QGIS Server 3.40 User Guide

QGIS Project

iul. 14, 2026

Cuprins

1 Introducere **3**

2 Noțiuni de bază **5**
2.1 Installation on Debian-based systems . . . 5
2.1.1 Server HTTP Apache . . . 7
2.1.2 Server HTTP NGINX . . . 9
2.1.3 Xvfb . . . 13
2.2 Installation on Windows . . . 14
2.3 Serve a project . . . 16
2.4 Configure your project . . . 17
2.4.1 WMS capabilities . . . 19
2.4.2 WMTS capabilities . . . 21
2.4.3 WFS/OAPIF capabilities . . . 22
2.4.4 WCS capabilities . . . 23
2.4.5 Reglarea fină a OWS-ului dvs. . . . 24
2.5 Integration with third parties . . . 24
2.5.1 Integration with QGIS Desktop . . . 24
2.5.2 Integration with MapProxy . . . 24
2.5.3 Integration with QWC2 . . . 24

3 Servicii **25**
3.1 Basics . . . 25
3.1.1 SERVICE . . . 26
3.1.2 REQUEST . . . 26
3.1.3 MAP . . . 26
3.1.4 FILE_NAME . . . 26
3.1.5 Short name . . . 27
3.2 Web Map Service (WMS) . . . 27
3.2.1 GetCapabilities . . . 28
3.2.2 GetMap . . . 28
3.2.3 GetFeatureInfo . . . 36
3.2.4 GetLegendGraphic . . . 39
3.2.5 GetStyle(s) . . . 54
3.2.6 DescribeLayer . . . 54
3.2.7 GetPrint . . . 55
3.2.8 GetProjectSettings . . . 59
3.2.9 GetSchemaExtension . . . 59
3.2.10 External WMS layers . . . 60
3.2.11 Redlining . . . 60
3.3 Web Feature Service (WFS) . . . 63
3.3.1 GetCapabilities . . . 63

3.3.2 GetFeature . . . 64
3.3.3 DescribeFeatureType . . . 70
3.3.4 Tranzacție . . . 71
3.4 Web Coverage Service (WCS) . . . 74
3.4.1 GetCapabilities . . . 74
3.4.2 DescribeCoverage . . . 75
3.4.3 GetCoverage . . . 77
3.5 Web Map Tile Service (WMTS) . . . 78
3.5.1 GetCapabilities . . . 78
3.5.2 GetTile . . . 79
3.5.3 GetFeatureInfo . . . 80
3.6 OGC API Features . . . 81
3.6.1 Resource representation . . . 82
3.6.2 Endpoints . . . 82
3.6.3 Pagination . . . 86
3.6.4 Filtrarea entităților . . . 86
3.6.5 Feature sorting . . . 88
3.6.6 Attribute selection . . . 88
3.6.7 Customize the HTML pages . . . 88

4 Catalog **91**

5 Plugin-uri **95**
5.1 Lista plugin-urilor . . . 95
5.2 Location of plugins . . . 95
5.3 Instalarea . . . 95
5.3.1 Manually with a ZIP . . . 95
5.3.2 Cu un instrument în linie de comandă . . . 96
5.4 Configurarea serverului HTTP . . . 96
5.4.1 Apache . . . 96
5.5 Cum se utilizează un plugin . . . 97

6 Configurare avansată **99**
6.1 Jurnalizarea . . . 99
6.2 Variabile de mediu . . . 99
6.3 Settings summary . . . 104
6.4 Connection to service file . . . 105
6.5 Add fonts to your linux server . . . 105

7 Development Server **107**

8 Containerized deployment **109**
8.1 Simple docker images . . . 109
8.1.1 First run . . . 111
8.1.2 Usable sample . . . 111
8.1.3 Cleanup . . . 112
8.2 Docker stacks . . . 112
8.2.1 Swarm/docker-compose . . . 112
8.2.2 Kubernetes . . . 113
8.3 Cloud deployment . . . 117
8.3.1 AWS usecase . . . 117

9 Frequently Asked Question **119**

CAPITOLUL 1

Introducere

QGIS Server is an open source WMS, WFS, OGC API for Features 1.0 (WFS3) and WCS implementation that, in addition, implements advanced cartographic features for thematic mapping. QGIS Server is a FastCGI/CGI (Common Gateway Interface) application written in C++ that works together with a web server (e.g., Apache, Nginx). It has Python plugin support allowing for fast and efficient development and deployment of new features.

Pentru logica GIS și pentru redarea hărților, serverul QGIS utilizează în fundal aplicația QGIS. În plus, grafica și programarea în C++ fac uz de biblioteca independentă de platformă Qt. Spre deosebire de alte aplicații WMS, serverul QGIS utilizează ca limbaj de configurare regulile cartografice, atât pentru configurația serverului, cât și pentru regulile definite de utilizator.

Atât timp cât QGIS desktop și QGIS Server folosesc aceleași biblioteci de vizualizare, hărțile care sunt publicate pe web arată la fel ca și în GIS desktop.

In the following sections, we will provide a sample configuration to set up a QGIS Server on Linux (Debian, Ubuntu and derivatives) and on Windows. For more information about server plugin development, please read server_plugins.

Se acordă permisiunea de a copia, distribui și/sau modifica acest document conform termenilor GNU Free Documentation License, versiunea 1.3 sau orice altă versiune ulterioară, publicată de Free Software Foundation; fără Secțiunile Invariabile și fără textele de pe coperțile Față/Spate.

O copie a acestei licențe este inclusă în secțiunea gnu_fdl.

CAPITOLUL 2

Noțiuni de bază

2.1 Installation on Debian-based systems

We will give a short and simple installation how-to for a minimal working configuration on Debian based systems (including Ubuntu and derivatives). However, many other distributions and OSs provide packages for QGIS Server.

