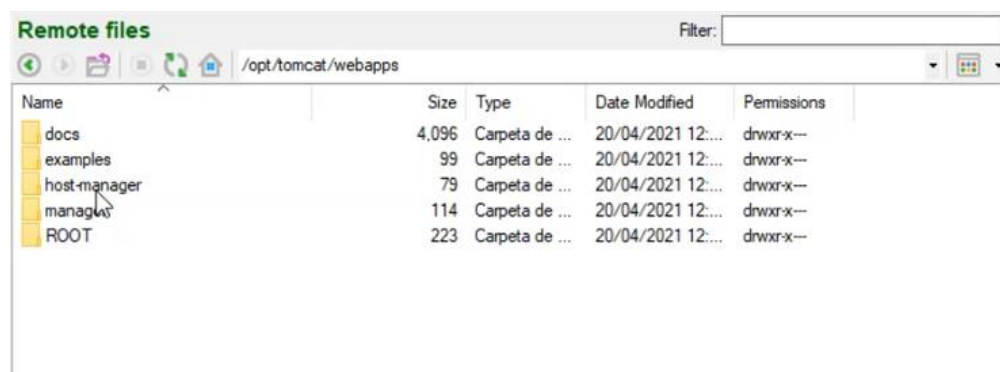


Figura 25: Directorio webapps.

En la carpeta cliente_webapps_map del repositorio de SIMOS se encuentran las aplicaciones para el core MxSIG sobre el que esta montado SIMOS.

Las siguientes carpetas deben ser copiadas dentro del directorio /opt/tomcat/webapps.

- **GeneraKML**: Servicio encargado de general KML a partir de las capas cargadas en SIMOS para su descarga.
- **mdmdownloadfile**: Servicio encargado de generar archivos csv con los datos de capas seleccionadas de SIMOS para su descarga.
- **mdmexport**: Servicio encargado de exportar capas de SIMOS.
- **mdmSearchEngine**: Servicio encargado de buscar ubicaciones por nombre y coordenadas en SIMOS.
- **mdmservices**: Servicio encargado de la consulta de datos de las capas en SIMOS.

En ocasiones, cuando los archivos de los servicios son desplegados a través de una conexión por FTP, no obtiene los permisos necesarios para que tomcat pueda leerlos, para eso, se ejecuta el siguiente comando.

chown tomcat:tomcat GeneraKML mdmdownloadfile mdmexport mdmSearchEngine mdmservices

Para corroborar que los cambios tomaron efecto se listan los archivos con el comando *ls -la*, como se muestra en la figura 26. Nótese que no es necesario tener los archivos con extensión war.

```
[root@localhost webapps]# ls -la
total 48256
drwxr-x--- 11 tomcat tomcat    256 sep  6  2021 .
drwxr-xr-x  9 tomcat tomcat    220 may 21  2021 ..
drwxr-x--- 15 tomcat tomcat   4096 may 21  2021 docs
drwxr-x---  4 tomcat tomcat    77 sep  6  2021 GeneraKML
-rw-r----- 1 tomcat tomcat 359448 may 26  2021 GeneraKML.war
drwxr-x---  6 tomcat tomcat    79 may 21  2021 host-manager
drwxr-x---  6 tomcat tomcat   114 may 21  2021 manager
drwxr-x---  4 tomcat tomcat    37 sep  6  2021 mdmdownloadfile
-rw-r----- 1 tomcat tomcat 10051443 may 26  2021 mdmdownloadfile.war
drwxr-x---  4 tomcat tomcat    97 sep  6  2021 mdmexport
-rw-r----- 1 tomcat tomcat 6299246 may 26  2021 mdmexport.war
drwxr-x---  8 tomcat tomcat   117 may 28  2021 mdmSearchEngine
drwxr-x---  4 tomcat tomcat    37 sep  6  2021 mdmservices
-rw-r----- 1 tomcat tomcat 32696442 may 26  2021 mdmservices.war
drwxr-x---  3 tomcat tomcat    223 may 21  2021 ROOT
```

Figura 26: Archivos listados.

9.2. Despliegue de Apache Solr

Apache Solr es un proyecto de código abierto que se utiliza como servidor para realizar búsquedas e indexaciones aceleradas. Este se ejecuta en contenedores-servidor como Tomcat. Solr es el motor utilizado para realizar búsquedas de ubicaciones en SIMOS. [14]

En la carpeta **Solr del repositorio de SIMOS** se encuentra una copia de Apache Solr compatible con la versión de Tomcat desplegada en el paso 5. Instalación de Tomcat. [1]

Nuevamente, a través de una conexión FTP con la quina con CentOS, se copia el archivo tomcat-solr.zip al siguiente directorio.

`/usr/local/instalacion`

Después, para descomprimir el archivo, se utiliza el siguiente comando. Nótese que se debe estar ubicado en el directorio `/usr/local/instalacion` para ejecutarlo apropiadamente.

`unzip tomcat-solr.zip`

De manera que el archivo descomprimido también se encuentra en la carpeta `/usr/local/instalacion`, como se muestra en la figura 27.

```
[root@mxsig instalacion]# ls -l
total 29424
-rw-r--r-- 1 root root 10523269 mar 30 07:02 apache-tomcat-8.5.65.tar.gz
-rw-r--r-- 1 root root 1201236 abr 5 2016 mapserver-7.0.0-1.el7.centos.x86_64.rpm
drwxr-xr-x 3 root root 188 abr 21 20:15 tomcat-solr
-rw-r--r-- 1 root root 18398875 mar 3 10:16 tomcat-solr.zip
[root@mxsig instalacion]#
```

Figura 27: Solr descomprimido.

Para que Tomcat ejecute Solr primero se debe agregar el contexto al archivo `mdmSearchEngine.xml` ubicado en el directorio `/opt/tomcat/conf/Catalina/localhost/`. Se ejecuta el siguiente comando para editar el archivo `mdmSearchEngine.xml` con nano.

`nano /opt/tomcat/conf/Catalina/localhost/mdmSearchEngine.xml`

Posteriormente, se agregan las siguientes líneas de código

```
<Context path="/solr" docBase="/usr/local/instalacion/tomcat-solr/solr-4.3.0.war" debug="0"
crossContext="true">
    <Environment name="solr/home" type="java.lang.String" value="/usr/local/instalacion/tomcat-
solr/solr-config/" override="true"/>
</Context>
```

Nótese que las líneas de código anteriores toman en consideración que Solr se encuentra descomprimido en el directorio `/usr/local/instalacion`.

Enseguida, dentro del directorio `/usr/local/instalacion/tomcat-solr` se encuentran una serie de librerías las cuales se deben copiar al directorio `/opt/tomcat/lib/` con el siguiente comando.

`cp *.jar /opt/tomcat/lib/`

Nótese que para que el comando se ejecute exitosamente se debe ubicar la línea de comandos en el directorio `/usr/local/instalacion/tomcat-solr`.

Además, se deben asignar permisos de escritura, lectura y ejecución a Solr en el directorio `mdm61`, esto se consigue con el siguiente comando.

`chmod -R 777 /usr/local/instalacion/tomcat-solr/solr-config/mdm61`

El funcionamiento del comando anterior se puede comprobar listando los archivos del directorio como se muestra en la siguiente figura.

```
[root@mxsig solr-config]# ls -l
total 4
drwxr-xr-x 2 root root 255 sep 27 2016 common-settings
drwxr-xr-x 2 root root 129 sep 26 2016 dih
drwxrwxrwx 3 root root 21 sep 26 2016 mdm61
-rw-r--r-- 1 root root 516 ago 2 2016 solr.xml
```

Figura 28: Privilegios de mdm61.

Finalmente, para que los cambios surtan efecto se reinicia Tomcat con el siguiente comando.

`systemctl restart tomcat`

Para corroborar que Solr se instaló adecuadamente, se puede consultar el Manager de Tomcat donde el despliegue de mdmSearchEngine se encuentra en estado true, como se muestra en la figura 29.

/GeneraKML	Ninguno especificado		true	0	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 30 minutos
/docs	Ninguno especificado	Tomcat Documentation	true	0	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 30 minutos
/host-manager	Ninguno especificado	Tomcat Host Manager Application	true	0	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 30 minutos
/manager	Ninguno especificado	Tomcat Manager Application	true	1	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 30 minutos
mdmSearchEngine	Ninguno especificado		true	0	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 30 minutos
/mdmdownloadfile	Ninguno especificado	mdm	true	0	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 10 minutos
/mdmexport	Ninguno especificado	MDMExport	true	0	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 30 minutos
/mdmservices	Ninguno especificado	Mapa Digital de Mexico v6.1	true	0	Arrancar Parar Recargar Replegar	Expirar sesiones sin trabajar ≥ 10 minutos

Figura 29: Estado de mdmSearchEngine.

Adicionalmente, se puede consultar Solr a través de un navegador web en el subdominio /mdmSearchEngine, como se ilustra en la figura 30.

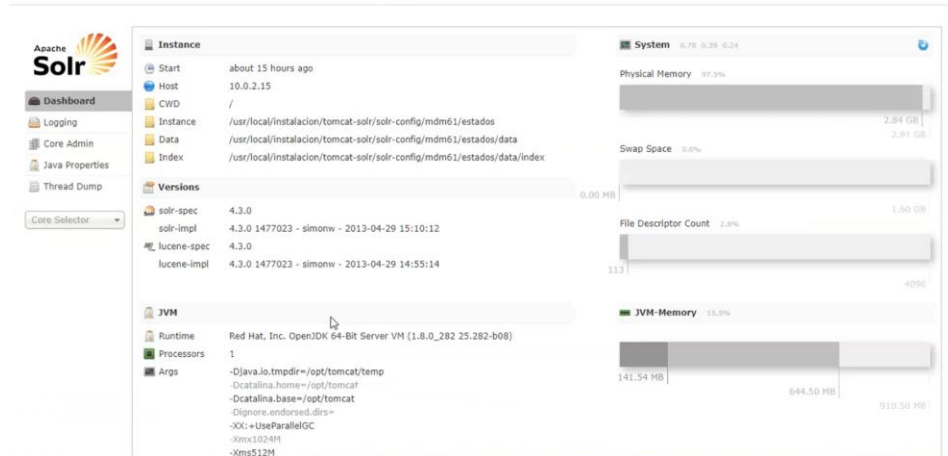


Figura 30: Interfaz gráfica de Solr.

