Taller de MapProxy

Release 2.0 siglibre VIII

Geoinquietos Valencia

March 30, 2014

Contents

CHAPTER 1

Introducción

1.1 Qué es MapProxy

MapProxy es un servidor de teselas que lee datos de servidores WMS, TMS, configuraciones de Mapserver o Mapnik, Google Maps, Bing Maps, etc. Podría decirse que MapProxy es un *acelerador* de mapas en Internet, aunque no solo ofrece servicios de *proxy*, también es un Servidor WMS, permite realizar *Sembrado (Seeding)* de capas, permite gestionar seguridad de acceso a capas, reproyectarlas, etc.

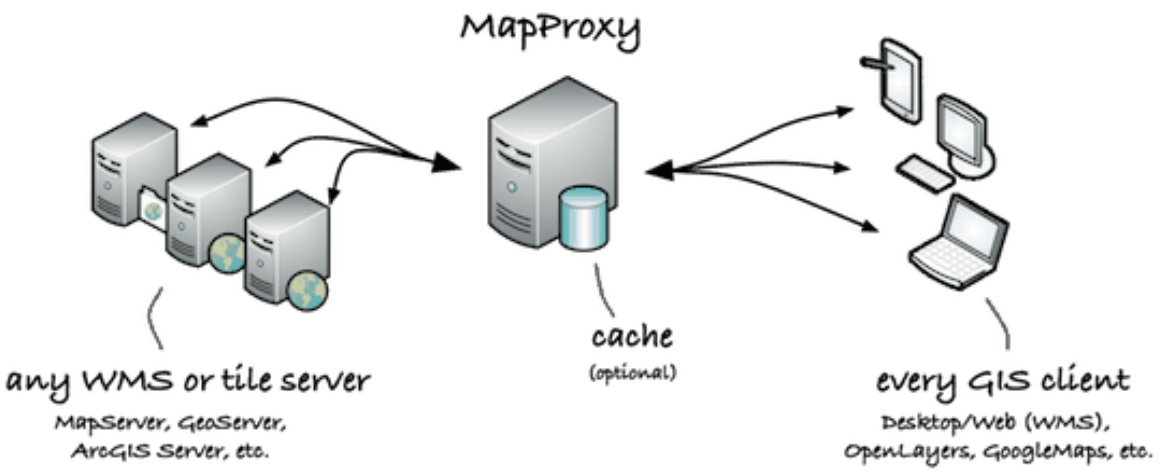


Figure 1.1: Esquema de una red con MapProxy configurado

1.2 Un poco más sobre MapProxy

- La web del proyecto es http://mapproxy.org
- Es un producto de Omniscale (desarrolladores también de ImpOSM)
- Oliver Tonnhofer es su desarrollador principal
- Está escrito en Python
- Es FOSS desde 2010 (licencia Apache)
- Tiene una lista de correo para soporte y dudas (en inglés)

1.3 Pero ¿para qué sirve?

Algunos casos de uso:

- Ofrecer acceso a servicios de mapas en zonas con acceso restringido a Internet
- Ofrecer a Internet ciertos servicios internos de una organización sin abrir todo el servidor de mapas corporativo
- Generar servicios de teselas (TMS/WMTS) a partir de un servidor WMS
- Acelerar el acceso a servicios de mapas *cacheando* la información
- Mezclar cartografía de diferentes servicios de mapas
- Descargar cartografía a equipos que se van a desplazar a zonas sin acceso a Internet (caso del equipo HOT de OSM)
- Servir cartografía diseñada con TileMill
- Ofrecer servicios en diferentes sistemas de coordenadas a partir de un servicio TMS que solo nos llega en el sistema Mercator.

1.4 ¿Cómo funciona?

Se trata de un software de servidor que se configura a través de ficheros escritos en YAML y *scripts* Python. Una vez correctamente configurado se *despliega* el servicio mediante alguno de los procedimientos para aplicaciones *Python* que siguen el estándar WSGI.

```
services:
  demo:
  kml:
  tms:
  wmts:
  wms:
    srs: ['EPSG:3857', 'EPSG:900913', 'EPSG:4258', 'EPSG:4326', 'EPSG:25831']
    image_formats: ['image/jpeg', 'image/png']
    md:
      # metadata used in capabilities documents
      title: Taller MapProxy
      abstract: Ejercicio de aceleración de WMS y OSM con MapProxy
      online_resource: http://localhost:8080/service
      contact:
        person: Pedro-Juan Ferrer, Iván Sánchez y Jorge Sanz
        position: Facilitadores
        organization: Geoinquietos Valencia
        email: pferrer@osgeo.org , jsanz@osgeo.org y ivan@sanchezortega.es
      access_constraints:
        Este servicio tiene únicamente objetivos educativos.
      fees: 'None'
```

CHAPTER 2

Instalación de MapProxy

Note: El siguiente proceso de instalación está orientado a una máquina GNU/Linux de tipo Debian/Ubuntu o similar. En esta entrada del geomaticblog puede encontrarse unas notas de instalación en *Windows*.

2.1 Descarga de dependencias del sistema

Vamos a instalar un par de paquetes iniciales, para ello abre un *emulador de terminal* y ejecuta::

```
$ sudo apt-get update
$ sudo apt-get install tree python-virtualenv
```

Note: Las líneas de esta documentación que comiencen con el símbolo del dólar indican instrucciones a ejecutar en una consola del sistema. Si vas a copiar estas líneas en tu consola debes hacerlo **sin incluir el dólar**.

Note: En el *OSGeo Live* puedes acceder al emulador de terminal desde el menú *Applications* → *Accessories* → *Terminal Emulator*

A continuación puedes instalar el resto de dependencias de MapProxy:

```
$ sudo apt-get install python-imaging \
  python-yaml libproj0 libgeos-dev python-lxml libgdal-dev \
  python-shapely build-essential python-dev libjpeg-dev \
  zlib1g-dev libfreetype6-dev
```

Esto descargará unas 200MB en binarios en un sistema nuevo, tardará un buen rato... A partir de aquí todo se ejecuta como un usuario normal. En el caso de *OSGeo Live* muchos de estos paquetes ya están instalados y por tanto solo instalará los necesarios.

2.2 Cómo instalar MapProxy

Primero vamos a descargar los materiales del taller. Usando la misma terminal de la sección anterior, si no te has cambiado estarás en la carpeta raíz del usuario. En el caso de *OSGeo Live* esta carpeta es `/home/user`. Encontrándonos en esta carpeta ejecutar:

```
$ mkdir mapproxy-workshop
$ wget -O mapproxy-workshop/mapproxy-workshop.pdf "http://bit.ly/mapproxy-workshop-gviii
```

Note: En una terminal Linux puedes volver en cualquier momento a la carpeta raíz del usuario utilizando el comando **cd** sin parámetros.