Notă

In Ubuntu you can use your regular user, prepending `sudo` to commands requiring admin permissions. In Debian you can work as admin (`root`), without using `sudo`.

Requirements and steps to add official QGIS repositories to install QGIS Server on a Debian based system are provided in QGIS installers page. You may want to install at least the latest Long Term Release.

Once the target version repository is configured and QGIS Server installed, you can test the installation with:

```
/usr/lib/cgi-bin/qgis_mapserv.fcgi
```

If you get the following output, the server is correctly installed.

Notă

Depending on the version of QGIS, you might see slightly different output reported when you run `qgis_mapserv.fcgi`.

```
QFSFileEngine::open: No file name specified
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment␣
↪variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment␣
↪variable.
Content-Length: 206
Content-Type: text/xml; charset=utf-8
```

(continues on next page)

(continuare din pagina precedentă)

```
<ServiceExceptionReport version="1.3.0" xmlns="https://www.opengis.net/ogc">
 <ServiceException code="Service configuration error">Service unknown or␣
↪unsupported</ServiceException>
</ServiceExceptionReport>
```

Notă

As seen below, QGIS reports a Status 400 code, which correctly identifies the request has failed because there is no active http session. This is not a bug and indicates the server is functioning properly.

```
Application path not initialized
Application path not initialized
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment␣
↪variable.
"Loading native module /usr/lib/qgis/server/libdummy.so"
"Loading native module /usr/lib/qgis/server/liblandingpage.so"
"Loading native module /usr/lib/qgis/server/libwcs.so"
"Loading native module /usr/lib/qgis/server/libwfs.so"
"Loading native module /usr/lib/qgis/server/libwfs3.so"
"Loading native module /usr/lib/qgis/server/libwms.so"
"Loading native module /usr/lib/qgis/server/libwmts.so"
QFSFileEngine::open: No file name specified
Content-Length: 102
Content-Type: application/json
Server:  QGIS FCGI server - QGIS version 3.40.0-Bratislava
Status:  400
[{"code":"Bad request error","description":"Requested URI does not match any␣
↪registered API handler"}]
```

Let's add a sample project. You can use your own, or one from Training demo data:

```
mkdir /home/qgis/projects/
cd /home/qgis/projects/
wget https://github.com/qgis/QGIS-Training-Data/archive/release_3.40.zip
unzip release_3.40.zip
mv QGIS-Training-Data-release_3.40/exercise_data/qgis-server-tutorial-data/world.
↪qgs .
mv QGIS-Training-Data-release_3.40/exercise_data/qgis-server-tutorial-data/
↪naturalearth.sqlite .
```

Of course, you can use your favorite GIS software to open this file and take a look at the configuration and available layers.

To properly deploy QGIS server you need a HTTP server. Recommended choices are Apache or NGINX.

2.1.1 Server HTTP Apache

> **Notă**
>
> In the following, please replace `qgis.demo` with the name or IP address of your server.

1. Install Apache and mod_fcgid:

```
apt install apache2 libapache2-mod-fcgid
```

2. You can run QGIS Server on your default website, but let's configure a virtualhost specifically for this, as follows.

   1. In the `/etc/apache2/sites-available` directory, create a file called `qgis.demo.conf`, with this content:

```
<VirtualHost *:80>
  ServerAdmin webmaster@localhost
  ServerName qgis.demo

  DocumentRoot /var/www/html

  # Apache logs (different than QGIS Server log)
  ErrorLog ${APACHE_LOG_DIR}/qgis.demo.error.log
  CustomLog ${APACHE_LOG_DIR}/qgis.demo.access.log combined

  # Longer timeout for WPS... default = 40
  FcgidIOTimeout 120

  FcgidInitialEnv LC_ALL "en_US.UTF-8"
  FcgidInitialEnv PYTHONIOENCODING UTF-8
  FcgidInitialEnv LANG "en_US.UTF-8"

  # QGIS log
  FcgidInitialEnv QGIS_SERVER_LOG_STDERR 1
  FcgidInitialEnv QGIS_SERVER_LOG_LEVEL 0

  # default QGIS project
  SetEnv QGIS_PROJECT_FILE /home/qgis/projects/world.qgs

  # QGIS_AUTH_DB_DIR_PATH must lead to a directory writeable by the Server
↪'s FCGI process user
  FcgidInitialEnv QGIS_AUTH_DB_DIR_PATH "/home/qgis/qgisserverdb/"
  FcgidInitialEnv QGIS_AUTH_PASSWORD_FILE "/home/qgis/qgisserverdb/qgis-
↪auth.db"

  # Set pg access via pg_service file
  SetEnv PGSERVICEFILE /home/qgis/.pg_service.conf
  FcgidInitialEnv PGPASSFILE "/home/qgis/.pgpass"

  # if qgis-server is installed from packages in debian based distros this␣
↪is usually /usr/lib/cgi-bin/
  # run "locate qgis_mapserv.fcgi" if you don't know where qgis_mapserv.
↪fcgi is
  ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
  <Directory "/usr/lib/cgi-bin/">
    AllowOverride None
    Options +ExecCGI -MultiViews -SymLinksIfOwnerMatch
    Require all granted
  </Directory>
```

(continues on next page)

(continuare din pagina precedentă)

```
  <IfModule mod_fcgid.c>
  FcgidMaxRequestLen 26214400
  FcgidConnectTimeout 60
  </IfModule>

</VirtualHost>
```

Notă

Some of the above configuration options are explained in the Server *environment variables* and pg_service file sections.