9.3. Despliegue del cliente web

El cliente web contiene el core de MxSIG en el que se basa SIMOS y las modificaciones adicionales exclusivas de SIMOS.

Para desplegarlo, se debe copiar el contenido de la **carpeta core del repositorio de SIMOS** al siguiente directorio. [1]

`/var/www/html/mxsig`

Nótese que, si el directorio `/var/www/html/mxsig` no existe, se debe crear con el comando `mkdir`.

Dentro del directorio `/var/www/html/mxsig/js/frameworks` existe un directorio nombrado **openlayers**. Este directorio contiene los archivos de OpenLayers, una biblioteca de código abierto para JavaScript utilizada para mostrar mapas interactivos. Debido a una serie de discrepancias con el código core de MxSIG es necesario crear una copia de ese directorio con el nombre **OpenLayers**. Haciendo hincapié en que MxSIG y CentOS diferencian las letras mayúsculas. Lo anterior se realiza ejecutando el siguiente comando desde el directorio `/var/www/html/mxsig/js/frameworks`.

`cp -R openlayers/ ./OpenLayers`

De modo que dentro del directorio `/var/www/html/mxsig/js/frameworks` existen los directorios `openlayers` y `OpenLayers` con el mismo contenido, como se ilustra en la figura 31.

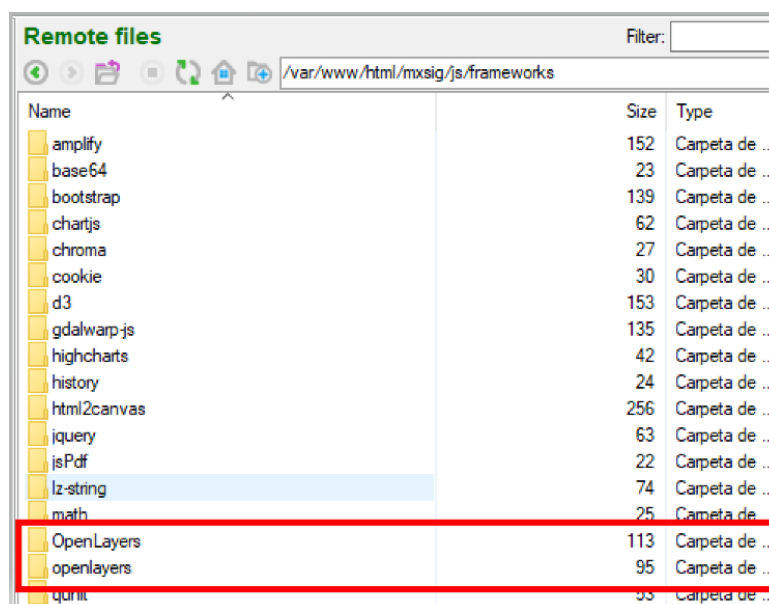


Figura 31: Directorios openlayers y OpenLayers.

Análogamente, dentro del directorio /var/www/html/mxsig/js/core/ui/widgets se encuentra el directorio fancybox que contiene los archivos de la biblioteca de código abierto de nombre homónimo, la cual permite mostrar imágenes y contenido “flotando” sobre páginas web. Debido a las discrepancias con MxSIG, también es necesario crear una copia de este directorio con el nombre fancyBox. Para esto, dentro del directorio /var/www/html/mxsig/js/core/ui/widgets se ejecuta el siguiente comando.

```
cp -R fancybox/ ./fancyBox/
```

Realizando los cambios anteriores, ya es posible consultar SIMOS desde un navegador web, aunque, en ocasiones puede mostrarse con algún error debido a que aún no está totalmente configurado, como se muestra en la figura 32.



Figura 32: Ventana de inicio de SIMOS.

9.4. Configuración básica de SIMOS

En esta sección se describe la configuración básica de SIMOS, por ejemplo, para carga de capas, control del layout y carrusel, entre otros.

9.4.1. Configuración de UI y Mapas.

Para iniciar con la configuración, desde el archivo `controlsConfig.js` en el directorio `/var/www/html/mxsig/config` se puede configurar lo siguiente:

- Mostrar u ocultar el minimapa.
- Mostrar u ocultar el tutorial de uso.
- Mostrar u ocultar la barra de capas.
- Abrir automáticamente la barra de temas al pasar por encima el cursor.
- Activar o desactivar la geolocalización.
- Activar o desactiva del cálculo de la elevación.
- Activar o desactivar la identificación de puntos y los marcadores.
- Activar o desactivar las cookies.

Únicamente se deben cambiar los valores de cada elemento a “true” o “false” según se requiera.

9.4.2. Configuración del origen de datos.

En esta configuración se establece la conexión entre el cliente de SIMOS y los servicios que se desplegaron en el directorio `/opt/tomcat/webapps`.

Se deben modificar todas las direcciones IP y colocar la dirección del servidor con CentOS en el archivo **`dataSourceConfig.js`** dentro del directorio `/var/www/html/mxsig/config`. Nótese que por default la dirección que está en el archivo es la de la maquina local 127.0.0.1, aunque esta dirección puede funcionar adecuadamente, se recomienda modificar por la dirección IP asignada a la máquina.

Al hacer lo anterior, las siguientes funciones quedan habilitadas:

- Búsqueda de ubicaciones.
- Identificación de puntos.
- Historial de capas encendidas.
- Exportación de capas.
- Compartir capas.
- Enviar por correo electrónico.
- Creación de buffers.
- Identificar detalles de punto.

9.4.3. Configuración de las capas.

Para comenzar con la configuración de las capas, dentro del directorio /opt se debe crear el directorio map. Se puede utilizar el siguiente comando o crear el directorio desde una ventana de FTP.

```
mkdir /opt/map
```

Dentro de /opt/map se deben copiar los siguientes archivos, disponibles en la carpeta **map del repositorio de SIMOS**. [1]

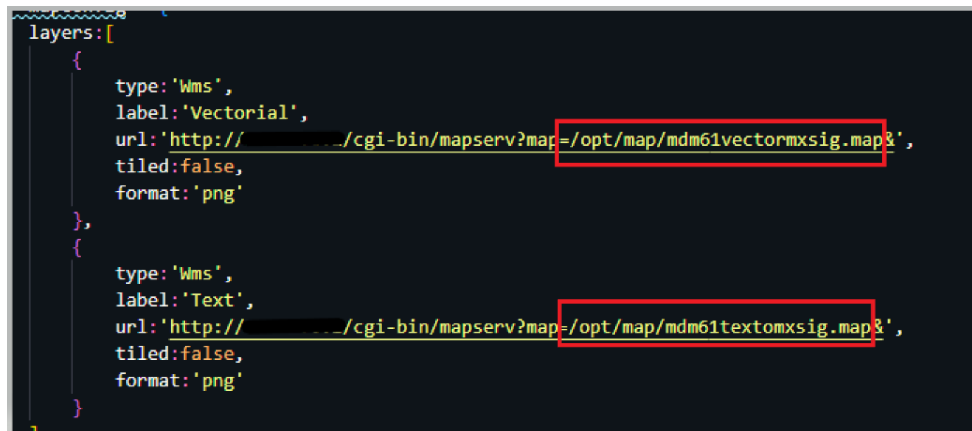
- fonts.zip
- mdm61leyendamxsig.map
- mdm61textomxsig.map
- mdm61vectormxsig.map
- sysms.zip

Los archivos fonts.zip y sysms.zip deben ser descomprimidos dentro del directorio /opt/map con el comando unzip.

Las capas que se muestran por el cliente de SIMOS se consumen por el cliente a través del archivo mapConfig.js del directorio /var/www/html/mxsig/config.

Dentro del archivo mapConfig.js se deben cambiar todas las direcciones IP a la dirección IP de la máquina con CentOS. Nótese que por default la dirección que está en el archivo es la de la máquina local 127.0.0.1, aunque esta dirección puede funcionar adecuadamente, se recomienda modificar por la dirección IP asignada a la máquina.