Con esto tendremos una nueva carpeta `mapproxy-workshop` con el documento pdf del taller. En *OSGeo Live* podemos abrir el archivo pdf usando el *software* disponible en *Applications* → *Office* → *Document Viewer*.

Moverse a la carpeta creada y crear el entorno virtual con:

```
$ cd mapproxy-workshop
$ virtualenv venv
```

Activar el entorno virtual con:

```
$ source venv/bin/activate
```

Note: Una vez activado el entorno virtual nos aparecerá entre paréntesis en el símbolo del sistema el nombre del mismo. Se indica igualmente en estas instrucciones para recordarlo.

Instalar la librería de tratamiento de imágenes Pillow con:

```
(venv)$ pip install Pillow
```

Y ya por fin podemos instalar MapProxy:

```
(venv)$ pip install MapProxy
```

Al finalizar podremos comprobar que MapProxy está instalado usando la instrucción `mapproxy-util`:

```
(venv)$ mapproxy-util --version
MapProxy 1.6.0
```

2.3 Crear un proyecto de demostración

Para comprobar que MapProxy está funcionando correctamente vamos a crear un proyecto de ejemplo y lo arrancaremos con el servidor de pruebas que MapProxy incorpora. Para ello, nos colocaremos en la carpeta raíz del taller y crearemos la carpeta `confs`. Nos movemos a esa carpeta y ejecutamos la herramienta que MapProxy incorpora para diferentes tareas **mapproxy-util**:

```
(venv)$ mkdir confs
(venv)$ cd confs
(venv)$ mapproxy-util create -t base-config test
```

Y veremos aparecer en pantalla la confirmación de que ha escrito los archivos:

```
writing test/mapproxy.yaml
writing test/seed.yaml
writing test/full_example.yaml
writing test/full_seed_example.yaml
```

Esta instrucción ha creado la carpeta `test` y dentro de ella cuatro ficheros de configuración dos de los cuales veremos en la siguiente parte del taller. El fichero `mapproxy.yaml` configura el servidor de teselas y `seed.yaml` las tareas de pregeneración y/o limpieza de teselas. Los otros dos son ejemplos que muestran configuraciones de casi todos los parámetros que admite MapProxy; no están pensados para ejecutarse (de hecho tienen configuraciones incompatibles) están pensados para demostrar opciones de configuración.

Para ejecutar el servidor de pruebas se utilizará de nuevo **mapproxy-util** esta vez con la tarea de arrancar el servidor de pruebas.:

```
(venv)$ cd test
(venv)$ mapproxy-util serve-develop mapproxy.yaml
```

Y veremos aparecer en pantalla líneas similares a las siguientes:

```
[2014-02-25 22:20:01,823] mapproxy.config - INFO - reading: /home/user/mapproxy-workshop
[info]  * Running on http://127.0.0.1:8080/
[info]  * Restarting with reloader: stat() polling
[2014-02-25 22:20:01,997] mapproxy.config - INFO - reading: /home/user/mapproxy-workshop
```

Si nos dirigimos con nuestro navegador a la dirección web http://localhost:8080 podremos ver un mensaje de bienvenida y si hacemos clic en el enlace *demo* MapProxy nos mostrará su interfaz de demostración de servicios. En esta página podemos ver diferentes enlaces a ficheros de capacidades y a visores. Podemos probar con el servicio **TMS** y ver la capa `osm` en el sistema de coordenadas `EPSG:3857` en formato `png`.

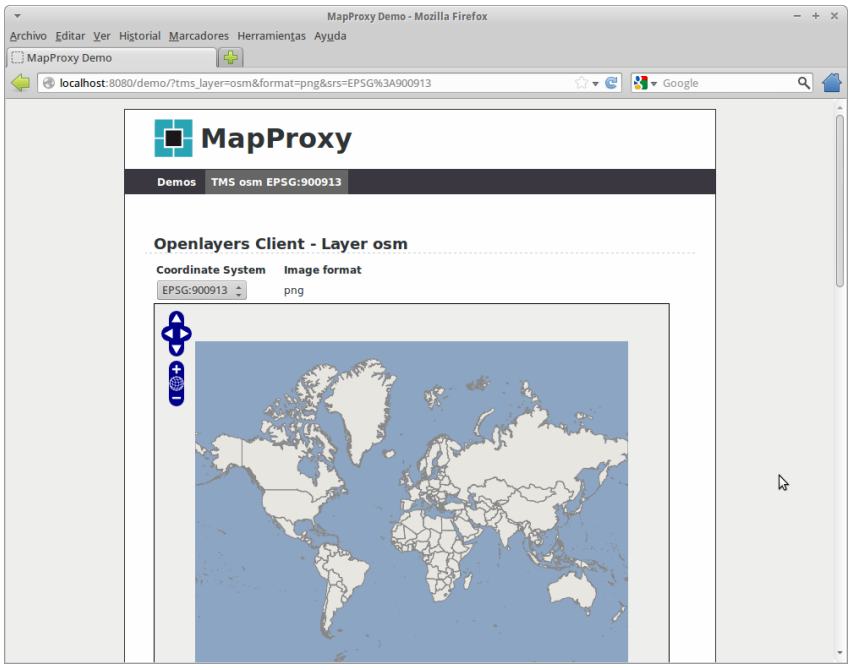


Figure 2.1: Interfaz de demostración de MapProxy

Esta interfaz además de permitir navegar por la cartografía, ofrece información adicional sobre la *cache* como las coordenadas de sus límites, los niveles de resolución así como el código mínimo necesario para cargar dicha capa usando la biblioteca de *webmapping* OpenLayers.

Note: Para apagar el servidor de pruebas se debe pulsar la combinación de teclas `Control+C`.

Si se observa cuidadosamente la salida de **mapproxy-util**, se pueden tanto las peticiones que mapproxy hace al *source*...:

```
[2014-02-25 22:20:13,844] mapproxy.source.request - INFO - GET http://osm.omniscale.net/
```

...como las peticiones que MapProxy *responde* al cliente:

```
[info] 127.0.0.1 - - [25/Feb/2014 22:20:13] "GET /tms/1.0.0/osm/webmercator/0/1/1.png HT
```

Finalmente, podemos comprobar cómo el servidor ha guardado algunas teselas al visitar la demostración en la carpeta `confs/test/cache_data` que podemos ver desde la consola si navegamos hasta esa carpeta y ejecutamos el comando **tree**:

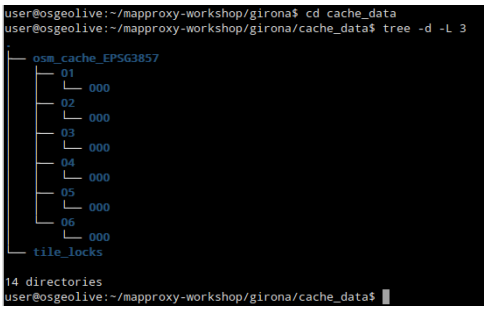


Figure 2.2: Presentando la estructura de carpetas de la *cache*

Como vemos MapProxy ha creado una carpeta para la *cache* de la capa `osm` y una estructura de carpetas donde se almacenan las imágenes.