2. Let's now create the directories that will store the QGIS Server logs and the authentication database:

```
mkdir -p /var/log/qgis/
chown www-data:www-data /var/log/qgis
mkdir -p /home/qgis/qgisserverdb
chown www-data:www-data /home/qgis/qgisserverdb
```

Notă

`www-data` is the Apache user on Debian based systems and we need Apache to have access to those locations or files. The `chown www-data...` commands change the owner of the respective directories and files to `www-data`.

3. We can now enable the virtual host and the `fcgid` mod if it's not already done:

```
a2enmod fcgid
a2enmod rewrite
a2ensite qgis.demo
```

4. Now restart Apache for the new configuration to be taken into account:

```
systemctl restart apache2
```

5. Now that Apache knows that he should answer requests to http://qgis.demo we also need to setup the client system so that it knows who `qgis.demo` is. We do that by adding `127.0.0.1 qgis.demo` in the hosts file.

```
# Replace 127.0.0.1 with the IP of your server.
sh -c "echo '127.0.0.1 qgis.demo' >> /etc/hosts"
```

Important

Remember that both the `qgis.demo.conf` and `/etc/hosts` files should be configured for your setup to work. You can also test the access to your QGIS Server from other clients on the network (e.g. Windows or macos machines) by going to their `/etc/hosts` file and point the `qgis.demo` name to whatever IP the server machine has on the network (not `127.0.0.1` as it is the local IP, only accessible from the local machine). On `*nix` machines the `hosts` file is located in `/etc`, while on Windows it's under the `C:\Windows\System32\drivers\etc` directory. Under Windows you need to start your text editor with administrator privileges before opening the hosts file.

QGIS Server is now available at http://qgis.demo. To check, type in a browser, as in the simple case:

```
http://qgis.demo/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&
↪REQUEST=GetCapabilities
```

2.1.2 Server HTTP NGINX

> **Notă**
>
> In the following, please replace `qgis.demo` with the name or IP address of your server.

You can also use QGIS Server with NGINX. Unlike Apache, NGINX does not automatically spawn FastCGI processes. The FastCGI processes are to be started by something else.

Instalare NGINX:

```
apt install nginx
```

- As a first option, you can use **spawn-fcgi** or **fcgiwrap** to start and manage the QGIS Server processes. Official Debian packages exist for both. When you have no X server running and you need, for example, printing, you can use *xvfb*.
- Another option is to rely on **Systemd**, the init system for GNU/Linux that most Linux distributions use today. One of the advantages of this method is that it requires no other components or processes. It's meant to be simple, yet robust and efficient for production deployments.

NGINX Configuration

The **include fastcgi_params;** used in the previous configuration is important, as it adds the parameters from `/etc/nginx/fastcgi_params`:

```
fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;

fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;
fastcgi_param  REQUEST_SCHEME     $scheme;
fastcgi_param  HTTPS              $https if_not_empty;

fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx/$nginx_version;

fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;
```

Moreover, you can use some *Variabile de mediu* to configure QGIS Server. In the NGINX configuration file, `/etc/nginx/nginx.conf`, you have to use `fastcgi_param` instruction to define these variables as shown below:

```
location /qgisserver {
     gzip            off;
     include         fastcgi_params;
     fastcgi_param   QGIS_SERVER_LOG_STDERR  1;
     fastcgi_param   QGIS_SERVER_LOG_LEVEL   0;
     fastcgi_pass    unix:/var/run/qgisserver.socket;
 }
```

FastCGI wrappers

Atenţionare

fcgiwrap is easier to set up than **spawn-fcgi**, because it's already wrapped in a Systemd service. But it also leads to a solution that is much slower than using spawn-fcgi. With fcgiwrap, a new QGIS Server process is created on each request, meaning that the QGIS Server initialization process, which includes reading and parsing the QGIS project file, is done on each request. With spawn-fcgi, the QGIS Server process remains alive between requests, resulting in much better performance. For that reason, spawn-fcgi is recommended for production use.

spawn-fcgi

If you want to use spawn-fcgi:

1. The first step is to install the package:

   ```
   apt install spawn-fcgi
   ```

2. Then, introduce the following block in your NGINX server configuration:

   ```
   location /qgisserver {
       gzip            off;
       include         fastcgi_params;
       fastcgi_pass    unix:/var/run/qgisserver.socket;
   }
   ```

3. And restart NGINX to take into account the new configuration:

   ```
   systemctl restart nginx
   ```

4. Finally, considering that there is no default service file for spawn-fcgi, you have to manually start QGIS Server in your terminal:

   ```
   spawn-fcgi -s /var/run/qgisserver.socket \
                  -U www-data -G www-data -n \
                  /usr/lib/cgi-bin/qgis_mapserv.fcgi
   ```

QGIS Server is now available at http://qgis.demo/qgisserver.

Notă

When using spawn-fcgi, you may directly define environment variables before running the server. For example: `export QGIS_SERVER_LOG_STDERR=1`

Of course, you can add an init script to start QGIS Server at boot time or whenever you want. For example with **systemd**:

1. Edit the file `/etc/systemd/system/qgis-server.service` with this content:

```
[Unit]
Description=QGIS server
After=network.target

[Service]
;; set env var as needed
;Environment="LANG=en_EN.UTF-8"
;Environment="QGIS_SERVER_PARALLEL_RENDERING=1"
;Environment="QGIS_SERVER_MAX_THREADS=12"
;Environment="QGIS_SERVER_LOG_LEVEL=0"
;Environment="QGIS_SERVER_LOG_STDERR=1"
;; or use a file:
;EnvironmentFile=/etc/qgis-server/env

ExecStart=spawn-fcgi -s /var/run/qgisserver.socket -U www-data -G www-data -n /
↪usr/lib/cgi-bin/qgis_mapserv.fcgi

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

2. Then enable and start the service:

```
systemctl enable --now qgis-server
```

Atenţionare

With the above commands spawn-fcgi spawns only one QGIS Server process.

fcgiwrap

Using fcgiwrap is much easier to setup than **spawn-fcgi** but it's much slower.

1. You first have to install the corresponding package:

```
apt install fcgiwrap
```

2. Then, introduce the following block in your NGINX server configuration:

```
location /qgisserver {
    gzip            off;
    include         fastcgi_params;
    fastcgi_pass    unix:/var/run/fcgiwrap.socket;
    fastcgi_param   SCRIPT_FILENAME /usr/lib/cgi-bin/qgis_mapserv.fcgi;
}
```

3. Finally, restart NGINX and **fcgiwrap** to take into account the new configuration:

```
systemctl restart nginx
systemctl restart fcgiwrap
```

QGIS Server is now available at http://qgis.demo/qgisserver.

Systemd

QGIS Server needs a running X Server to be fully usable, in particular for printing. In the case you already have a running X Server, you can use systemd services.

This method, to deploy QGIS Server, relies on two Systemd units to configure: a Socket unit and a Service unit.

1. The **QGIS Server Socket unit** defines and creates a file system socket, used by NGINX to start and communicate with QGIS Server. The Socket unit has to be configured with `Accept=false`, meaning that the calls to the `accept()` system call are delegated to the process created by the Service unit. It is located in `/etc/systemd/system/qgis-server@.socket`, which is actually a template:

```
[Unit]
Description=QGIS Server Listen Socket (instance %i)

[Socket]
Accept=false
ListenStream=/var/run/qgis-server-%i.sock
SocketUser=www-data
SocketGroup=www-data
SocketMode=0600

[Install]
WantedBy=sockets.target
```

2. Now enable and start sockets:

```
for i in 1 2 3 4; do systemctl enable --now qgis-server@$i.socket; done
```

3. The **QGIS Server Service unit** defines and starts the QGIS Server process. The important part is that the Service process' standard input is connected to the socket defined by the Socket unit. This has to be configured using `StandardInput=socket` in the Service unit configuration located in `/etc/systemd/system/qgis-server@.service`:

```
[Unit]
Description=QGIS Server Service (instance %i)

[Service]
User=www-data
Group=www-data
StandardOutput=null
StandardError=journal
StandardInput=socket
ExecStart=/usr/lib/cgi-bin/qgis_mapserv.fcgi
EnvironmentFile=/etc/qgis-server/env

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

> **Notă**
>
> The QGIS Server *environment variables* are defined in a separate file, `/etc/qgis-server/env`. It could look like this:
>
> ```
> QGIS_PROJECT_FILE=/etc/qgis/myproject.qgs
> QGIS_SERVER_LOG_STDERR=1
> QGIS_SERVER_LOG_LEVEL=3
> ```

4. Now start socket service:

```
for i in 1 2 3 4; do systemctl enable --now qgis-server@$i.service; done
```

5. Finally, for the NGINX HTTP server, let's introduce the configuration for this setup:

```
upstream qgis-server_backend {
   server unix:/var/run/qgis-server-1.sock;
   server unix:/var/run/qgis-server-2.sock;
   server unix:/var/run/qgis-server-3.sock;
   server unix:/var/run/qgis-server-4.sock;
}

server {
   …

   location /qgis-server {
       gzip off;
       include fastcgi_params;
       fastcgi_pass qgis-server_backend;
   }
}
```

6. Now restart NGINX for the new configuration to be taken into account:

```
systemctl restart nginx
```

Thanks to Oslandia for sharing their tutorial.

2.1.3 Xvfb

QGIS Server needs a running X Server to be fully usable, in particular for printing. On servers it is usually recommended not to install it, so you may use `xvfb` to have a virtual X environment.

If you're running the Server in graphic/X11 environment then there is no need to install xvfb. More info at https://www.itopen.it/qgis-server-setup-notes/.

1. To install the package:

```
apt install xvfb
```

2. Create the service file, `/etc/systemd/system/xvfb.service`, with this content:

```
[Unit]
Description=X Virtual Frame Buffer Service
After=network.target

[Service]
ExecStart=/usr/bin/Xvfb :99 -screen 0 1024x768x24 -ac +extension GLX +render -
↪noreset

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

3. Enable, start and check the status of the `xvfb.service`:

```
systemctl enable --now xvfb.service
systemctl status xvfb.service
```

4. Then, according to your HTTP server, you should configure the **DISPLAY** parameter or directly use **xvfb-run**.
   - Using Apache:
     1. Add to your *Fcgid* configuration (see *Server HTTP Apache*):

```
FcgidInitialEnv DISPLAY       ":99"
```

2. Restart Apache for the new configuration to be taken into account:

```
systemctl restart apache2
```

- Using NGINX
  - With spawn-fcgi using `xvfb-run`:

```
xvfb-run /usr/bin/spawn-fcgi -f /usr/lib/cgi-bin/qgis_mapserv.fcgi \
                             -s /tmp/qgisserver.socket \
                             -G www-data -U www-data -n
```

  - With the **DISPLAY** environment variable in the HTTP server configuration.

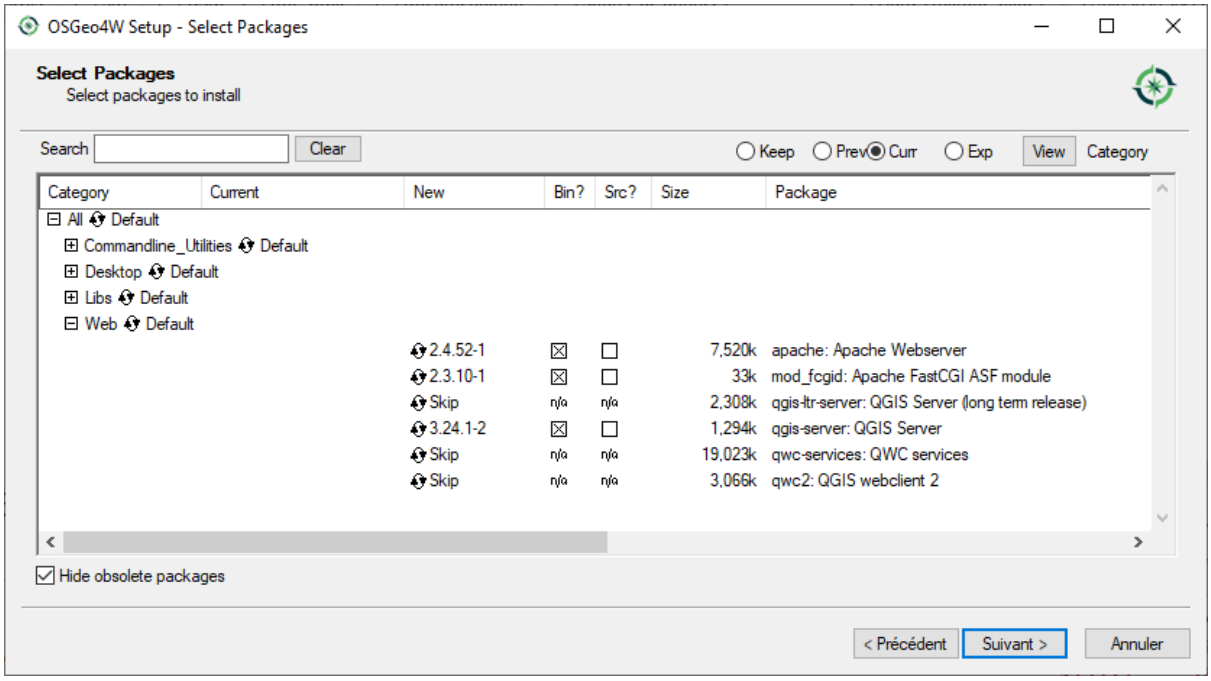
```
fastcgi_param  DISPLAY       ":99";
```

2.2 Installation on Windows

QGIS Server can also be installed on Windows systems using the OSGeo4W network installer (https://qgis.org/resources/installation-guide/#windows).

A simple procedure is the following:

1. Download and run the OSGeo4W installer
2. Follow the „Advanced Install" and install the **QGIS Desktop**, **QGIS Server apache** and **mod_fcgid** packages.



3. Apache is not directly installed as a service on Windows. You need to:
   1. Right-click the `OSGeo4W.bat` file at the root of the `C:\OSGeo4W\` folder (if the default installation paths have been used) and select *Run as administrator*
   2. In the console, run `apache-install.bat`, which will output

```
> apache-install.bat
Installing the 'Apache OSGeo4W Web Server' service
The 'Apache OSGeo4W Web Server' service is successfully installed.
Testing httpd.conf....
Errors reported here must be corrected before the service can be started.
...
```

The service is started as you can notice in the report. But the server may fail to run due to missing custom configuration.

4. Edit the `C:\OSGeo4w\apps\apache\conf\httpd.conf` file with the following changes (various other combinations are possible):

Scop	Existing config	Înlocuire
(Optional) Customize the address to listen to using an IP and/or port, You can and add as many entries as you wish.	`Listen ${SRVPORT}`	`Listen localhost:8080`
Indicate where to find the script files	`ScriptAlias /cgi-bin/ "${SRVROOT}/cgi-bin/"`	`ScriptAlias /cgi-bin/ "C:/OSGeo4W/apps/qgis/bin/"`

Provide permissions on the script folder	`<Directory "${SRVROOT}/cgi-bin">`
	`  AllowOverride None`
	`  Options None`
	`  Require all granted`
	`</Directory>`
		`<Directory "C:/OSGeo4W/apps/qgis/bin">`
		`  SetHandler cgi-script`
		`  AllowOverride None`
		`  Options ExecCGI`
		`  Require all granted`
		`</Directory>`

Enable file extensions to use for script files. Uncomment and complete	`#AddHandler cgi-script .cgi`	`AddHandler cgi-script .cgi .exe`

Add more OSGeo4W custom configuration variables	`# parse OSGeo4W apache conf files`
	`IncludeOptional "C:/OSGeo4W/httpd.d/httpd_*.conf"`
		`# parse OSGeo4W apache conf files`
		`IncludeOptional "C:/OSGeo4W/httpd.d/httpd_*.conf"`
		`SetEnv GDAL_DATA "C:/OSGeo4W/share/gdal"`
		`SetEnv QGIS_AUTH_DB_DIR_PATH "C:/OSGeo4W/apps/qgis/resources"`
		
		`# default QGIS project`
		`SetEnv QGIS_PROJECT_FILE "C:/Users/*Your USERNAME*/qgis_projects/qgis-server-tutorial-data/world.qgs"`

5. Restart the Apache web server

```
> apache-restart.bat
```

6. Let's add a sample project. You can use your own, or one from Training demo data. Make sure that you place it in the same folder that your *QGIS_PROJECT_FILE* environment variable is pointing to:

```
> "C:/Users/*Your USERNAME*/qgis_projects/qgis-server-tutorial-data/world.qgs"
```

7. Open browser window to testing a GetCapabilities request to QGIS Server. Replace `localhost:8080` with the IP and port you set to listen.