Se debe corroborar que para los valores de url de los objetos de Vectorial y Text, la ruta donde están los archivos mdm61textomxsig.map y mdm61vectormxsig.map es la indicada, como se muestra en la figura 33.



```
layers:[
  {
    type: 'Wms',
    label: 'Vectorial',
    url: 'http://[IP]/cgi-bin/mapserv?map=/opt/map/mdm61vectormxsig.map&',
    tiled: false,
    format: 'png'
  },
  {
    type: 'Wms',
    label: 'Text',
    url: 'http://[IP]/cgi-bin/mapserv?map=/opt/map/mdm61textomxsig.map&',
    tiled: false,
    format: 'png'
  }
]
```

Figura 33: Configuración de layers en mapConfig.js

Un aspecto importante de la configuración es que el direccionamiento al archivo mdm61leyendamxsig.map se hace en el archivo dataSourceConfig.js dentro del directorio /var/www/html/mxsig/config, como en la sección 9.4.2. Configuración del origen de datos. Se debe verificar la IP y que el archivo este en la ruta correcta, como se muestra en la figura 34.

```
school:'',
//Otras Url de informacion-----mdm61leyendamxsig
legendUrl:'http://[redacted]/cgi-bin/mapserv?map=/opt/map/mdm61leyendamxsig.map&
Request=GetLegendGraphic&format=image/png&Version=1.1.1&Service=WMS&LAYERS=';
synonyms:{
  list:{
    /*farmacia:['botica','drogeria'],
    banco:['cajero'],
```

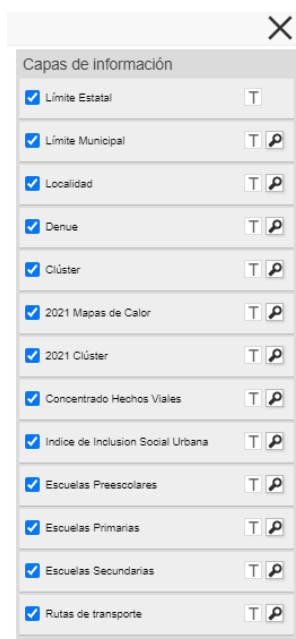
Figura 34: Direccionamiento al archivo mdm61leyendamxsig.map

De vuelta en el archivo mapConfig.js del directorio /var/www/html/mxsig/config se puede configurar la vista inicial de SIMOS, a través de un par de coordenadas que se colocan en el parámetro initialExtent, como se muestra en la figura 35. Las coordenadas por default muestran el estado de Colima, México. Nótese que, el valor del parámetro lon es un par de coordenadas que representan la esquina superior izquierda de la vista de SIMOS, análogamente, el parámetro lat representa la esquina inferior derecha de la vista.

```
},
projection:"EPSG:4326",
initialExtent:{lon:[-103.77878818791268,18.689033437490526],lat:[-103.89139804383733,20.
27364672788709]}, // Vista inicial, Michoacán
//initialExtent:{lon:[-120.9103, 10.9999 ],lat:[-83.3810,34.5985]},
//initialExtent:{lon:[-120.9103, 10.9999 ],lat:[-83.3810,34.5985]}
```

Figura 35: Coordenadas de la vista inicial de SIMOS.

La versión de SIMOS disponible en el repositorio incluye las capas pre-cargadas de la figura 36.



Capas de información	
<input checked="" type="checkbox"/> Límite Estatal	T
<input checked="" type="checkbox"/> Límite Municipal	T P
<input checked="" type="checkbox"/> Localidad	T P
<input checked="" type="checkbox"/> Denue	T P
<input checked="" type="checkbox"/> Clúster	T P
<input checked="" type="checkbox"/> 2021 Mapas de Calor	T P
<input checked="" type="checkbox"/> 2021 Clúster	T P
<input checked="" type="checkbox"/> Concentrado Hechos Viales	T P
<input checked="" type="checkbox"/> Índice de Inclusión Social Urbana	T P
<input checked="" type="checkbox"/> Escuelas Preescolares	T P
<input checked="" type="checkbox"/> Escuelas Primarias	T P
<input checked="" type="checkbox"/> Escuelas Secundarias	T P
<input checked="" type="checkbox"/> Rutas de transporte	T P

Figura 36: Capas pre-cargadas.

De igual forma cuenta con el tema pre-cargado del DENUE 2022 y los hechos viales del estado de Colima 2021.

Para que las capas y temas pre-cargados funcionen adecuadamente se deben agregar las credenciales de la base de datos en los archivos `mdm61vectormxsig.map` y `mdm61leyendamxsig.map` del directorio `/opt/map/`, como se muestra en la figura 37. Por defecto tienen los valores que se enlistan a continuación.

- `user=postgres`
- `password=password1`
- `dbname=mdm6data`
- `host=127.0.0.1`
- `port=5432`

```
CONNECTIONTYPE postgres
CONNECTION "user=postgres password=password1 dbname=mdm6data host=127.0.0.1 port=5432"
PROCESSING "CLOSE_CONNECTION=DEFER"
```

Figura 37: Configuración de las credenciales de base de datos en el archivo `mdm61vectormxsig.map`

Nótese que se deben agregar las credenciales para cada una de las capas definidas en ambos archivos.

Para configurar las capas que se muestran en el árbol de selección de capas y la manera en que estas se organizan, se debe modificar el archivo `tree.js` en el directorio `/var/www/html/mxsig/config`. Las capas se organizan por temas y grupos. En la figura 38 se muestra un fragmento del archivo `tree.js` donde se observa el grupo "Límites del Marco Geoestadístico Nacional", identificado como G11. Dentro del grupo se encuentran las capas de Límite Estatal y Límite Municipal, identificadas como c100 y c101. Por convención se recomienda identificar a los temas iniciando con la letra "T" a los grupos con la letra "G" y las capas con la letra "c", seguidos de un número consecutivo o un nombre significativo.

```
layers: {
  groups: {
    G11: {
      label: "Límites del Marco Geoestadístico Nacional",
      layers: {
        c100: {
          label: "Límite Estatal",
          synonymous: ["limite", "estatal"],
          scale: 0,
          position: 40,
          active: false,
          texts: {
            scale: 0,
            active: false,
          }, //,
          //metadato: 'http://geoweb.inegi.org.mx/WSCBuscador/MuestraMetadatos.jsp?par1=523928&par2=INEGI1'
        },
        c101: {
          label: "Límite Municipal",
          synonymous: ["municipio", "municipales", "municipal"],
          scale: 866685,
          position: 41,
          active: false,
          texts: {
            scale: 0,
            active: false,
          }, //,
        },
      },
    },
  },
}
```

Figura 38: Archivo `tree.js`

9.4.4. Configuración de Solr

Para que la indexación funcione adecuadamente, es necesario configurar Apache Solr, añadiendo algunos motores para la búsqueda.

Desde el directorio `/usr/local/instalacion/tomcat-solr`, que es donde se descomprimió el archivo de Solr, descrito en la sección 9.2 Despliegue de Apache Solr, se encuentra el directorio `solr-config`, el cual contiene el archivo `solr.xml`.

El contenido del archivo `solr.xml` se encuentra como en la figura 39.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <solr persistent="true">
3   <cores adminPath="/admin/cores" defaultCoreName="busq-entidades">
4     <core name="busq-entidades" instanceDir="mdm61/estados" properties="../../common-settings/solr.properties"/>
5     <!--<core name="busq-municipal" instanceDir="mdm61/municipal" properties="../../common-settings/solr.properties"/>
6     <core name="busq-localidadesurb" instanceDir="mdm61/localidadesurb" properties="../../common-settings/solr.properties"/>
7   </cores>
8 </solr>
9

```

Figura 39: Contenido inicial del archivo solr.xml

Se debe editar el archivo `solr.xml` y eliminar los comentarios de la sección de `cores`. Los comentarios en xml se realizan con los caracteres `<!--` y `-->`.

A continuación, para crear los directorios de trabajo de los core que se descomentaron, `mdm61/estados` y `mdm61/localidadesurb`, se toma como base el core que ya se encuentra precargado, `mdm61/estados`.

Dentro del directorio `/usr/local/instalacion/tomcat-solr/solr-config/mdm61` se encuentra el directorio del core “estados”. Utilizando los siguientes comandos se hace una copia idéntica del core pero con el nombre “municipal” y “localidadesurb”, además se les da permisos de escritura, lectura y ejecución, respectivamente.

```

cp -R estados/ ./municipal/
chmod -R 777 municipal/
cp -R estados/ ./localidadesurb/
chmod -R 777 localidadesurb/

```

En seguida, dentro del directorio `/usr/local/instalacion/tomcat-solr/solr-config/mdm61/estados` se encuentra el directorio `conf` el cual contiene la configuración de la conexión con la base de datos a través del archivo `db-data-config.xml`.

Dentro del archivo `data-config.xml` se deben colocar las credenciales de la base de datos de PostgreSQL, como se ilustra en la figura 40. Además, se debe colocar el nombre correcto del core en el apartado de `name`. El nombre del core debe ser el mismo que en el archivo `solr.xml`.

```

1  <dataConfig>
2    <dataSource driver="org.postgresql.Driver"
3      url="jdbc:postgresql://127.0.0.1:5432/mdm6data"
4      user="postgres"
5      password="XXXXXXXXXX" />
6    <document>
7      <entity name="busq-entidades" query="SELECT gid, gid as id, busqueda, tipo, coord
8    </entity>
9    </document>
10  </dataConfig>
11

```

Figura 40: Archivo data-config.xml

Asimismo, se debe corroborar que en el valor del query se coloque el esquema y tabla correctos. En la figura 41 se ilustra el esquema y tabla para las entidades. Se recomienda corroborar que los campos del query también estén presentes en la tabla dentro de la base de datos, de lo contrario se pueden generar errores.

```

query="SELECT gid, gid as id, busqueda, tipo, coord_merc, locacion, tabla, nombre from mdm.ent"

```

Figura 41: Query en el archivo data-config.xml

Análogamente, esta configuración debe realizarse para los core busq-municipal y busq-localidadesurb. Nótese que la tabla de localidadesurb es mdm.l y la de municipal es mdm.mun.

Por otro lado, en el archivo solrconfig.xml dentro del directorio /usr/local/instalacion/tomcat-solr/solr-config/mdm61/estados/conf en el apartado de shards se deben agregar los core busq-municipal y busq-localidadesurb como se muestra en la figura 42.