Attention: ¿Qué tamaño tienen las imágenes? ¿En qué formato están? Si tenemos *imagemagick* instalado en nuestro ordenador, podemos ver información sobre las imágenes del caché rápidamente ejecutando **identify 'find . | grep png'**

2.4 Despliegue

No es objetivo de este taller describir el proceso de despliegue de MapProxy en un servidor de producción. MapProxy es una aplicación escrita en Python que sigue el estándar WSGI de publicación de aplicaciones *web*. Este estándar permite publicar aplicaciones de diferentes formas que dependerán en parte de nuestro entorno. En la documentación de despliegue de MapProxy se detallan las más importantes entre las que se podrían destacar:

- Mediante Apache + mod_WSGI: en esta alternativa se activa el módulo WSGI de Apache y se prepara una sección en la configuración que apunte a la ubicación de nuestro *server script*. Esta variante funciona tanto en Windows como en servidores GNU/Linux.
- Mediante Gunicorn: en este caso se configura un servicio que arranca un servidor gunicorn que se podrá a continuación exponer directamente u ofrecer a través de un proxy inverso con otro servidor web Apache, Nginx o cualquier otro. Esta variante solo se puede configurar en máquinas GNU/Linux.

En ambos casos se utiliza un *script* de arranque de la aplicación WSGI que se puede generar con la herramienta **mapproxy-util**.

CHAPTER 3

El archivo de configuración `mapproxy.yaml`

3.1 Introducción

Las diferentes funcionalidades de MapProxy se configuran a través de archivos *YAML*, un estandar de serialización de datos que se emplea en diversos lenguajes de programación.

MapProxy se configura a través de los archivos `mapproxy.yaml` y `seed.yaml`, definiendo para cada archivo una serie de secciones y de directivas en las secciones. Estos nombres de fichero son solo una propuesta, desde luego se pueden elegir otros que se adecuen a nuestro proyecto.

En la presente sección hablaremos solo del archivo principal de configuración `mapproxy.yaml`. Dejaremos el archivo `seed.yaml` para la sección *El archivo de configuración seed.yaml*.

Es muy importante respetar la indentación en los archivos, y esta debe realizarse con **espacios** y **NUNCA** con tabuladores.

Para seguir el taller crearemos un proyecto llamado *girona01* y editaremos el contenido de su archivo `mapproxy.yaml`:

```
$ cd /home/user/mapproxy-workshop/
$ mapproxy-util create -t base-config girona01
$ cd girona01
$ leafpad mapproxy.yaml &
```

Note: **leafpad** es un editor de texto disponible en *OSGeo Live*. Puedes usar cualquier editor de ficheros de texto plano para trabajar con ficheros `.yaml` siempre y cuando se respete el uso de espacios y la codificación de caracteres.

3.2 `mapproxy.yaml`

El archivo está compuesto de las siguientes secciones

services: Definición de los servicios que se van a ofrecer.

layers: Definición de las capas que se servirán. Cada capa puede estar constituida por varias *sources* y *caches*

caches: En esta sección se configuran las cachés internas de los servicios.

sources: Definición de los orígenes de datos de los servicios.

grids: En esta sección se definen las rejillas sobre las que se alinean las imágenes que genera MapProxy.

globals: En esta sección generalmente se definen parámetros que son comunes a todas las secciones.

El orden en el que aparecen las secciones **no** es importante.

El archivo puede subdividirse en varios archivos utilizando la directiva **base** (documentación).

3.3 Relación entre los componentes

Para tener una idea global de cómo interrelacionan los distintos componentes de MapProxy podemos consultar el mapa conceptual de la figura *Mapa conceptual de interrelacion entre los componentes de MapProxy*.

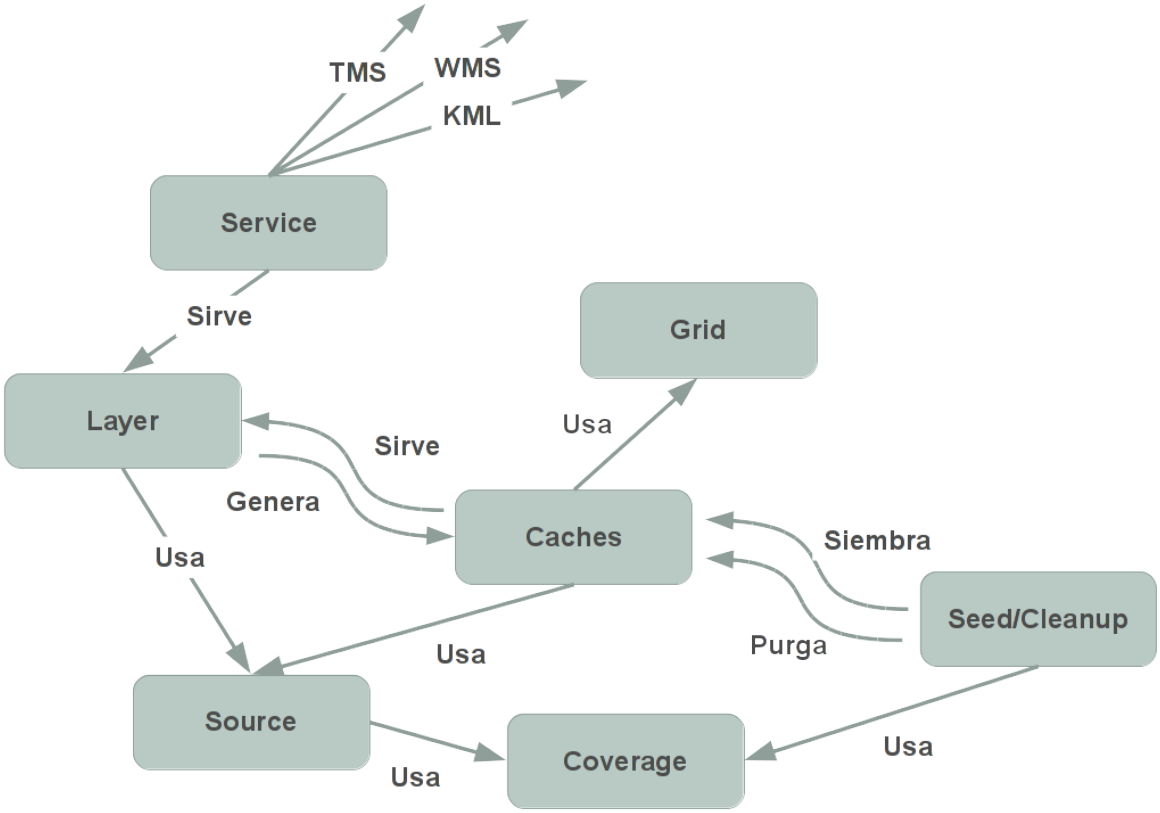


Figure 3.1: Mapa conceptual de interrelacion entre los componentes de MapProxy

3.4 services

MapProxy puede generar los siguientes tipos de servicio:

- Web Map Service (OGC WMS) y WMS-C [**wms**]
- Tiled Map Services (TMS) [**tms**]
- Keyhole Markup Language (OGC KML) [**kml**]

- Web Map Tile Services (WMTS) [**wmts**]
- MapProxy Demo Service [**demo**]

Para cada uno se emplea su propia clave, que aparece listada entre corchetes, y en algunos casos se pueden configurar opciones adicionales.

