



# **Proyecto YuppGIS**

## **Manual de Usuario**

### **Autores**

Emilia Rosa

Jorge Davison

Martin Taruselli

German Schnyder

1.	Instalación .....	3
1.1.	Requisitos .....	3
1.2.	Configuración Adicional .....	3
1.2.1.	MapServer .....	3
1.3.	Instalación de YuppGIS.....	4
2.	Creación de una Aplicación .....	4
2.1.	Configuración de la aplicación .....	4
2.2.	Definición de Objetos con datos geográficos.....	5
2.3.	Generación de tablas .....	6
3.	Uso de Helpers .....	6
3.1.	Mostrar un mapa.....	6
3.2.	Mostrar filtro por Capas.....	7
3.3.	Filtro por atributos de un objeto del modelo .....	8
3.4.	Filtro por Tag .....	9
3.5.	Visualización .....	9
4.	Consultas Geográficas .....	9
5.	Manejo de KML .....	11
6.	Manejo de WKT .....	11
7.	Referencias.....	12

## 1. Instalación

### 1.1. Requisitos

Para la ejecución y uso de YuppGIS deberá de contar con:

1. Servidor Web Apache [1]
  - Versión mayor a 2.2.
  - Es necesario tener habilitado el módulo *MOD\_REWRITE* [2].
2. PHP [3]
  - Versión mayor a  $\geq 5.3.x$ .
  - Es necesario tener soporte de PHP para PostgreSQL, MySQL y SQLite.
3. PostgreSQL [4] + Postgis [5]
  - Versión de PostgreSQL mayor a 8.4.7.
  - Versión de Postgis mayor a 1.5.1.
  - Crear una base de datos con extensión de funciones geográficas.
4. MySQL [6]
  - Versión mayor a 5.1.49.
  - Crear una base de datos.

Requisitos opcionales:

1. MapServer [7]
  - Versión mayor a 5.6.

### 1.2. Configuración Adicional

#### 1.2.1. MapServer

Si se requiere poder trabajar con mapas de MapServer y Google Maps al mismo tiempo, es necesario configurar MapServer para que pueda trabajar con la proyección de Google Maps. De esta manera se puede trabajar con un mismo conjunto de datos e utilizar cualquiera de las capas base indistintamente. Para lograr dicha configuración es necesario agregar la siguiente línea (proyección EPSG) al archivo de configuración EPSG de MapServer (en Ubuntu */usr/share/proj/epsg*).

```
# spherical mercator
<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 <>
```

Esta regla define la proyección *900913* en MapServer para que pueda ser utilizada en la definición del mapa (mapfile). Para luego poder hacer re-proyecciones desde la proyección *X* a la *900913*, es necesario especificar el parámetro *wms\_srs* dentro del archivo de definición del mapa en

MapServer (mapfile), dentro de la sección WEB → METADATA, con el valor “EPSG:X EPSG:900913”. Siendo X la proyección a la que se quiere re proyectar en 900913.

### 1.3. Instalación de YuppGIS

Para instalar YuppGIS se deberán seguir los siguientes pasos:

1. Descargar última versión de la extensión YuppGIS, el cual incluye el framework Yupp.
  - <http://code.google.com/p/yuppgis/>
2. Descomprimir descargable en la carpeta **www** del servidor web Apache.
3. Abrir el navegador y dirigirse a la web en que se configuro YuppGIS
  - <http://localhost/yuppgis>  
(En caso de haber extraído el contenido del descargable y no haber cambiado el nombre de la carpeta)

Para descargarse la versión en desarrollo de YuppGIS, deberá realizar *checkout* del repositorio del proyecto <http://yuppgis.googlecode.com/svn/trunk/>.

En Linux se realiza con el siguiente comando:

```
svn checkout http://yuppgis.googlecode.com/svn/trunk/ yuppgis-read-only
```

## 2. Creación de una Aplicación

### 2.1. Configuración de la aplicación

Para crear una aplicación utilizando YuppGIS, lo primero que se debe hacer es crear una Aplicación de la forma tradicional de Yupp [8].