```

213  <requestHandler name="/shard" class="solr.SearchHandler" default="false">
214    <lst name="default">
215      <str name="shards">${server.ip}:${server.port}/${solr.app.context}/busq-entidades,${server.ip}:${server.port}/${solr.app.context}/busq-municipal,${server.ip}:${server.port}/${solr.app.context}/busq-localidadesurb</str>
216      <str name="q.op">AND</str>
217      <str name="qf">busqueda</str>
218      <str name="defType">edismax</str>
219    </lst>
220  </requestHandler>
221

```

Figura 42: Configuración de shards en el archivo solrconfig.xml

La configuración de shard se puede replicar en los directorios de municipal y localidadesurb, sin embargo, es esencial que se realice en el directorio de estados, debido a que este es el motor por default.

Para aplicar los cambios, se reinicia tomcat con el siguiente comando.

`systemctl restart tomcat.`

Después, a través de un navegador web se accede a Solr en la ruta /mdmSearchEngine y los tres motores deben estar disponibles, como se ilustra en la figura 43.

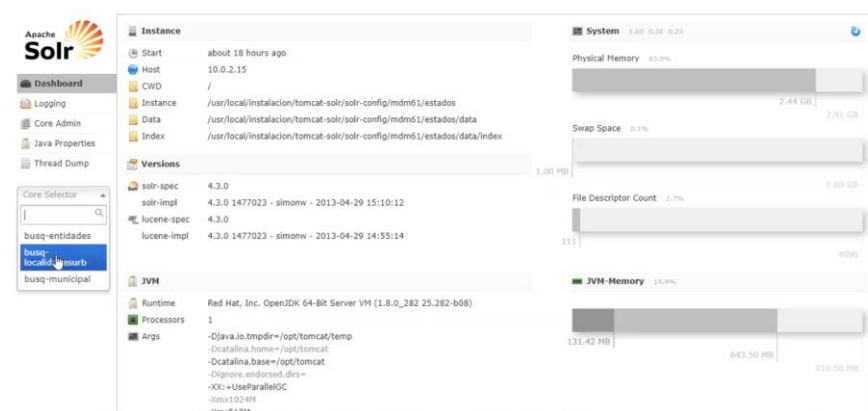


Figura 43: Motores agregados a Solr.

Para alimentar el motor con datos, se selecciona busq-entidades y luego se da click en Dataimport de lado inferior izquierdo, como en la figura 44.

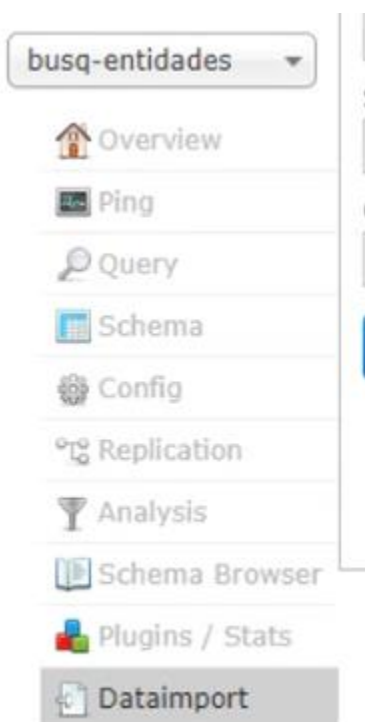


Figura 44: Botón de importación para el motor busq-entidades.

Posteriormente, se especifica que se realizara una importación completa y se seleccionan las opciones verbose, clean y commit, además, se selecciona la entidad correspondiente al motor de entidades, busq-entidades. Se ejecuta y la indexación debe completarse. Lo anterior se ilustra en la figura 45.

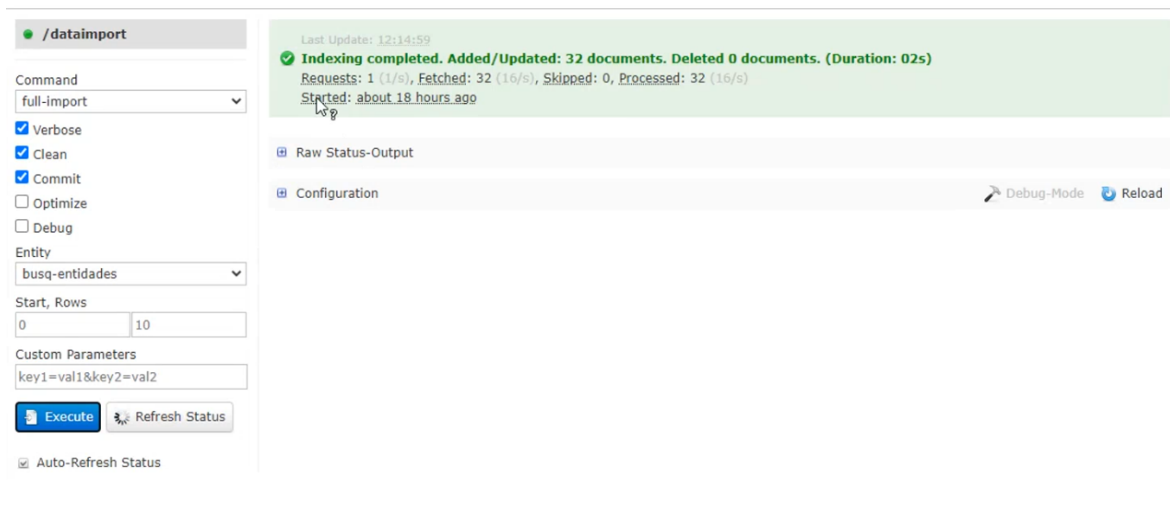


Figura 45: Proceso de indexación completado.

Análogamente, se deben realizar la misma importación para los motores busq-localidad y busq-localidadesurb.

Para corroborar que la importación funcionó, se puede ejecutar un query o consulta y ver si este arroja algún resultado, como en la figura 46.

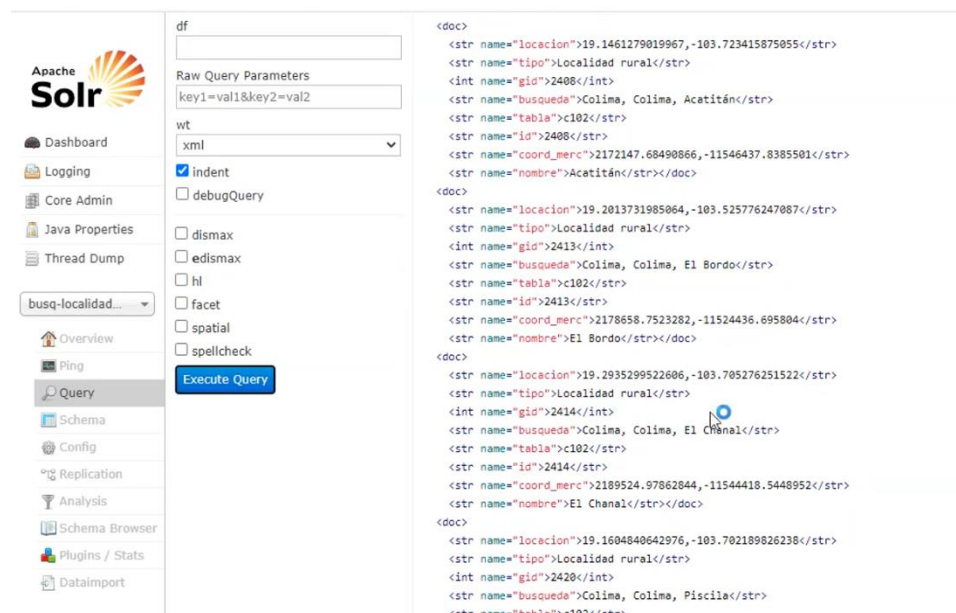


Figura 46: Consulta de los datos del motor busq-localidad.

También, es posible utilizar el buscador de SIMOS y corroborar que funciona, como en la figura 47.

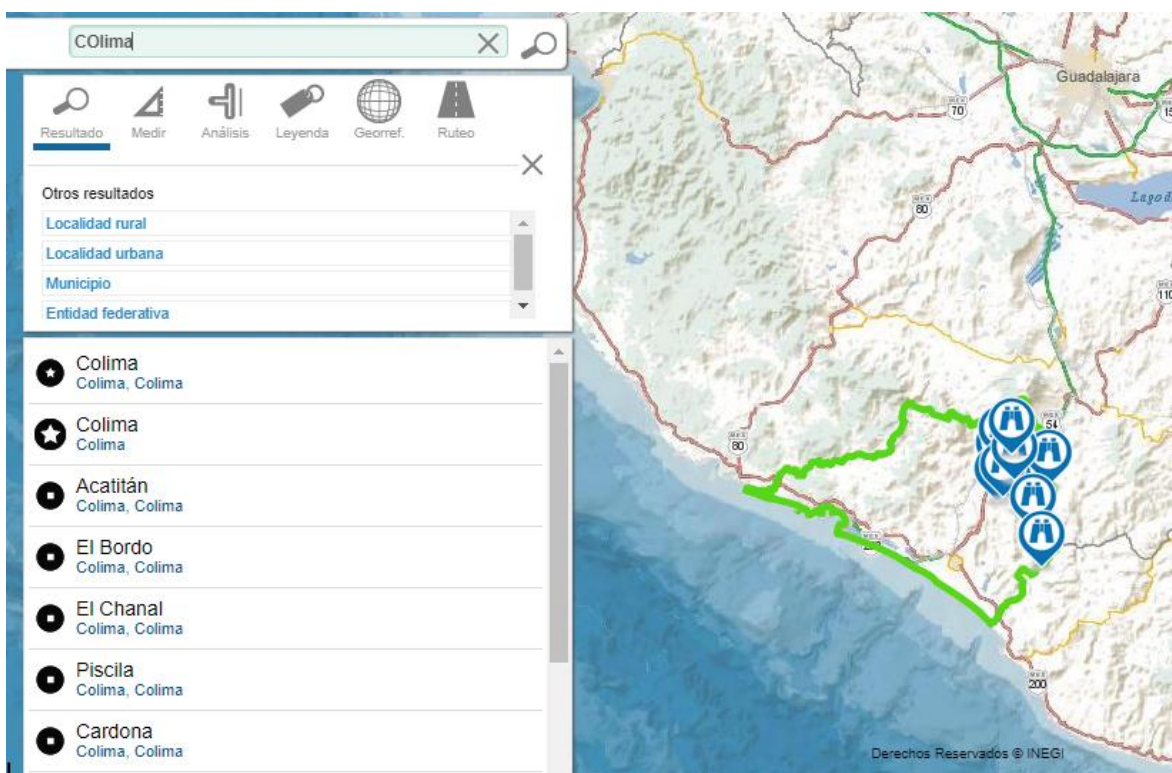


Figura 47: Buscador de SIMOS con datos indexados.

9.4.5. Configuración de identificación y análisis espacial

Para configurar la conexión entre los servicios de identificación y análisis se deben añadir las credenciales necesarias en los servicios desplegados en la sección 1.1 Despliegue de servicios.

Primero, en el directorio `/opt/tomcat/webapps/mdmservices/WEB-INF/classes/` se ubica el archivo `Servers.xml` en que se deben agregar las credenciales de conexión con la base de datos, como se ilustra en la figura 48.