Para el presente taller utilizaremos el servicio *wms* que se configura indicando los sistemas de referencia en los que se va a servir (**srs**), los formatos de imagen (**image_formats**) y metadatos adicionales (**md**):

Remplaza el contenido de la sección *services* por el contenido que hay a continuación:

```
services:
  wms:
    srs: ['EPSG:3857', 'EPSG:900913', 'EPSG:4258', 'EPSG:4326', 'EPSG:25831']
    image_formats: ['image/jpeg', 'image/png']
    md:
      # metadata used in capabilities documents
      title: Taller MapProxy
      abstract: Ejercicio de aceleración de WMS y OSM con MapProxy
      online_resource: http://localhost:8080/service
      contact:
        person: Pedro-Juan Ferrer, Iván Sánchez y Jorge Sanz
        position: Facilitadores
        organization: Geoinquietos Valencia
        email: pferrer@osgeo.org , jsanz@osgeo.org y ivan@sanchezortega.es
      access_constraints:
        Este servicio tiene únicamente objetivos educativos.
      fees: 'None'
```

Puede encontrarse una descripción más completa de las claves y opciones de los servicios en la página de documentación de services de MapProxy

3.5 layers

Las capas definen la información que MapProxy proporciona y están formadas por una lista (una lista de *YAML*) de pares clave - valor.

La información mínima que se requiere es el nombre (**name**) como identificador único, el título (**title**) como pequeña descripción y el origen u orígenes de datos (del propio archivo de MapProxy) que la conforman (**source**):

Remplaza el contenido de la sección *layers* por el contenido que hay a continuación:

```
layers:
  - name: orto5m-icc-proxy
    title: Ortofoto 1:5000 del ICC de la zona de Girona
    sources: [icc_cache]
```

Puede encontrarse más información sobre las capas así como otros parámetros configurables de las mismas en la sección de layers de la página de configuración de la documentación de MapProxy

3.6 caches

En *caches* se configura la manera en la que se almacena una copia de la información en disco, para no tenerla que volver a pedir al servidor. La información que hay que proporcionar en este caso es el origen de datos (**sources**) y el grid o grids (**grids**) sobre los que queremos guardar los cachés. En caso de haber varios grids se creará una caché separada por cada capa y cada *grid*

Remplaza el contenido de la sección *caches* por el contenido que hay a continuación:

```
caches:
  icc_cache:
    grids: [utm_girona]
    sources: [icc_wms]
```

Puede encontrarse más información sobre las caches así como otros parámetros configurables de los mismos en la sección de caches de la página de configuración de la documentación de MapProxy

3.7 sources

En esta sección se definen los diferentes orígenes de datos de los servicios que ofrece el archivo de MapProxy, se define el nombre del origen de datos y se configuran parámetros del mismo como el tipo (**type**) del que admite *wms*, *tiles*, *mapserver*, *mapnik* y *debug*. Cada tipo tiene sus propias configuraciones.

Remplaza el contenido de la sección *sources* por el contenido que hay a continuación:

```
sources:
  icc_wms:
    type: wms
    req:
      url: http://shagrat.icc.es/lizardtech/iserv/ows
      layers: orto5m
    supported_srs: ['EPSG:4326', 'EPSG:25831']
    coverage:
      bbox: [2.67,41.88,2.97,42.07]
      bbox_srs: 'EPSG:4326'
```

Puede encontrarse una descripción más completa de las claves de cada tipo en la página de sources de la documentación de MapProxy

3.8 grids

La sección de grids define las rejillas que emplea MapProxy a nivel interno para almacenar las imágenes generadas. Hay varias opciones de configuración, muchas pueden emplearse simultáneamente aunque tengan efectos contradictorios y produzcan resultados ambiguos.

En general lo mínimo a definir *debería* ser el nombre, el sistema de referencia (**srs**), el *bounding box* (**bbox**) y las resoluciones (**min_res** y **max_res**) aunque en los grids que están basados en otros grids la lista de parámetros puede ser menor.

Remplaza el contenido de la sección *grids* por el contenido que hay a continuación:

```
grids:
  utm_girona:
    srs: 'EPSG:25831'
```

```
    bbox: [2.67,41.88,2.97,42.07]
    bbox_srs: 'EPSG:4326'
    min_res: 2000
    max_res: .5
```

Attention: La resolución se mide en unidades del SRS por pixel. Como estamos usando EPSG:25831, que es una proyección UTM, podemos suponer que la resolución mínima es de 2000 metros/pixel y la máxima de 50 cm/pixel.

Se puede consultar más información sobre las claves en la sección de grids de la página de configuración de la documentación de MapProxy

3.9 globals

En esta sección se colocan directivas y claves que son comunes a todas las otras secciones o son internas de MapProxy.

```
globals:
  cache:
    base_dir: 'cache_data'
    lock_dir: 'cache_data/locks'

  image:
      resampling_method: bilinear
      jpeg_quality: 90
```

Attention: Si el directorio de caché no empieza por una barra "/", se supone que es un directorio *relativo* a donde se encuentre el fichero `mapproxy.yaml`.

Una vez más hay amplia información sobre las claves y directivas en la sección de globals de la página de configuración de la documentación de MapProxy

CHAPTER 4

El archivo de configuración `seed.yaml`

4.1 Introducción

MapProxy genera teselas bajo demanda y las puede almacenar en una cache, pero para acelerar el proceso, sobretodo de capas que no se prevea que vayan a cambiar demasiado, se puede *sembrar* la caché para tener imágenes pregeneradas.

El proceso de sembrado o *seeding* se puede lanzar a través de una herramienta de consola llamada **mapproxy-seed** y configurarse fácilmente a través de un script en *YAML* llamado *seed.yaml*

Para seguir el taller seguiremos empleando el proyecto llamado girona01 creado en la sección *El archivo de configuración mapproxy.yaml*. Editaremos el contenido del archivo `seed.yaml`:

```
$ cd /home/user/mapproxy-workshop/girona01
$ leafpad seed.yaml &
```

4.2 `seed.yaml`

El archivo consta de las siguientes secciones

seeds En esta sección se configuran las opciones de *sembrado* de las capas.

cleanups En esta sección se configuran las purgas del sembrado para liberar espacio en disco eliminando imágenes viejas.

coverages En esta sección se definen zonas que después se pueden emplear tanto en el sembrado como en las purgas.