```
http://localhost:8080/cgi-bin/qgis_mapserv.fcgi.exe?SERVICE=WMS&VERSION=1.3.0&
↪REQUEST=GetCapabilities
```

   A XML file with the capabilities should be returned. Your server is ready to use.

2.3 Serve a project

Now that QGIS Server is installed and running, we just have to use it.

Obviously, we need a QGIS project to work on. Of course, you can fully customize your project by defining contact information, precise some restrictions on CRS or even exclude some layers. Everything you need to know about that is described later in *Configure your project*.

But for now, we are going to use a simple project already configured and previously downloaded in `/home/qgis/projects/world.qgs`, as described above.

By opening the project and taking a quick look on layers, we know that 4 layers are currently available:

- airports
- places
- countries
- countries_shapeburst

You don't have to understand the full request for now but you may retrieve a map with some of the previous layers thanks to QGIS Server by doing something like this in your web browser to retrieve the *countries* layer:

- If you followed the above instructions to install an Apache HTTP Server:

```
http://qgis.demo/cgi-bin/qgis_mapserv.fcgi?MAP=/home/qgis/projects/world.qgs&
↪LAYERS=countries&SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&CRS=EPSG:4326&
↪WIDTH=400&HEIGHT=200&BBOX=-90,-180,90,180
```

- If you followed the above instructions to install an NGINX HTTP Server:

```
http://qgis.demo/qgisserver?MAP=/home/qgis/projects/world.qgs&LAYERS=countries&
↪SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&CRS=EPSG:4326&WIDTH=400&HEIGHT=200&
↪BBOX=-90,-180,90,180
```

- If you followed the instructions for Windows:

```
http://localhost:8080/cgi-bin/qgis_mapserv.fcgi.exe?SERVICE=WMS&
↪LAYERS=countries&VERSION=1.3.0&REQUEST=GetMap&CRS=EPSG:4326&WIDTH=400&
↪HEIGHT=200&BBOX=-90,-180,90,180
```

If you obtain the next image, then QGIS Server is running correctly:

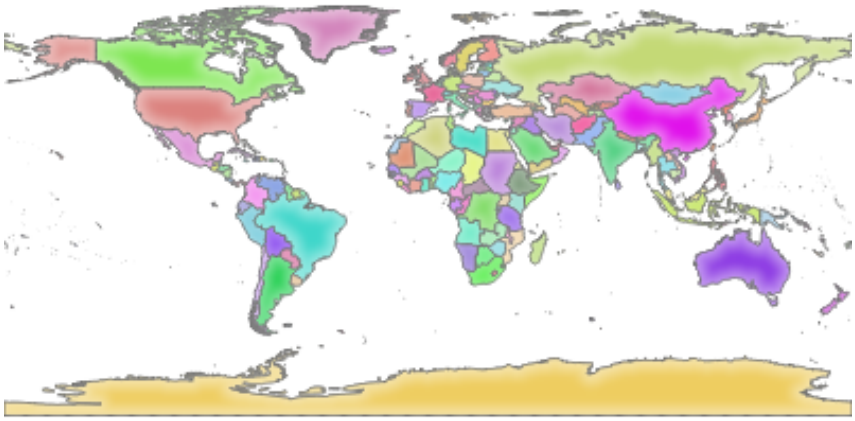


Fig. 2.1: Server response to a basic GetMap request

Note that you may define **QGIS_PROJECT_FILE** environment variable to use a project by default instead of giving a **MAP** parameter (see *Variabile de mediu*).

For example with spawn-fcgi:

```
export QGIS_PROJECT_FILE=/home/qgis/projects/world.qgs
spawn-fcgi -f /usr/lib/bin/cgi-bin/qgis_mapserv.fcgi \
           -s /var/run/qgisserver.socket \
           -U www-data -G www-data -n
```

2.4 Configure your project

To provide a new QGIS Server WMS, WFS, OAPIF or WCS, you have to create a QGIS project file with some data or use one of your current projects. Define the colors and styles of the layers in QGIS and the project CRS, if not already defined. Then, go to the *QGIS Server* menu of the *Project* ► *Properties…* dialog and provide some information about the OWS in the *Service Capabilities* tab.

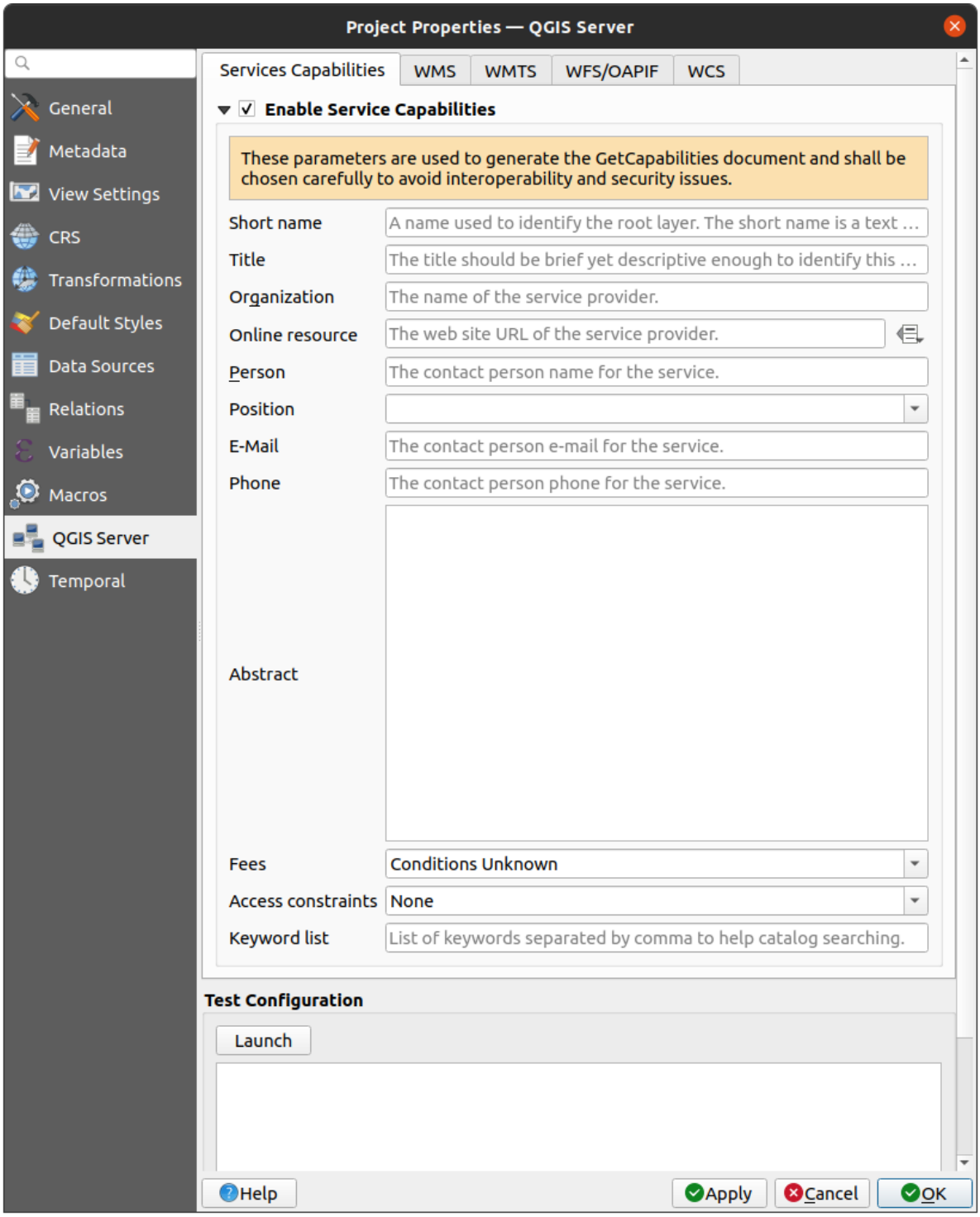


Fig. 2.2: Definitions for a QGIS Server WMS/WFS/WCS project

You have to *Enable Service Capabilities* first, if it is deactivated. This will appear in the GetCapabilities response of the WMS, WFS or WCS. If you don't check *Enable Service capabilities*, QGIS Server will use the information given in the `wms_metadata.xml` file located in the `cgi-bin` folder.

2.4.1 WMS capabilities

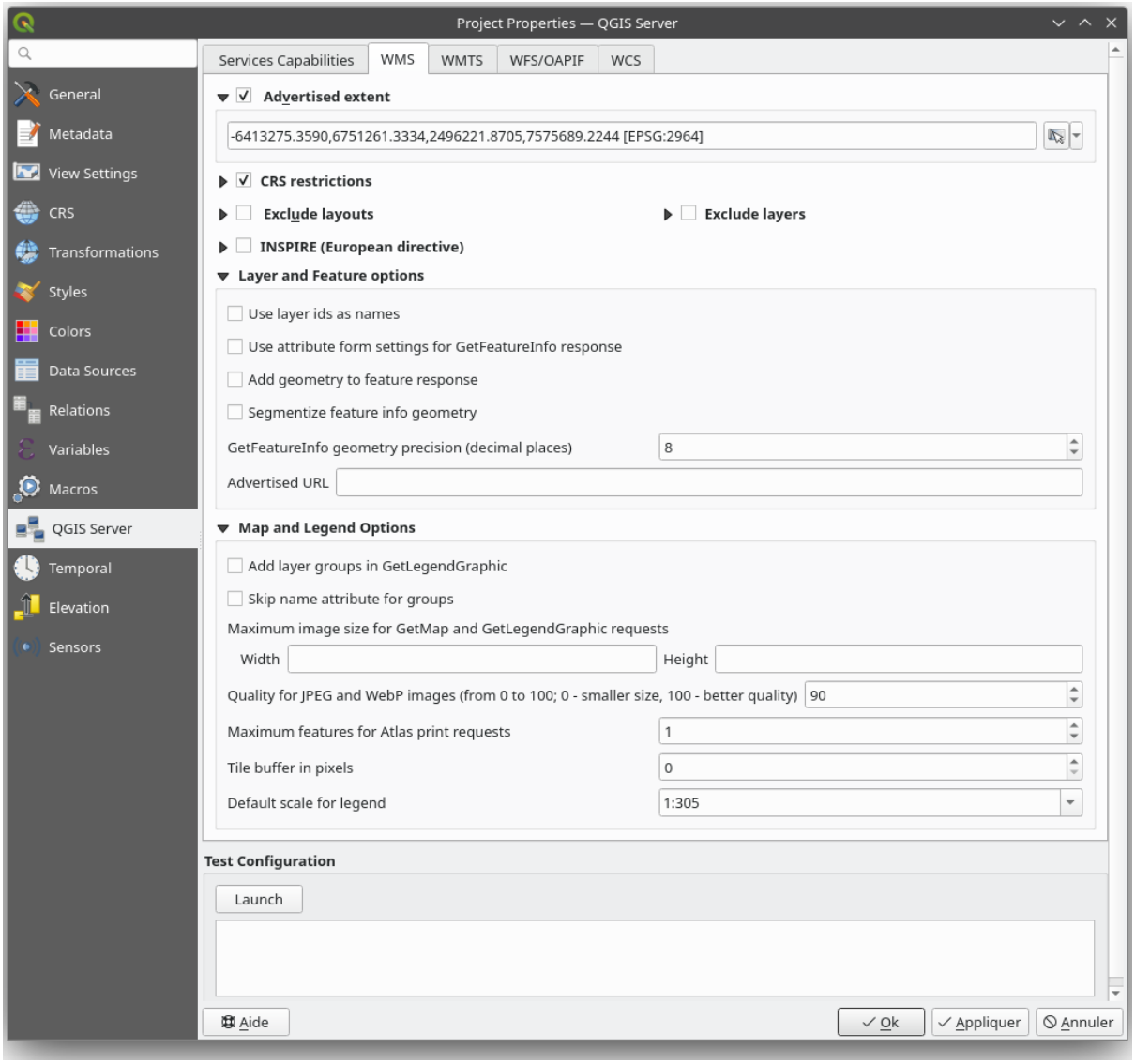


Fig. 2.3: Definitions in the WMS tab

In the *WMS* tab, you can define the options for the WMS capabilities.

- Check *Advertised extent* to define the extent advertised in the WMS GetCapabilities response. The spatial extent selector widget helps you enter the extent as a `xmin, xmax, ymin, ymax` text or pick it from the map canvas, layers, bookmarks…
- By checking *CRS restrictions*, you can restrict in which coordinate reference systems (CRS) QGIS Server will offer to render maps. It is recommended that you restrict the offered CRS as this reduces the size of the WMS GetCapabilities response. Use the button below to select those CRSs from the Coordinate Reference System Selector, or click *Used* to add the CRSs used in the QGIS project to the list.
- If you have print layouts defined in your project, they will be listed in the `GetProjectSettings` response, and they can be used by the GetPrint request to create prints, using one of the print layouts as a template. This is a QGIS-specific extension to the WMS 1.3.0 specification. If you want to exclude any print layout from being published by the WMS, check *Exclude layouts* and click the button below. Then, select a print layout from the *Select print layout* dialog in order to add it to the excluded layouts list.

- If you want to exclude any layer or layer group from being published by the WMS, check *Exclude Layers* and click the button below. This opens the *Select restricted layers and groups* dialog, which allows you to choose the layers and groups that you don't want to be published. Use the `Shift` or `Ctrl` key if you want to select multiple entries. It is recommended that you exclude from publishing the layers that you don't need as this reduces the size of the WMS GetCapabilities response which leads to faster loading times on the client side.
- *Layer and Feature Options*

  You can receive requested GetFeatureInfo as plain text, XML and GML. The default is XML.

  - If you check *Use layer ids as name*, layer ids will be used to reference layers in the `GetCapabilities` response or `GetMap LAYERS` parameter. If not, layer name or short name if defined (see vectorservermenu) is used.

    By default, layer names are used to expose layers through WMS. If multiple layers have the same name, they will be merged to a single WMS layer and cannot be requested individually. You need to take care of these layers being compatible with each other. Using the *Use layer ids as name* option ensures that multiple layers with the same name can be requested as individual layers.