```
webapps > mdmservices > web-inf > classes > config > Servers.xml
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <servers>
3      <server>
4          <alias>servidoresote</alias>
5          <ip>127.0.0.1</ip>
6          <port>5432</port>
7          <user>postgres</user>
8          <password>passowrd1</password>
9          <url>jdbc:postgresql://%s:%s/%s</url>
10         <driverClass>org.postgresql.Driver</driverClass>
11         <validationQuery>select version()</validationQuery>
12     </server>
13 </servers>
14
```

Figura 48: Credenciales de conexión a la base de datos en el archivo Servers.xml

Después, dentro del directorio /opt/tomcat/webapps/mdmservices/WEB-INF/classes/config/xml se encuentra el archivo mdm6.xml en el que se definen las capas que tendrán capacidad de identificación y análisis.

Si bien, SIMOS ya contiene una serie de capas pre-cargadas para el análisis espacial, se recomienda que al agregar una capa nueva con capacidad de análisis, se tome el siguiente ejemplo y se agregue en el archivo mdm6.xml

```
<table search="true" identify="true" buffer="true" user_alias="Nombre del
análisis" projects="mdm6">
    <server>servidoresote</server>
    <database>mdm6data</database>
    <schema>Nombre del esquema</schema>
    <name>Nombre de la tabla</name>
    <alias>Alias de la tabla</alias>
    <geometry>the_geom</geometry>
    <projection>900913</projection>
    <resolution>
        <min>0.298582141</min>
        <max>305.748113098</max>
    </resolution>
    <fields>
        <field identify="true">
            <name>gid</name>
            <alias>ID</alias>
        </field>
    </fields>
</table>
```



```
</field>

<field>
  <name>gid</name>
  <alias>buffer</alias>
</field>
<field identify="true" >
  <name>Campo1</name>
  <alias>Campo 1</alias>
</field>
<field identify="true">
  <name>Campo2</name>
  <alias>Campo 2</alias>
</field>

<field search_display="false" query_display="true">
  <functions>
    <function order="2">
      <fname>ST_AsText</fname>
    </function>
    <function order="1">
      <fname>ST_Envelope</fname>
    </function>
  </functions>
  <name>the_geom</name>
  <alias>ubicacion</alias>
</field>
<field search_display="false" query_display="true">
  <functions>
    <function order="2">
      <fname>ST_AsText</fname>
    </function>
    <function order="1">
      <fname>ST_PointOnSurface</fname>
    </function>
  </functions>
  <name>the_geom</name>
  <alias>coordenada</alias>
</field>
</fields>
```



```
<search>
  <!-- el primer field segun definido aqui es el obligatorio -->
  <field type="tsearch">
    <name>spvector</name>
  </field>
  <field type="tsearch">
    <name>spvectorref</name>
  </field>
</search>
</table>
```

Al añadir una nueva capa al análisis se debe considerar lo siguiente:

- El server es el mismo que el definido en el archivo `/opt/tomcat/webapps/mdmservices/WEB-INF/classes/config/Servers.xml`
- Database es el nombre de la base de datos de PostgreSQL.
- Schema es el esquema donde se almacena la tabla con los datos de la capa en la base de datos.
- Name es el nombre de la tabla que contiene los datos de la capa.
- Alias es el nombre por el que se hace referencia a la capa, el cual se define en los archivos `/opt/map/mdm61vectormxsig.map` y `/var/www/html/mxsig/config/tree.js`.
- Geometry es el nombre del campo que contiene los datos geométricos de los registros de la tabla.
- Projection es la proyección espacial de la capa, para que SIMOS la reconozca adecuadamente siempre debe ser 900913.
- Resolution es la escala mínima y máxima de zoom en la que la capa podrá ser analizada.
- Fields son los campos provenientes de la tabla que desean mostrarse como resultado del análisis. Name es el nombre del campo en la base de datos y alias la manera en que aparecen en la interfaz gráfica. Además, se puede agregar el parámetro `identify`, que cuando es true añade el campo a la descarga de datos, cuando esta ocurre.
- Los campos con alias ID, buffer, coordenada y ubicación son obligatorios.
- Se recomienda dejar todos los otros parámetros con sus valores por defecto, debido a que son parte de la configuración de la aplicación.

Por otro lado, para permitir las conexiones a los servicios, se debe agregar la IP de la máquina con CentOS al archivo de orígenes permitidos, `allow-origin.xml`, en el directorio `/opt/tomcat/webapps/mdmservices/WEB-INF/classes/`. Se recomienda agregar la dirección junto con los puertos donde se ejecuta Apache y Tomcat, si se cuenta con una IP fija o un dominio, también deben ser agregados. Lo anterior se muestra en la figura 49.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <allow-origin>
3      <origin>http://localhost</origin>
4      <origin>http://127.0.0.1</origin>
5      <origin>http://127.0.0.1:80</origin>
6      <origin>http://127.0.0.1:8080</origin>
7
8  </allow-origin>

```

Figura 49: Orígenes permitidos.

Finalmente, para aplicar los cambios, se reinicia tomcat con el siguiente comando.

systemctl restart tomcat.

Sí los cambios no se reflejan en el navegador web, se debe recargar la página y eliminar las cookies.

10. Funcionalidades adicionales

10.1. Visualización de capas tipo cluster y mapa de calor

Dado que SIMOS utiliza MapServer para presentar dinámicamente las capas que están almacenadas en la base de datos este tiene la capacidad de generar capas de cluster y mapas de calor en base a capas de tipo punto.

10.1.1. Capas Cluster.

Las capas tipo cluster fueron añadidas en la versión 6 de MapServer, en la que se añadió la capacidad de combinar características de diferentes puntos de una capa con base en su posición relativa. Únicamente se pueden generar clusters con capas de tipo punto. [15]

Una vez que una capa está cargada en la base de datos de SIMOS, se debe agregar en el archivo `mdm61vectormxsig.map` ubicado en el directorio `/opt/map/` como se ejemplifica en la sección 9.4.5 Configuración de identificación y análisis espacial.

Cuando la capa tipo punto ha sido agregada en el archivo `mdm61vectormxsig.map`, se puede utilizar para crear capas con funciones agregadas, como cluster. Posterior a la declaración de la capa, se puede utilizar el siguiente fragmento de código para generar un cluster.

```

LAYER
    NAME 'cCluster'

```



```
GROUP 'mdm6'
CONNECTIONTYPE postgis
CONNECTION "user=postgres password=password1 dbname=mdm6data
host=127.0.0.1 port=5432"
DATA "the_geom from prueba.tabla using unique gid using
srid=900913"
PROCESSING "CLOSE_CONNECTION=DEFER"
MAXSCALE 60000000
PROJECTION
    "init=epsg:900913"
END #end projection

TYPE point
STATUS ON
CLUSTER
    MAXDISTANCE 250 # in pixels
    REGION "ellipse" # can be rectangle or ellipse
    #GROUP (expression) # an expression to create separate groups
for each value
    #FILTER (expression) # a logical expression to specify the
grouping condition
END
LABELITEM "Cluster_FeatureCount"
CLASS
    STYLE
        SIZE 50
        SYMBOL "circulo"
        COLOR 170 146 4
        OUTLINECOLOR 170 146 4

        WIDTH 3
    END #end style
    LABEL
        ANGLE auto
        SIZE 16
        COLOR 0 0 0
        MINFEATURESIZE 100
        TYPE truetype
        FONT arial #times # verdana
        ANTIALIAS true
        #OUTLINECOLOR 220 220 220
```

```

    OUTLINEWIDTH 1
    MAXLENGTH 9
    #WRAP ","
    ALIGN center
    POSITION CC
    PARTIALS false

    END
  END # end class
END # end layer

```

Al añadir la definición de la capa de cluster se debe considerar lo siguiente:

- Name es el nombre con el que se va a identificar a la capa.
- Connection contiene la cadena de conexión con la base de datos.
- Data contiene una consulta al esquema y tabla que contienen los datos de la capa. Nótese que, el campo the_geom es el campo por defecto que contiene los datos geométricos.
- En la sección de CLUSTER se coloca la distancia en pixeles y la forma geométrica en que se van a agrupar los puntos.
- Dentro de la sección CLASS se coloca el estilo de la forma geométrica que se presenta en la interfaz gráfica de SIMOS, por ejemplo, en el fragmento de código se agrupan con forma de círculo y se presenta un círculo en la interfaz, con un tamaño de 50px, un relleno de color rgb(170,146,4) y una línea externa del mismo color, como se ilustra en la figura 50.
- En la sub-sección LABEL se coloca lo referente al texto que muestran las agrupaciones, se puede definir su ángulo, posición, color y tipo de fuente, entre otros, como se muestra en la figura 50.