4.3 seeds

En la sección se define **qué** debe ser sembrado haciendo referencia tanto a las caches (**caches**), como a las rejillas (**gids**) y por supuesto a los niveles de zoom (**levels**) pudiendo emplearse además claves de zonas (**coverages**).

Remplaza el contenido de la sección *seeds* por el contenido que hay a continuación:

```
seeds:
  girona_icc:
    caches: [icc_cache]
    grids: [utm_girona]
```

```
    levels:
      from: 1
      to: 7
    coverages: [girona]
```

Puede encontrarse más información sobre estas y otras claves de la sección en la correspondiente sección sobre seeds de la página de seeding de la documentación de MapProxy

4.4 cleanups

La sección permite configurar las purgas de las cachés para evitar que se acumulen imágenes viejas en disco.

Se debe dar un nombre a cada configuración de purga y definir a que cachés van a atacar (**caches**), en qué rejillas (**grids**), a qué niveles (**levels**) o en que coberturas (**coverages**) y por supuesto la resolución temporal de la purgas (**remove_before**).

Remplaza el contenido de la sección *cleanups* por el contenido que hay a continuación:

```
cleanups:
  girona:
    caches: [icc_cache]
    grids: [GLOBAL_MERCATOR, GLOBAL_GEODETIC, utm_girona]
    levels:
      from: 8
    coverages: [girona]
    remove_before:
      weeks: 1
      days: 2
      hours: 3
      minutes: 4
```

Puede encontrarse más información sobre estas y otras claves de la sección en la correspondiente sección sobre cleanups de la página de seeding de la documentación de MapProxy

4.5 coverages

Por último, el archivo permite la definición de zonas en las que aplicar la tanto el sembrado como las purgas.

Estas zonas pueden definirse tanto como un *bounding box* o como una región definida con *WKT* en un archivo de texto o a través de un polígono que pueda leerse empleando OGR.

Añade el contenido hay a continuación a la sección *coverages*:

```
coverages:
  girona:
    bbox: [2.67,41.88,2.97,42.07]
    bbox_srs: "EPSG:4326"
```

Se pueden encontrar algunos ejemplos de configuración en la correspondiente sección sobre coverages de la página de seeding de la documentación de MapProxy

CHAPTER 5

Ejercicios

5.1 Ejercicio: acelerar el acceso a un WMS

Te sugerimos que para resolver los ejercicios inicies un proyecto nuevo llamado *ej01*:

```
$ cd /home/user/mapproxy-workshop/
$ mapproxy-util create -t base-config ej01
$ cd ej01
$ leafpad mapproxy.yaml &
```

y borres el contenido del archivo usando la combinación de teclas `Control+A` y después la tecla `Supr`.

5.1.1 Primera parte: acceder a un servicio de ortoimágenes

Supongamos que trabajamos en una oficina con un acceso restringido a Internet. Vamos a crear un *proxy* a la capa `orto5m` ofrecida por el Institut Cartogràfic Català en su servicio de ortofotos y mapas *raster* http://shagrat.icc.es/lizardtech/iserv/ows. En concreto vamos a trabajar sobre la zona de la ciudad de Girona y alrededores con las siguientes coordenadas de rectángulo máximo:

- Longitud mínima 2.67
- Latitud mínima: 41.88
- Longitud máxima: 2.97
- Latitud máxima: 42.07

Te recordamos que para lanzar un servidor debes usar la orden:

```
$ mapproxy-util serve-develop mapproxy.yaml
```

y para pararlo se debe pulsar la combinación de teclas `Control+C`.

5.1.2 Segunda parte: cachear un servicio de ortoimágenes

En nuestra oficina hay un cierto número de técnicos que necesitan acceder a diario a un servicio de ortoimágenes por WMS. Sería muy conveniente que pudiéramos almacenar una *cache* de dicho servicio para que el acceso a esta información fuera más rápida y eficiente, ahorrando además una considerable cantidad de ancho de banda a nuestra organización (y procesamiento al ICC).

Trabajaremos con el mismo servidor, capa y extensión del ejercicio anterior por lo que el *service* configurado nos servirá sin hacer cambios.

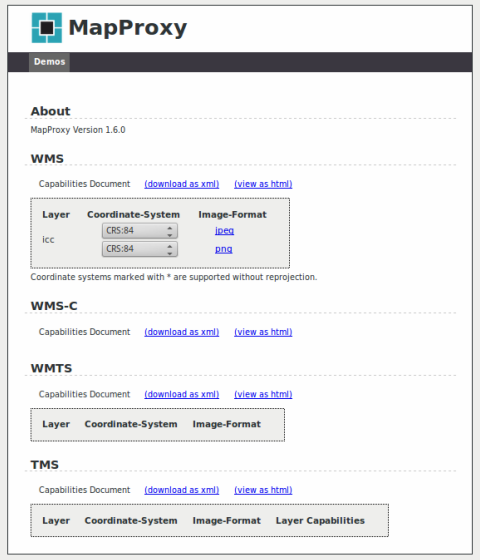


Figure 5.1: Servicio de demostración del proxy de nuestro WMS

El ejercicio por tanto consiste en crear una configuración de MapProxy que ofrezca una capa que almacene *caches* en los sistemas `EPSG:900913` y `EPSG:4326` de esta capa del servicio WMS del ICC para la zona delimitada. El servidor WMS debe ofrecer además de estos dos sistemas de referencia, también en el más estándar `EPSG:3857` y también en UTM31N, es decir en `EPSG:25831`.

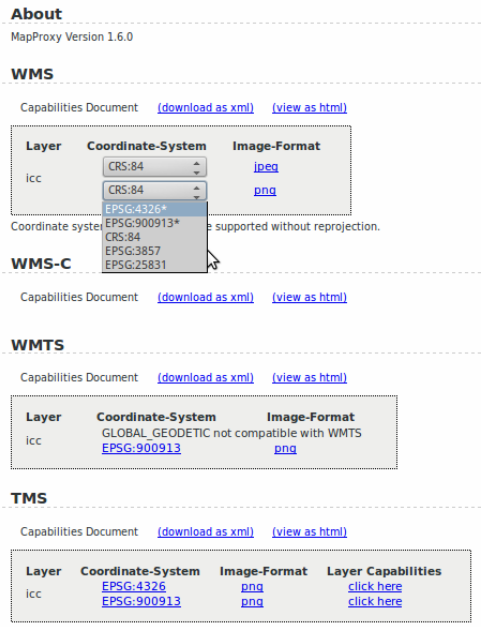


Figure 5.2: Servicio de demostración de nuestra capa cacheada

Tip: Puedes ver cómo se configuran los *grids* en el apartado correspondiente de la documentación de MapProxy.