Luego de tener una aplicación creada, debemos editar los archivos de configuración para que la misma tenga el comportamiento deseado. Para ello, se cuenta con dos archivos de configuración: **db\_config.php** y **yuppgis\_config.php**. El primero se utiliza, al igual que en Yupp, especificando la conexión a la base de datos según el modo [9]. Pasaremos a explicar los posibles valores del segundo:

- *\$yuppgis\_mode*: se utiliza para indicar el modo de trabajo de YuppGIS (Básico o Premium).
- *\$google\_maps\_key*: Este parámetro se utiliza para especificar la llave Google Maps API a utilizar si se utiliza Google Maps.
- *\$srid*: se utiliza para especificar el SRID a utilizar por YuppGIS
- *\$wms\_url*: define la URL del servicio WMS para el caso de utilizar mapas con capa base que no sea de ‘google’.
- *\$wms\_map\_file*: ruta del archivo a utilizar por el servicio WMS para el caso de utilizar mapas con capa base que no sea de ‘google’.
- *\$wms\_layers*: propiedad que define las capas a pedir al servicio WMS para el caso de utilizar mapas con capa base que no sea de ‘google’.

- *\$wms\_format*: propiedad que define el formato de la imagen que debe retornar el servicio WMS para el caso de utilizar mapas con capa base que no sea de 'google'.

Luego dependiendo del modo trabajo se pueden definir las siguientes propiedades de configuración

#### Básico

- *\$basic\_giswsdal\_class*: En el cual se especifica el conector a utilizar para acceder al repositorio de datos geográficos.
- *\$basic\_url*: Se define la URL base para el conector por defecto de YuppGIS (RestGISWSDAL)
- *\$basic\_get\_url*: Se define la URL de obtención de elementos para conector por defecto (RestGISWSDAL)
- *\$basic\_save\_url*: Se define la URL de salvado de elementos para conector por defecto (RestGISWSDAL)
- *\$basic\_delete\_url*: Se define la URL de eliminación de elementos para conector por defecto (RestGISWSDAL)

#### Premium

- *\$gisdb*: se utiliza para definir la conexión a la base de datos geográficos.

## 2.2. Definición de Objetos con datos geográficos

Otra diferencia con Yupp a la hora de crear una aplicación, es la definición de objetos del modelo que contienen información geográfica. Para ello es necesario que nuestros objetos del modelo extiendan la clase *GISPersistentObject*, extendiendo dicha clase se está indicando que esta clase será persistente (*GISPersistentObject* extiende de *PersistentObject* [9]) y que a su vez contendrá información geográfica (atributos de tipo *Geometry*).

```
class Paciente extends GISPersistentObject {
```

Luego de indicar a YuppGIS que nuestra clase contendrá información geográfica solo falta agregar los atributos geográficos a la misma.

```
$this->addAttribute("ubicacion", GISDatatypes::POINT);
```

En esta última línea se está indicando que la clase tendrá un atributo de nombre *ubicación* y de tipo *POINT* (punto).

Los tipos de datos geográficos soportados son:

- *POINT*
- *LINESTRING*
- *LINE*

- *LINERING*
- *SURFACE*
- *POLYGON*
- *CURVE*
- *MULTIPOINT*
- *MULTICURVE*
- *MULTILINESTRING*
- *MULTISURFACE*
- *MULTIPOLYGON*
- *GEOMETRYCOLLECTION*

### 2.3. Generación de tablas

Luego de generado el modelo y configurada la aplicación, YuppGIS, extendiendo la funcionalidad de Yupp, da la opción de generar las tablas en la base de datos configurada. De esta forma el desarrollador solo necesita tener instalado uno de los manejadores soportados por YuppGIS, con su extensión para datos geográficos correspondiente.

## 3. Uso de Helpers

YuppGIS brinda una serie de Helpers (por medio de la clase *GISHelpers*) para el manejo de mapas e utilidades en interfaz de usuario, en esta sección se pasan a describir los principales. Estos Helpers podrán ser utilizados por los controladores definidos por el desarrollador siempre que estos extiendan del controlador *GISController*.

### 3.1. Mostrar un mapa

La función *Map* puede ser llamada desde una vista para generar la visualización de un mapa, se debe invocar de dicha manera:

```
<?php echo GISHelpers::Map(array(MapParams::ID=>1,
MapParams::CENTER => array("-6251096.6093197", "-
4149355.4159976"), MapParams::ZOOM => 14 )); ?>
```

Con la línea de código anterior se generara el Html correspondiente para la visualización de un mapa. En el ejemplo se pasan como parámetros, identificador del mapa a visualizar (obligatorio), centro del mapa (latitud-longitud) y el zoom.

A continuación se detallan los posibles parámetros que se pueden pasar a la función para la creación del mapa:

- *ID*: representa el identificador del mapa.
- *OpenLayerJS\_URL*: para indicar la url para obtener la imagen base del mapa en caso de usar Google Map.