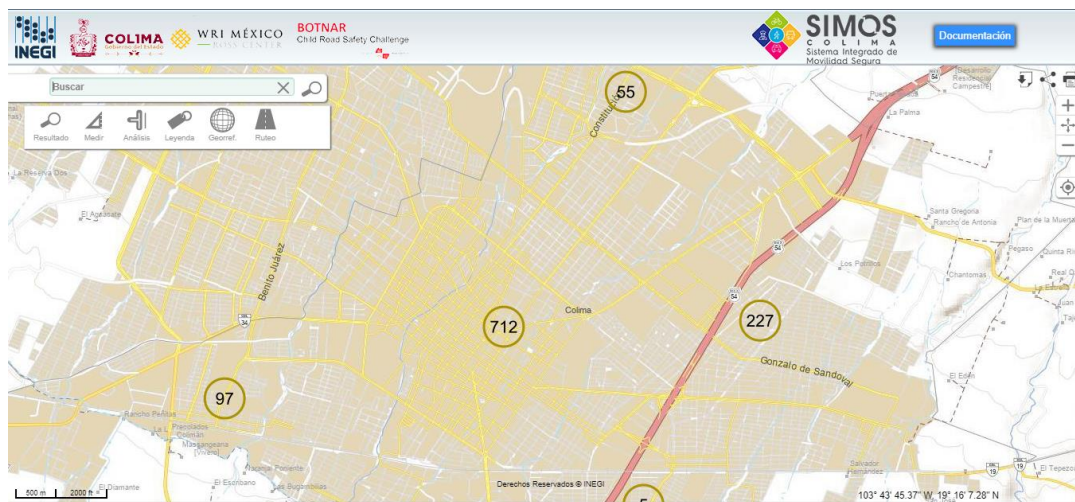


Figura 50: Capa cluster.

Por último, para poder seleccionar la capa de cluster desde un tema o el árbol de capas, se debe agregar la capa a través del nombre declarado en el archivo `mdm61vectormxsig.map` al archivo `tree.js` del directorio `/var/www/html/mxsig/config`, como se describe en la sección 9.4.3 Configuración de las capas.

10.1.2. Capas Mapa de Calor.

El mapa de calor es un método popular para representar el esparcimiento de datos de una capa, estas capas están fuertemente influenciadas por la distancia entre cada muestra o punto de las capas. Las capas de mapa de calor utilizan la estimación Kernel de la densidad para calcular los contenedores y colores con los que se muestran los puntos. Las capas de tipo mapa de calor solo se pueden realizar tomando como base una capa tipo punto. [16]

Una vez que una capa tipo punto está cargada en la base de datos de SIMOS, se debe agregar en el archivo `mdm61vectormxsig.map` ubicado en el directorio `/opt/map/` como se ejemplifica en la sección 9.4.5 Configuración de identificación y análisis espacial.

Cuando la capa tipo punto ha sido agregada en el archivo `mdm61vectormxsig.map`, se puede utilizar para crear capas con funciones agregadas, como mapa de calor. Posterior a la declaración de la capa, se puede utilizar el siguiente fragmento de código para generar un mapa de calor.

```
LAYER
    NAME "cmapas_calor"
    TYPE raster
    CONNECTIONTYPE kerneldensity
    CONNECTION "cCapa_punto"
    STATUS ON
    PROCESSING "RANGE_COLORSPACE=HSL"
    PROCESSING "KERNELDENSITY_RADIUS=9"
    PROCESSING "KERNELDENSITY_COMPUTE_BORDERS=ON"
    PROCESSING "KERNELDENSITY_NORMALIZATION=3"
    OFFSITE 0 0 0
    CLASS
        STYLE
            COLORRANGE "#0000ff00" "#0000ffff"
            DATARANGE 0 32
        END #END STYLE
        STYLE
            COLORRANGE "#0000ffff" "#ff0000ff"
            DATARANGE 32 260
        END #END STYLE
    END #END CLASS
END #END LAYER
```

Al añadir la definición de capa de mapa de calor se debe considerar lo siguiente:

- NAME es el nombre con el que se identificara a la capa.
- TYPE debe tener el valor de raster, que es el tipo de capa con el que se genera el mapa de calor.
- CONNECTIONTYPE debe ser kerneldensity, la técnica estadística con la que se calcula el mapa y sus colores.
- CONNECTION debe tener el valor de la capa tipo punto en la que se basa el mapa de calor y la cual debe estar previamente definida en el archivo `/opt/map/ mdm61vectormxsig.map`.

- En la sección CLASS se definen los colores y los rangos en los que se agrupan los datos, por ejemplo, en el fragmento de código se definen dos rangos, uno de 0 a 32 con un gradiente de colores de #0000ff00 a #0000ffff y otro de 32 a 260 con un gradiente de colores de #0000ffff a #ff0000ff. Nótese que los colores se definen por su valor hexadecimal.

Por último, para poder seleccionar la capa de mapa de calor desde un tema o el árbol de capas, se debe agregar la capa a través del nombre declarado en el parámetro NAME al archivo tree.js del directorio /var/www/html/mxsig/config, como se describe en la sección 9.4.3 Configuración de las capas.

El resultado del mapa de calor es el que se muestra en la figura 51.

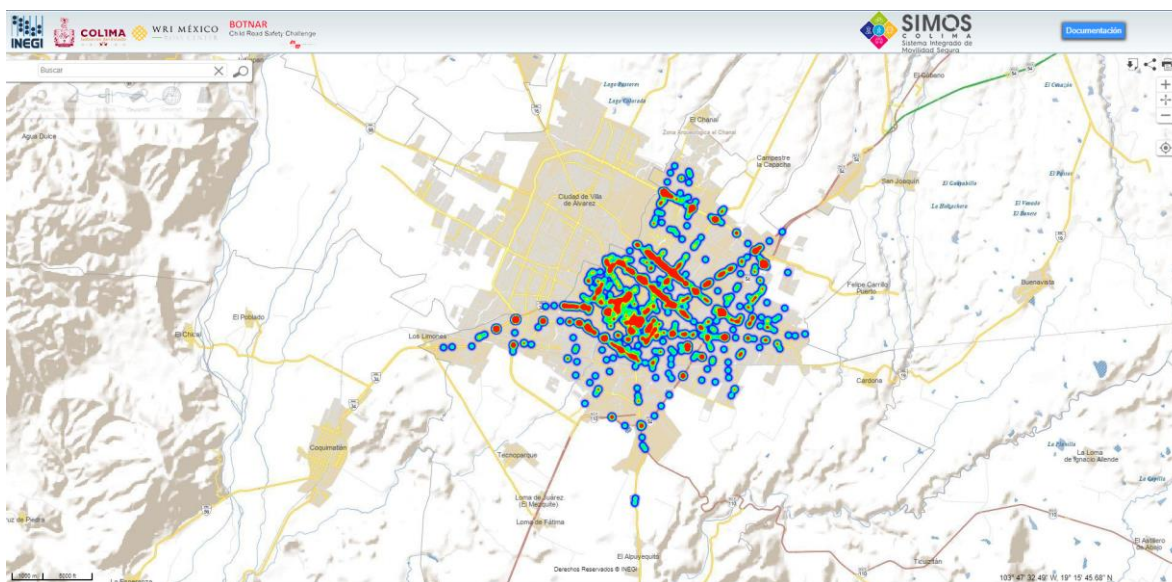


Figura 51: Mapa de calor.

10.2. Leyenda personalizada

En ocasiones se requiere colocar avisos o cuadros de texto con leyendas personalizadas para capas en las que SIMOS no puede interpretar la simbología, por ejemplo, para capas de tipo mapa de calor.

SIMOS soporta la inyección de código y HTML a través de JQuery, permitiendo desplegar leyendas personalizadas que describen una capa de tipo mapa de calor cuando esta es seleccionada. Para añadir una leyenda personalizada se debe crear primero un contenedor HTML en donde se inyecte el código, se recomienda que este contenedor se coloque en el archivo /var/www/html/mxsig/index.html. Como se muestra en la figura 52, se coloca un elemento <div> en el index.html dentro del body y se le asigna el id "popup_heat". Note que el contenido del div es irrelevante, pues este será inyectado con JQuery.


```

78 </style>
79
80 <body>
81 <!-- Indicadores para mapa de calor -->
82 <div class="heat_container">
83 <div id="popup_heat" flagged="false">
84 <div class="heat-action">
85 <span></span>
86 </div>
87 <div class="heat-header">
88 <h3>Leyenda Mapa de Calor</h3>
89 </div>
90 <div class="heat-content">
91 <div>
92 <div class="squareR"></div>
93 <div>&nbsp;&nbsp; Muy alta concentración</div>
94 </div>
95 <br />
96 <div>
97 <div class="squareO"></div>
98 <div>&nbsp;&nbsp; Alta concentración</div>
99 </div>
100 <br />
101 <div>
102 <div class="squareY"></div>
103 <div>&nbsp;&nbsp; Media concentración</div>
104 </div>
105 <br />
106 <div>
107 <div class="squareG"></div>
108 <div>&nbsp;&nbsp; Baja concentración</div>
109 </div>
110 <br />
111 <div>
112 <div class="squareB"></div>
113 <div>&nbsp;&nbsp; Muy baja concentración</div>
114 </div>
115 </div>
116 </div>
117 </div>
118

```

Figura 52: Elemento HTML contenedor.