Tip: Resulta conveniente definir en el origen los dos sistemas de coordenadas soportados por el servidor WMS `EPSG:4326` y `EPSG:2581` (documentación).

Attention: Con esta configuración recomendada, ¿qué *cache* se rellenará al pedir teselas en el sistema `EPSG:900913`? ¿Sabrías decir por qué?

Como nuestros técnicos usan a menudo cartografía en coordenadas UTM, sería interesante que crearas una *cache* expresamente para ese sistema de coordenadas, de forma que MapProxy no tenga que reproyectar las teselas todo el tiempo.

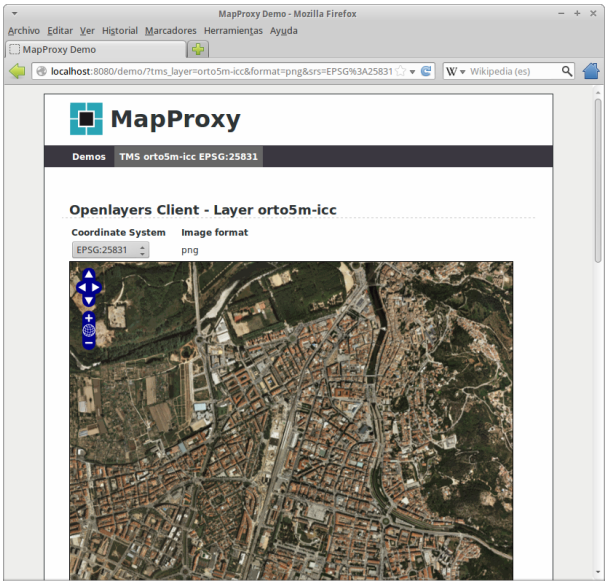


Figure 5.3: TMS de la ortofoto del ICC

Note: Por defecto las cachés hemos visto que se almacenan en formato `png`. Esta *cache* es de una ortofoto por lo que parece más adecuado utilizar el formato `jpeg` para almacenar y transmitir nuestras teselas. ¿Cómo configuramos MapProxy para que nuestra *cache* se almacene en este formato?

5.1.3 Tercera parte: cachear las teselas de OpenStreetMap

OpenStreetMap es la mayor base de datos de información geográfica generada por la comunidad. Este proyecto proporciona teselas que podemos utilizar en nuestros proyectos, siempre que sigamos su licencia.

El ejercicio consiste en añadir a nuestro servicio para la zona de Girona una nueva capa con las teselas de OSM. Para ello definiremos una nueva capa, un nuevo servicio, una nueva *cache* y un nuevo *grid* de acuerdo a las especificaciones de OSM. Podemos usar como base la configuración que ofrece el proyecto en su wiki pero hay que trabajar un poco más para conseguir que nuestra capa se centre en la ciudad de Girona.

5.2 Ejercicio: *seeding* y borrado de *caches*

Te sugerimos que para resolver los ejercicios inicies un proyecto nuevo llamado *ej02*:

```
$ cd /home/user/mapproxy-workshop/
$ mapproxy-util create -t base-config ej02
```

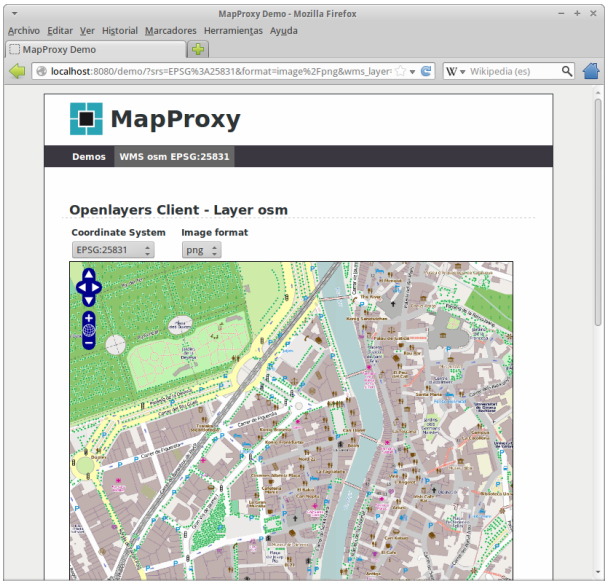


Figure 5.4: WMS de OpenStreetMap servido en UTM 31N

```
$ cd ej02
$ leafpad seed.yaml &
```

y borres el contenido del archivo usando la combinación de teclas `Control+A` y después la tecla `Supr`.

Attention: Como hemos creado un proyecto nuevo el archivo `mapproxy.yaml` será el generado por defecto. Pero para este ejercicio deberás emplear un archivo con la configuración trabajada en el *Ejercicio: acelerar el acceso a un WMS*.

Note: Puedes copiar el archivo `mapproxy.yaml` del *Ejercicio: acelerar el acceso a un WMS* o descargarlo de aquí.

5.2.1 Sembrar una *cache*

Llamamos **sembrar** una *cache* a generar toda la *cache* de antemano. Hay un par de casos de uso típicos para los que es adecuado sembrar:

- Usar cartografía en portátiles sin una conexión fiable a Internet (en campo, en el extranjero, o en una demo)
- Acelerar el acceso a las capas cacheadas, descargando todo (por ejemplo) la noche anterior

En este ejercicio vamos a sembrar los datos de OSM en el área de Girona, pero sólo para unos cuantos niveles de *zoom*. Una vez hecho el sembrado, veremos cómo MapProxy sirve las imágenes sin necesidad de pedirlas al origen.

5.2.2 Sembrado sencillo

La tarea más sencilla es lanzar **una** tarea de sembrado un cache en **una** cobertura (área) para **algunos** niveles de *zoom*. La *cache* (con sus correspondientes *capas* y *orígenes*) deberían estar ya definidos en

vuestros `mapproxy.yaml`. Las tareas de sembrado y las coberturas se definen en un fichero aparte, normalmente nombrado `seed.yaml`.

Hay que recordar que la *cache* es siempre una pirámide de imágenes, y que su extensión y niveles de *zoom* vienen referidos por el *grid* del fichero `mapproxy.yaml`. Por eso, cuando se siembra una *cache*, se hace referencia a los niveles de *zoom* de esta pirámide.

Primera parte

Primero queremos sembrar la *cache* de la capa de OpenStreetMap, en la zona de Girona. Para hacer esto, escribid un fichero `seed.yaml` que contenga una tarea de sembrado que haga referencia a la *cache* apropiada y a una cobertura con el *bounding box* de Girona, para niveles de *zoom* del **1** al **7**.