- **WIDTH:** para especificar el ancho del mapa.
- **HEIGHT:** para especificar el alto del mapa.
- **BORDER:** para especificar el borde del mapa.
- **CLICK\_HANDLERS:** para asignar el manejador del evento click sobre el mapa
- **SELECT\_HANDLERS:** para asignar el manejador del evento seleccionar sobre el mapa.
- **TYPE:** para indicar el tipo del mapa (Google Map o MapServer).
- **STATE:** para conservar los parámetros que se pesaron a la vista que muestra el mapa.
- **SRID:** para asignar el SRID que se usa para el mapa.
- **CENTER:** para asignar en qué punto se encuentra el centro del mapa.
- **ZOOM:** para asignar el zoom inicial sobre el mapa.

Por más información de los parámetros posibles, ver documentación del código.



Figura 1 – Html generado por YuppGIS para mostrar un mapa.

### 3.2. Mostrar filtro por Capas

Otro helper que se proporciona es para la generación del filtro por capas de un mapa. Para poder utilizar dicho helper solo se necesita la siguiente línea de código

```
<?php echo GISHelpers::MapLayers(array(MapParams::ID => 1)); ?>
```

A la función *MapLayers*, solo se le debe indicar el identificador del Mapa del cual se quieren seleccionar las capas para filtrar.

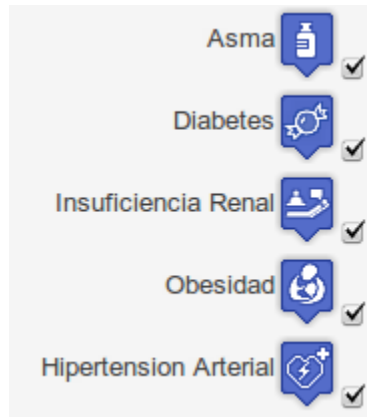


Figura 2 - Html generado por YuppGIS para mostrar el filtro por capas.

### 3.3. Filtro por atributos de un objeto del modelo

Además se brinda un helper para poder generar filtros de forma dinámica a partir de los atributos de una clase del modelo. Para esto solo hace falta agregar la siguiente línea

```
<?php echo GISHelpers::FiltersMenu('Paciente', 1); ?>
```

En este ejemplo se pasa como parámetros el nombre de la entidad (*Paciente*) e identificador del mapa. A dicha función también se le puede pasar por parámetro nombre de la método a invocar, identificador de una de las capas del mapa y si genera múltiples condiciones de filtrado o no. Para obtener más información ver la documentación del código.

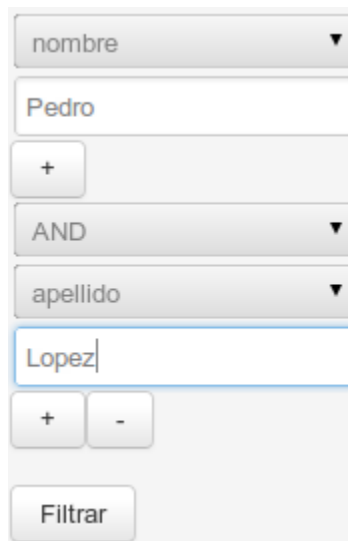


Figura 3 - Html generado por YuppGIS para mostrar el filtro por atributos.



### 3.4. Filtro por Tag

Al igual que se proporciona un Helper para filtrar por capas, se brinda un Helper para filtrar por Tags que tienen asociadas las capas de un mapa.

```
<?php echo GISHelpers::TagLayers(array(MapParams::ID => 1)); ?>
```

Simplemente con la línea anterior obtenemos dicho Helper, solo se necesita pasar por parámetro el identificador del mapa.

### 3.5. Visualización

Se proporciona un Helper que facilita el guardado y cargado de los estados de visualización de un mapa, solo es necesario pasar el identificador del mapa.

```
<?php echo GISHelpers::VisualizationState(1); ?>
```

## 4. Consultas Geográficas

Una funcionalidad destacada de YuppGIS es la generación de consultas geográficas de forma sencilla, trabajando sobre el esquema de trabajo *Premium*.

Esta funcionalidad también es una extensión de consultas que se pueden generar en Yupp [11].

```
$q = new GISQuery();  
  
$q->addProjection('p', 'ubicacion', 'ubicacion_de_p');  
  
$q->setCondition(Condition::_AND()  
    ->add(Condition::EQ('p', 'nombre', 'Juan'))  
    ->add(GISCondition::EQGEO('p', 'ubicacion', new Point(-  
56.181948, -34.884621)))  
    );  
  
$q->addFrom(Paciente::getClassName(), 'p');
```

La consulta anterior se transforma en una consulta sql, donde se consultan los pacientes que tengan como ubicación el punto -56.181948, -34.884621.