Los estilos para el contenedor se pueden almacenar en un archivo CSS externo, en línea o declarados dentro del mismo archivo index.html. Como se espera que la leyenda solo sea visible cuando la capa de mapa de calor esta seleccionada, se utiliza el atributo display:none para que el contenedor este oculto. Además, para que el contenedor se muestre frente a la interfaz de SIMOS, se le debe colocar el atributo z.index con un valor mayor al del contenido de SIMOS, por ejemplo, 4, como se muestra en la figura 53.

```

/* Estilos del popup de leyendas del mapa de calor */
#popup_heat {
  display: none;
  background-color: #fff;
  width: 18%;
  z-index: 4;
  padding: 2rem;
  position: absolute;
  bottom: 10%;
  left: 0;
  border-radius: 1rem;
  -webkit-box-shadow: 0px 0px 17px 0px rgba(0, 0, 0, 0.75);
  -moz-box-shadow: 0px 0px 17px 0px rgba(0, 0, 0, 0.75);
  box-shadow: 0px 0px 17px 0px rgba(0, 0, 0, 0.75);
  font-size: 1.2rem;
}
#popup_heat h3 {
  text-align: center;
  font-size: 1.5rem;
  margin: 3rem 0 2rem 0;
}
#popup_heat span {
  font-size: 1.5rem;
  font-weight: 700;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0.5rem 1rem;
  border-radius: 4px;
  margin: 1rem 1rem 2rem 0;
  background-color: #gray;
  color: #aliceblue;
  transition: all 0.6s ease-in-out;
}
#popup_heat span:hover {
  background-color: #lightgrey;
  color: #000;
  cursor: pointer;
}

```

Figura 53: Estilos del contenedor HTML tipo pop-up

En seguida, el archivo donde se manejan los eventos con JQuery es el archivo `/var/www/html/mxsig/js/core/ui/widgets/layerDisplay/jquery.ui.layerDisplay.js`. En este archivo, entre otras cosas, se controla el evento de seleccionar una capa y agregarla a una lista de capas activas dentro del árbol de capas o quitarla cuando se deselecciona.

Dentro del archivo `jquery.ui.layerDisplay.js` existe la función **printLayerSelectedList**, encargada de mostrar la lista de capas seleccionadas en el árbol de capas, a través de esta función es posible identificar cuando una capa es activada y su identificador de capa o alias, como es definido en el archivo `/opt/map/mdm61vectormxsig.map`. Al final de la función, luego de que la capa es añadida a la lista del árbol se puede colocar un filtro con una estructura “if”, de modo que se identifique cuando una capa específica sea activada, para esto hay que comparar la variable `idLayer` que contiene el alias de la capa y utilizar JQuery para verificar el componente HTML creado con el mismo alias de la capa, este último es un contenedor de checkbox con el nombre de la capa que se agrega a la lista, y revisar el atributo “checked” para saber si la capa fue activada o desactivada. Lo anterior se muestra en la figura 54.

```
if( ( (idLayer == 'cCalor' && $('#cCalor').prop('checked')) || (idLayer == 'cmapas_calor_colima_2021' && $
('#cmapas_calor_colima_2021').prop('checked')) || (idLayer == 'cHVial_Calor_Concentrado_2019_2021' && $
('#cHVial_Calor_Concentrado_2019_2021').prop('checked')) ) ){
    console.log('Este es el layername antes: ' + idLayer );
```

Figura 54: Filtro de capas con estructura if.

En la figura 54 se aprecia que el filtro funciona siempre que alguna de las capas `cCalor`, `cmapas_calor_colima_2021` o `cHVial_Calor_Concentrado_2019_2021` sea activada.

Dentro de la estructura if, se agrega una estructura switch para agregar un título dependiendo de la capa como en la figura 55.

```
switch(idLayer ) {
    case 'cCalor':
        // code block
        layername = 'Colima 2019'
        break;
    case 'cmapas_calor_colima_2021':
        // code block
        layername = 'Colima 2021'
        break;
    case 'cHVial_Calor_Concentrado_2019_2021':
        // code block
        layername = 'Concentrado 2019 - 2021'
        break;
    default:
        // code block
}
```

Figura 55: Estructura switch para elección del título del pop-up

Posteriormente, se utiliza JQuery para cambiar la propiedad CSS del contenedor `popup_heat` y hacerlo visible. Así mismo se inyecta el HTML del contenido del contenedor. Lo anterior se puede ver en la figura 56.


```

$("#popup_heat").css("display", "block");

$("#popup_heat").html('<div class="heatpop" style="display: block;" id="innerheatlay"><span onclick="document.
getElementById(&#39;popup_heat&#39;).style.display=&#39;none&#39;" class="closesmall" title="Cerrar">x</span><h3>Leyenda Mapa
de Calor '+ layername +'</h3><div><div class="squareR"></div><div>&nbsp;Muy alta concentración </div></div><br><div
><div class="squareO"></div><div>&nbsp;Alta concentración</div></div><br><div><div class="squareY"></div><div>&nbsp;Media
concentración</div></div><br><div><div class="squareG"></div><div>&nbsp;Baja concentración</div></div><br><div
><div class="squareB"></div><div>&nbsp;Muy baja concentración</div></div></div>');
$("#popup_heat").attr('flagged','true')
  
```

Figura 56: Inyección del contenido del contenedor HTML.

Nótese que, dentro de la propia inyección de HTML se encuentra un el botón de cerrado, el cual a través del evento onClicK cambia la propiedad display del estilo del contenedor para ocultarlo.

El resultado de la función, al seleccionar una capa valida es el de la figura 57.

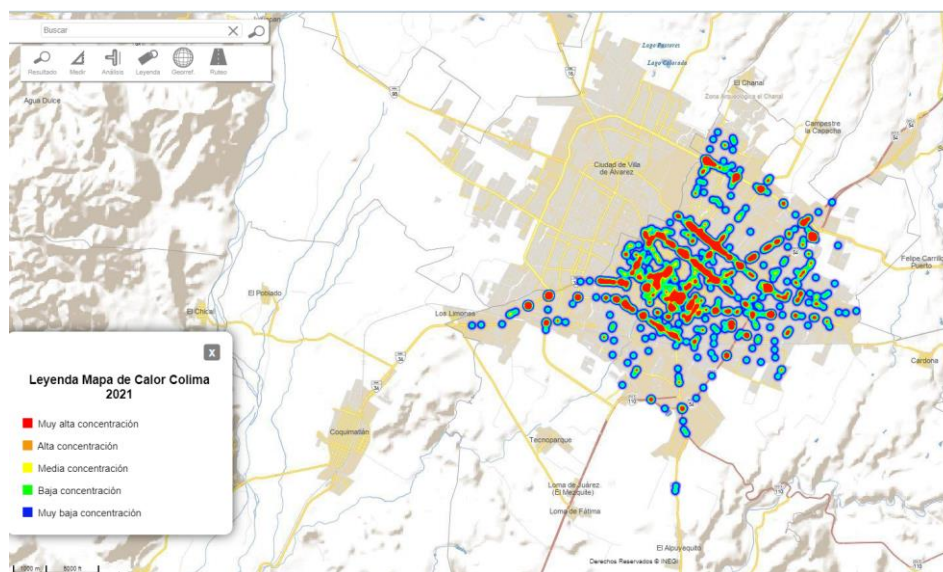


Figura 57: Ejemplo de mapa de calor con leyenda personalizada.

10.3. Estadísticas y graficas con pop-up

SIMOS tiene la capacidad de integrar un pop-up con gráficas y estadísticas que se muestra en pantalla cuando se activan ciertos temas.

Para integrar las graficas dentro de SIMOS, se utilizan las librerías Chart.js y Highcharts, cuyo código fuente está en el directorio /var/www/html/mxsig/js/frameworks. [17][18]

Desde el archivo /var/www/html/mxsig/index.html ya tienen importadas las librerías Chart.js y Highcharts. En el mismo archivo se crea un contenedor con botones para cambiar entre graficas y los sub-contenedores de las gráficas. En el siguiente fragmento de código se muestra un ejemplo de pop-up de gráficas y estadísticas de SIMOS.

```
<div class="popup_conatiner">
  <div id="item_popup">
    <div class="popup">
      <div class="close-graphs" id="close-graphs">
        <span>X</span>
      </div>
      <div >
        <span style="color: gray;
          font-size: 18px">Titulo</span>
      </div>
      <!-- Slideshow container -->
      <div class="slideshow-container">
        <!-- Full-width images with number and caption text -->
        <div class="mySlides fade" style="display: block">
          <div class="card text-center">
            <div class="grafica">
              <h1 class="grafica-title">
                Tipo de hechos viales registrados, 2019 - 2021
              </h1>
              <div>
                <canvas id="myChart_1" width="100" height="100"
></canvas>
              </div>
            </div>
          </div>
        </div>
        <div class="mySlides fade">
          <div class="grafica">
            <h1 class="grafica-title">
              Tipo de persona usuaria víctima (herida o muerta) en
              hechos viales, 2019 - 2021
            </h1>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```

        </h1>
        <div>
            <canvas id="myChart_2" width="100"
height="100"></canvas>
        </div>
        </div>
        </div>
        <div class="mySlides fade">
            <div class="grafica">
                <h1 class="grafica-title grafica-title_margin">
                    Temporalidad de hechos viales cada hora entre semana y
fin de
                    semana, 2019 - 2021
                </h1>
                <div>
                    <canvas id="myChart_3" width="200" height="00"></canvas>
                </div>
            </div>
        </div>
        <!-- The dots/circles CLUSTERS -->
        <div style="text-align: center" class="estados">
            <span class="dot" onclick="currentSlide(1)"></span>
            <span class="dot" onclick="currentSlide(2)"></span>
            <span class="dot" onclick="currentSlide(3)"></span>
        </div>
    </div>
</div>

```

En el fragmento de código anterior, el contenedor de las gráficas tiene el identificador “**item_popup**”. El elemento div que tiene el id y clase “**close-graphs**” es el utilizado para el botón de cerrar del pop-up. EL siguiente elemento div inmediato es utilizado para un título.