Una vez escrito el fichero `seed.yaml`, se puede hacer el sembrado ejecutando:

```
$ mapproxy-seed -f mapproxy.yaml -s seed.yaml -i
```

Si el servicio estuviera en producción, cambiaríamos **-i** por **-seed=ALL** para poder automatizarlo.

Segunda parte

A continuación puedes crear una tarea de *cache* de la capa de la ortofoto para el grid UTM o el `GLOBAL_MERCATOR`, para niveles de *zoom* del 1 al 7 y el mismo *coverage*.

5.2.3 Limpiando *caches*

Para asegurar que solo tenemos la *cache* de los datos que se usan en la oficina, vamos a crear una tarea de limpieza que borre los datos a partir del nivel 8 de la cache de la ortofoto del ICC en coordenadas UTM, pero solo aquellas teselas que tengan más de **1 semana**, **2 días**, **3 horas** y **4 minutos**.

De esta forma mantenemos los niveles superiores pero nos deshacemos de aquellas teselas que no se visitan desde hace un tiempo.

5.2.4 Comprobación de tareas del seed

Si ejecutamos el comando **mapproxy-seed** pasando como parámetro la opción **–summary**

```
$ mapproxy-seed -f mapproxy.yaml -s seed.yaml --summary
```

obtendremos el siguiente resumen de las tareas de sembrado y limpieza de teselas.

```
========== Seeding tasks ==========
  girona_osm:
    Seeding cache 'osm_cache' with grid 'GLOBAL_MERCATOR' in EPSG:900913
    Limited to: 2.67000, 41.88000, 2.97000, 42.07000 (EPSG:4326)
    Levels: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    Overwriting: no tiles
  girona_icc:
    Seeding cache 'icc_cache' with grid 'utm_girona' in EPSG:25831
    Limited to: 2.66902, 41.87953, 2.97009, 42.07047 (EPSG:4326)
    Levels: [1, 2, 3, 4, 5, 6, 7]
    Overwriting: no tiles
========== Cleanup tasks ==========
```

```
  girona:
    Cleaning up cache 'icc_cache' with grid 'GLOBAL_MERCATOR' in EPSG:900913
    Limited to: 2.67000, 41.88000, 2.97000, 42.07000 (EPSG:4326)
    Levels: [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
    Remove: tiles older than 2013-01-25 15:20:58
  girona:
    Cleaning up cache 'icc_cache' with grid 'GLOBAL_GEODETIC' in EPSG:4326
    Limited to: 2.67000, 41.88000, 2.97000, 42.07000 (EPSG:4326)
    Levels: [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
    Remove: tiles older than 2013-01-25 15:20:58
  girona:
    Cleaning up cache 'icc_cache' with grid 'utm_girona' in EPSG:25831
    Limited to: 2.66902, 41.87953, 2.97009, 42.07047 (EPSG:4326)
    Levels: [8, 9, 10, 11]
    Remove: tiles older than 2013-01-25 15:20:58
```

Por otra parte, si ejecutamos el servidor de pruebas (comando **mapproxy-util**) después de haber sembrado la *cache*, en su salida por consola se ven las peticiones WMS que está sirviendo, pero **no** las peticiones al *source* que debería estar haciendo (porque todas esas peticiones se han hecho durante el proceso de sembrado).

Te recordamos que para lanzar un servidor debes usar la orden:

```
$ mapproxy-util serve-develop mapproxy.yaml
```

5.3 Servir un fichero *MBTiles* creado con TileMill

Te sugerimos que para resolver los ejercicios inicies un proyecto nuevo llamado *ej03*:

```
$ cd /home/user/mapproxy-workshop/
$ mapproxy-util create -t base-config ej03
$ cd ej03
$ leafpad mapproxy.yaml &
```

y borres el contenido del archivo usando la combinación de teclas `Control+A` y después la tecla `Supr`.

Attention: Para este ejercicio deberás emplear un archivo con la configuración trabajada en el *Ejercicio: acelerar el acceso a un WMS*.

Note: Puedes copiar el archivo `mapproxy.yaml` del *Ejercicio: acelerar el acceso a un WMS* o descargarlo de aquí.

El objetivo de este ejercicio es montar una capa en MapProxy que sirva una *cache* en formato MBTiles generada en TileMill. Es decir, realizamos todo el proceso de diseño cartográfico con esta herramienta y después ofrecemos a nuestros usuarios dicho trabajo con cualquiera de los servicios de MapProxy, aunque como es normal, se tendrán los mejores resultados en clientes que consuman directamente la *cache* sin tener que *resamplear* las teselas al tratarse de un dato vectorial.

Note: TileMill es una aplicación de *software* libre para el diseño de cartografía usando un lenguaje similar a las hojas de estilo CSS que se utilizan en diseño *web*. Una de las salidas de TileMill es la *cache* en formato MBTiles.

Note: El formato MBTiles es en esencia una base de datos SQLite con un esquema predefinido para almacenar teselas. Tiene la ventaja de ser muy compacto porque en un único fichero se pueden almacenar miles de imágenes de una forma estandarizada.

El fichero *MBTiles* proporcionado consiste en una capa de la zona de trabajo del taller en la que se muestran carreteras y edificios en tonos de gris y una serie de puntos con la ubicación de zonas de aparcamiento. El archivo se puede descargar de aquí.

El *grid* que define el fichero *MBTiles* es igual que el usado por Google Maps solo que se han exportado las teselas hasta el nivel 16, es decir:

```
grids:
  parkings:
    base: GLOBAL_MERCATOR
    num_levels: 17
```

5.3.1 Parte única

Este ejercicio consiste en definir una nueva capa en MapProxy que apunte a una *cache* que no tiene *sources* (se debe indicar como una lista vacía porque el elemento es obligatorio). La cache ha configurar es de tipo `mbtiles` y hay que indicar la ubicación del fichero que habrá que dejar en la carpeta `cache_data`.

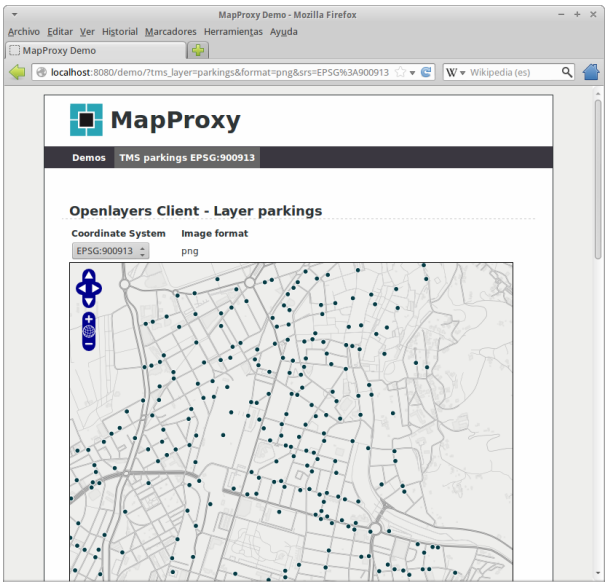


Figure 5.5: TMS de la capa de *parkings* diseñada en TileMill

Te recordamos que para lanzar un servidor debes usar la orden:

```
$ mapproxy-util serve-develop mapproxy.yaml
```

En la siguiente figura se muestran las dos capas accedidas por separado desde un cliente GIS de escritorio (QGis) en el que se ha establecido una transparencia del 50% a la capa de ortofoto de tal forma que las zonas de aparcamiento se visualizan de forma más efectiva.