Luego de generar la consulta, se debe invocar al PersistenceManagerFactory, para que ejecute dicha consulta. Esto se realiza de la siguiente forma:

```
$pm = PersistentManagerFactory::getManager();  
  
$result = $pm->findByQuery($q);
```

También se permite ejecutar funciones sobre datos geográficos, generando *GISQuery*, dichas funciones son resueltas por el manejador de base de datos. Un ejemplo podría ser una consulta que retorna el área de un polígono, para ello se debe generar una consulta como:

```
$q = new GISQuery();  
$q->addFunction(GISFunction::AREA('m', 'zona', 'area'));  
$q->addFrom(Medico::getClassName(), 'm');
```

Condiciones geográficas disponibles:

- *CONTAINS*: Condición de pertenencia, A contiene a B.
- *ISCONTAINED*: Condición de pertenencia, B pertenece a A.
- *EQGEO*: Condición de igualdad de objetos geográficos.
- *EQGEOA*: Condición de igualdad de objetos geográficos según un alias.
- *INTERSECTS*: Condición de intersección, A intersecta a B.
- *DWITHIN*: Condición de distancia, A se encuentra a X distancia de B.

Por más información, ver documentación de YuppGIS [12].

Funciones geográficas disponibles:

- *DISTANCE*: Función de distancia, distancia de A a B.
- *DISTANCE\_TO*: Función de distancia, distancia de A a B.
- *AREA*: Función de área.
- *INTERSECTION*: Función de intersección.
- *INTERSECTION\_TO*: Función de intersección.
- *UNION*: Función de unión.
- *UNION\_TO*: Función de unionunión.
- *DIFFERENCE*: Función de diferencia.
- *DIFFERENCE\_TO*: Función de diferencia.

Por más información, ver documentación de YuppGIS [12].

## 5. Manejo de KML

YuppGIS además le brinda al desarrollador una herramienta para el manejo de KML (*KMLUtilities*), que le permite de forma fácil convertir objetos del modelo del dominio a una representación en KML.

Dicha herramienta permite generar el KML para una capa de un determinado mapa, el KML resultante contendrá la información de todos los elementos de una capa. O generar el KML a partir de un elemento particular, un objeto del modelo que tenga un atributo geográfico.

A su vez también permite generar una geometría (*Geometry*) a partir de un KML que la represente. Las funciones que brinda esta clase son:

- *toKML*
- *fromKML*

## 6. Manejo de WKT

Al igual que para el manejo de KML, YuppGIS ofrece una herramienta que permite al desarrollador transformar objetos geográficos (*Geometry*) a su representación en WKT y viceversa. Dichas funcionalidades se encuentra en la clase *WKTGEO*.

Las funciones que brinda esta clase son:

- *toText*
- *fromText*

## 7. Referencias

1. *Servidor web Apache*  
<http://httpd.apache.org/>
2. *Módulo Apache MOD\_REWRITE*  
[http://httpd.apache.org/docs/current/mod/mod\\_rewrite.html](http://httpd.apache.org/docs/current/mod/mod_rewrite.html)
3. *PHP*  
<http://www.php.net/>
4. *PostgreSQL*  
<http://www.postgresql.org/>
5. *Postgis*  
<http://postgis.refractory.net/>
6. *MySQL*  
<http://www.mysql.com/>
7. *MapServer*  
<http://www.mapserver.org/>
8. *Creando una aplicacion con Yupp*  
[http://www.simplewebportal.net/yupp\\_framework\\_php\\_doc/creando\\_una\\_aplicacion.html](http://www.simplewebportal.net/yupp_framework_php_doc/creando_una_aplicacion.html)
9. *Yupp FAQ*  
<http://code.google.com/p/yupp/wiki/FAQ>
10. *Yupp Modelo*  
[http://www.simplewebportal.net/yupp\\_framework\\_php\\_doc/2\\_modelo.html](http://www.simplewebportal.net/yupp_framework_php_doc/2_modelo.html)
11. *Yupp Criteria*  
[http://www.simplewebportal.net/yupp\\_framework\\_php\\_doc/2\\_2\\_criteria.html](http://www.simplewebportal.net/yupp_framework_php_doc/2_2_criteria.html)
12. *Documentación YuppGIS*  
<http://yuppgis.googlecode.com/svn/yuppgis-apidoc/index.html>