Cada elemento item_pop-up tiene los siguientes estilos de CSS que regulan que estén cerrados hasta que se seleccione un tema, que se muestren delante de la interfaz grafica de SIMOS, su tamaño y color.

```

#item_popup {
    /*font-size: 0rem;*/

```

```
display: none;
background-color: #fff;
margin: 0;
width: 48%;
z-index: 3;
padding: 2rem;
position: absolute;
top: 43%;
left: 50%;
transform: translate(-50%, -50%);
text-align: center;
border-radius: 2rem;
-webkit-box-shadow: 0px 0px 17px 0px rgba(0, 0, 0, 0.75);
-moz-box-shadow: 0px 0px 17px 0px rgba(0, 0, 0, 0.75);
box-shadow: 0px 0px 17px 0px rgba(0, 0, 0, 0.75);
}
```

Las graficas se presentan como slides entre las que se puede intercambiar dando click en botones al final del pop-up, para lograr esto se utiliza un elemento div con la clase “**slideshow-container**” que servirá para almacenar cada slide.

Para cada slide se utiliza un div con la clase css “**mySlides**” y la animación “**fade**” para las transiciones entre las slides. Dentro de cada slide existen diferentes elementos div para contener las gráficas, los dos principales son los que tienen la clase “grafica-title” que es donde se coloca el titulo de la grafica y la grafica en si en un elemento canvas.

El elemento HTML canvas es utilizado para dibujar gráficos, imágenes o animaciones a través de comandos, generalmente de JavaScript. En este elemento se inyectan las graficas utilizando su identificador. Por ejemplo, en el fragmento de código mostrado arriba, existe una grafica con identificador myChart_1, otra diferente con identificador myChart_2 y una tercera con identificador myChart_1.

Las gráficas se inyectan desde el mismo archivo index HTML utilizando el id del canvas. Por ejemplo, en el siguiente fragmento de código se muestra una inyección al canvas con identificador myChart_1 de una gráfica de pastel.

```
var ctx_1 = document.getElementById("myChart_1").getContext("2d");
labels_1 = [
    "(76%) Colisión con vehículo automotor",
    "(11%) Colisión con motocicleta",
    "(5%) Colisión con ciclista",
    "(5%) Colisión con objeto fijo",
    "(1%) Colisión con peatón (atropellamiento)",
    "(1%) Otro",
    "(0%) Salida del camino",
```

```

    "(0%) Volcadura",
    "(0%) Colisión con ferrocarril",
    "(0%) Colisión con animal",
    "(0%) Caída de pasajero",
];
var myChart = new Chart(ctx_1, {
  type: "pie",
  data: {
    labels: labels_1,
    datasets: [
      {
        data: [2393, 341, 153, 146, 44, 16, 15, 11, 9, 8, 2],
        backgroundColor: [
          "rgb(255, 99, 132)",
          "rgb(54, 162, 235)",
          "rgb(255, 205, 86)",
          "#D02FB9",
          "#2FD05C",
          "#3390FF",
          "#A833FF",
          "#F033FF",
          "#33FFD1",
          "#FF3333",
          "#FFE933",
        ],
      },
    ],
    hoverOffset: 4,
  },
  options: {},
  // plugins: [legendMargin],
});

```

Nótese que la referencia al canvas se almacena en la variable **“ctx_1”**. La variable **“labels”** contiene las etiquetas de la grafica y la variable **myChart** es el objeto de la gráfica. El objeto myChart recibe el canvas donde se va a mostrar, seguido del tipo de grafica que se va a representar. Después, el objeto myChart recibe un objeto con clave **“data”**, el cual a su vez se compone de lo siguiente: [19]

- Labels: Un arreglo que contiene las etiquetas de cada fragmento de la gráfica de pastel o dato.
- Datasets: Un arreglo que contiene un objeto con lo siguiente:
 - Data: Los datos que serán graficados.
 - backgroundColor: El color de fondo que tendrá cada fragmento del pastel.
 - hoverOffset: Es el desplazamiento que se presenta sobre una etiqueta cuando el mouse esta sobre un fragmento del pastel.

SIMOS contiene varios ejemplos de graficas de pastel, líneas y barras precargados en el archivo `index.html`.

En seguida, para que las gráficas se muestren al seleccionar un tema se utiliza el archivo `init.js` en el directorio `/var/www/html/mxsig/projects`. En el archivo `init.js` existe una función nombrada `contenidoTema`, la cual se ejecuta cuando se selecciona un tema y se invoca en el archivo `/var/www/html/mxsig/js/core/ui/widgets/layerManager/jquery.ui.layerManager.js`. Esta consiste en una estructura `switch` que muestra que cambia la propiedad CSS de `display` de un `pop-up` y oculta los demás, con base en el identificador de la capa, el identificador de la capa es el `alias` con el que la capa es nombrada en el archivo `/opt/map/mdm61/vectormxsig.map`.

Finalmente, al dar click en uno de los temas agregados al switch, se mostrarán las gráficas como en las figuras 58, 59 y 60.

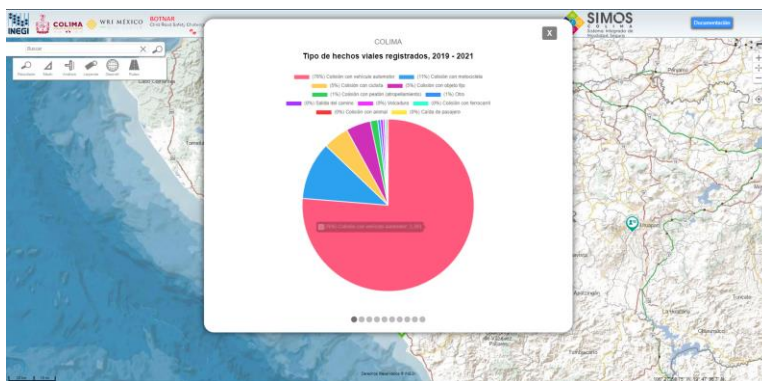


Figura 58: Grafica de pastel.

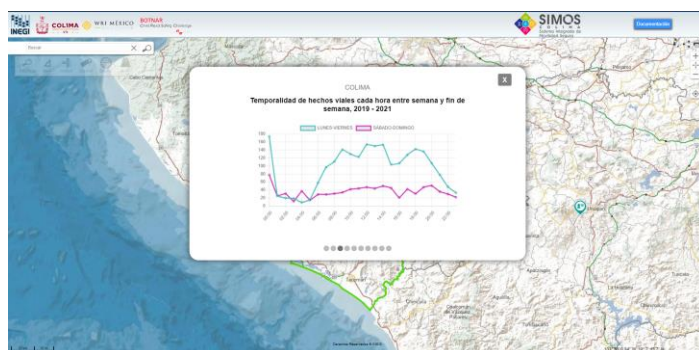


Figura 59: Grafica de líneas.



Figura 60: Grafica de barras.

10.4 Carga de capas

La carga de capas a SIMOS es un proceso que involucra herramientas externas para reproyectar las capas que se desean cargar a la proyección 900913 y luego utilizar un software compatible con postGIS para convertir archivos tipo shape (.shp) a tablas de datos y subirlas a la base de datos, como es shp2pgsql. Este procedimiento completo esta descrito en el documento "Proceso de carga de capas a SIMOS", disponible en la carpeta docs del repositorio oficial de SIMOS. [1] [20]

Referencias.

- [1] WRI México, “SIMOS” [Online]. Disponible: <https://github.com/wri-mexico/SIMOS>
- [2] INEGI, “MxSIG repositorio GitHub” [Online]. Disponible: <https://github.com/MxSIG>
- [3] Red Hat, “¿Qué es SELinux?” [Online]. Disponible: <https://www.redhat.com/es/topics/linux/what-is-selinux>
- [4] Apache, “HTTP Server Project” [Online]. Disponible: <https://httpd.apache.org/>
- [5] Fedora Project, “Extra Packages for Enterprise Linux (EPEL)” [Online]. Disponible: <https://docs.fedoraproject.org/en-US/epel/>
- [6] Red Hat Customer Portal, “8.4. Configuring Yum and Yum Repositories” [Online]. Disponible: https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/6/html/deployment_guide/sec-configuring_yum_and_yum_repositories
- [7] PHP, “¿Qué es PHP?” [Online]. Disponible: <https://www.php.net/manual/es/intro-what-is.php>
- [8] Apache Tomcat, “Apache Tomcat” [Online]. Disponible: <https://tomcat.apache.org/>
- [9] PostgreSQL, “PostgreSQL” [Online]. Disponible: <https://www.postgresql.org/>
- [10] PostgreSQL, “Repo RPMs” [Online]. Disponible: <https://yum.postgresql.org/repopackages/>
- [11] PostGIS, “PostGIS” [Online]. Disponible: <https://postgis.net/>
- [12] MapServer, “MapServer” [Online]. Disponible: <https://mapserver.org/>
- [13] pgAdmin, “pgAdmin” [Online]. Disponible: <https://www.pgadmin.org/>
- [14] IONOS, “Solr: el servidor de búsqueda de Apache” [Online]. Disponible: <https://www.ionos.mx/digitalguide/servidores/configuracion/apache-solr/>
- [15] MapServer, “Cluster” [Online]. Disponible: <https://mapserver.org/mapfile/cluster.html>
- [16] MapServer, “Kernel Density Estimation (Dynamic Heatmap)” [Online]. Disponible: <https://mapserver.org/output/kerneldensity.html>
- [17] Chart.js, “Chart.js” [Online]. Disponible: <https://www.chartjs.org/>
- [18] Highcharts, “Highcharts” [Online]. Disponible: <https://www.highcharts.com/>
- [19] Chart.js, “new Charts” [Online]. Disponible: <https://www.chartjs.org/docs/latest/developers/charts.html>
- [20] J. R. Escobar, “Proceso de carga de capas a SIMOS” [Online]. Disponible: <https://github.com/wri-mexico/SIMOS/blob/main/Docs/Proceso%20de%20carga%20de%20capas%20a%20SIMOS%20v10.pdf>