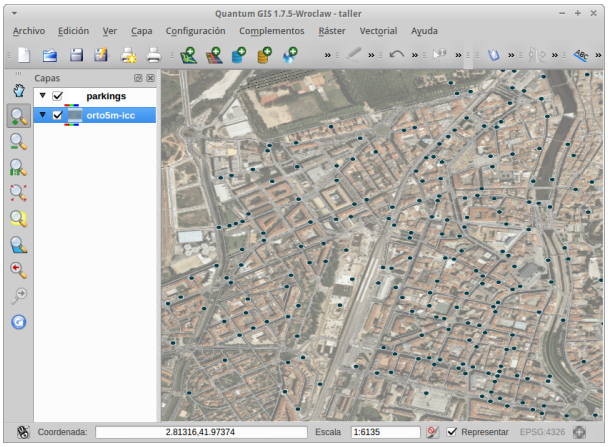


Figure 5.6: Acceso a las dos capas mediante WMS

5.4 Extensión: propuesta de ejercicios

A continuación se proponen algunos ejercicios que afianzan los contenidos repasados en el taller y van mas allá en las funcionalidades de MapProxy que no se han visitado en el taller.

1. Ofrecer WMTS/TMS de servicios propios

   Esto es, a partir de un servicio WMS de nuestra organización (o de los ofrecidos por los diferentes servidores de mapa de *OSGeo Live*), ofrecer un servicio TMS y WMTS cacheado de ciertas capas para permitir un acceso más eficiente a las mismas.

2. Restructurar árboles de capas como un nuevo servicio

   Como continuación del anterior ejercicio, a partir de nuevo de un conjunto de servicios WMS de nuestra organización, reordenarlos y presentarlos a nuestros usuarios de una forma diferente, integrando varios orígenes de datos en un único servicio, creando grupos de capas por diferentes temáticas, etc.

3. Redirigir el `getLegendgraphic` y el `getFeatureInfo`

   El protocolo WMS dispone de dos peticiones adicionales a la petición de mapa `getMap`. MapProxy permite dar acceso a estas dos peticiones e incluso transformarlos usando hojas de estilo XSL.

4. Publicar servicios diseñados con TileMill (XML de Mapnik)

   Además de publicar un MBTiles, podemos publicar en MapProxy directamente un archivo de configuración de Mapnik, que puede haber sido generado con TileMill por ejemplo. Esto convierte a MapProxy efectivamente en un servidor de mapas.

5. Configurar coberturas con orígenes de datos OGR

   Utilizar un *shapefile* o una tabla en PostGIS para delimitar las zonas que afectan tanto a un *source* como a las tareas de *seeding* y *cleanup*. Por ejemplo, puedes integrar varios servicios WMS municipales limitando cada origen a su límite administrativo.

6. Modo multimapa

   Hasta ahora solo hemos visto la generación de un servicio de MapProxy a partir de un único archivo de configuración. MapProxy admite también un modo *multimapa* en el que es posible publicar un número indeterminado de archivos de configuración.

CHAPTER 6

Referencias

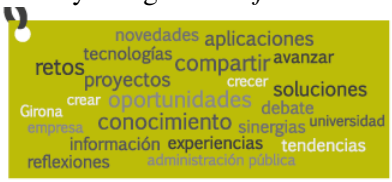
- Materiales del taller https://github.com/geoinquietosvlc/mapproxy-workshop
- Documentación del taller https://mapproxy-workshop.readthedocs.org/en/feature-sig_libre_viii
- Web oficial de MapProxy http://mapproxy.org
- http://valencia.geoinquietos.org

CHAPTER 7

Sobre el taller

La principal motivación de las Jornadas de SIG Libre en Girona, es arrojar un poco más de luz sobre el tema del *software libre geoespacial* y que este evento sea el punto de encuentro para una Comunidad, la de usuarios de SIG, en el cual poder compartir e intercambiar experiencias y conocimientos, así como establecer un espacio de debate activo poniendo de manifiesto la salud y el vigor del *software* SIG Libre.

8as Jornadas SIG Libre
26, 27 y 28 marzo 2014
Girona



Note: La url del taller es **http://bit.ly/siglibre8-mapproxy**

7.1 Facilitadores

- Pedro-Juan Ferrer @vehrka · pferrer@osgeo.org
- Jorge Sanz @xurxosanz · jsanz@osgeo.org

7.2 Nivel requerido: básico

Los asistentes deberán conocer conceptos básicos del protocolo WMS y manejo básico de consola GNU/Linux (cambiar de carpeta, listar contenidos).

7.3 Qué vamos a ver

MapProxy es un servidor de teselas y proxy WMS Open Source, acelera las aplicaciones de mapas a través de la pregeneración de tiles integrando múltiples fuentes de datos y almacenándolos en una caché

El objetivo del taller es dar a conocer la aplicación MapProxy; explicando cuáles son sus funcionalidades básicas, cuáles son sus potencialidades, repasar algunos casos de éxito y finalmente escribir y desplegar una configuración básica con las opciones más comunes.

La primera parte del taller consistirá en realizar una introducción, instalación del software, creación de un proyecto de ejemplo y comprobar su funcionamiento.

En la segunda parte del taller se revisarán algunos casos de uso de la aplicación y se realizarán ejercicios que resuelvan algunas de las dudas más frecuentes a la hora de empezar a usar este software.

7.4 Aplicaciones necesarias

Se recomienda emplear un sistema Operativo GNU/Linux basado en Debian/Ubuntu con los siguientes paquetes instalados:

- Navegador web
- Consola
- Editor de ficheros (gedit sirve pero **vim its a win!!!**)
- Algunas librerías de desarrollo y componentes Python

7.5 Autores

- Pedro-Juan Ferrer @vehrka · pferrer@osgeo.org
  - Project Manager en Omnium SI
  - Geofriki
- Iván Sanchez @realivansanchez · ivan@sanchezortega.es
  - Developer en Aptomar
  - Más geofriki aún si cabe
- Jorge Sanz @xurxosanz · jsanz@osgeo.org
  - GISguy en Prodevelop
  - Geofriki

7.6 Licencia