  - If you wish, you can check *Add geometry to feature response*. This will include the bounding box for each feature in the GetFeatureInfo response. See also the *WITH_GEOMETRY* parameter.
  - As many web clients can't display circular arcs in geometries you have the option to segmentize the geometry before sending it to the client in a GetFeatureInfo response. This allows such clients to still display a feature's geometry (e.g. for highlighting the feature). You need to check the *Segmentize feature info geometry* to activate the option.
  - You can also use the *GetFeatureInfo geometry precision* option to set the precision of the GetFeatureInfo geometry. This enables you to save bandwidth when you don't need the full precision.
  - If one of your layers uses the Map Tip display (i.e. to show text using expressions) this will be listed inside the GetFeatureInfo output. If the layer uses a Value Map for one of its attributes, this information will also be shown in the GetFeatureInfo output.
  - If you want QGIS Server to advertise specific request URLs in the WMS GetCapabilities response, enter the corresponding URL in the *Advertised URL* field.
- *Map and Legend Options*
  - When a layer group is passed to `GetLegendGraphic` request, all of its leaf layers are added to the legend picture (however without the groups» labels). Check the *Add layer groups in GetLegendGraphic* option if you want to also insert the layer groups (and subgroups) names into the layer tree, just like in QGIS Desktop legend.
  - When QGIS project contains layer groups, they are listed in WMS capabilities document alongside with layers. If a group (its name as listed in capabilities) is included in WMS GetMap `LAYERS` parameter alongside with names of layers in that group, QGIS would duplicate the layers: once for the group and once for specific layer. If you check the *Skip name attribute for groups* option, GetCapabilities will only return title attribute for the group but not its name attribute, making it impossible to include groups in list of layers of GetMap request.
  - Furthermore, you can restrict the maximum size of the maps returned by the requests by entering the maximum width and height into the respective fields under *Maximum image size for GetMap and GetLegendGraphic requests*.
  - You can change the *Quality for JPEG and WebP images* factor. The quality factor must be in the range 0 to 100. Specify 0 for maximum compression, 100 for no compression.
  - You can change the limit for atlas features to be printed in one request by setting the *Maximum features for Atlas print requests* field.
  - When QGIS Server is used in tiled mode (see *TILED parameter*), you can set the *Tile buffer in pixels*. The recommended value is the size of the largest symbol or line width in your QGIS project.

– Depending on whether the map uses a projected CRS or a geographic CRS and if there is no information to evaluate the map unit sized symbols, you can provide reference for size through either a *Default scale for legend* or *Default map units per mm in legend*.

2.4.2 WMTS capabilities

In the *WMTS* tab you can select the layers you want to publish as WMTS and specify if you want to publish as PNG or JPEG.

Under *Grids*, for each *restricted CRS*, you can indicate whether the tile matrices should be published and configure their top-left corner, minimum and maximum scales and the last level of the tile matrices. Note that for `EPSG:3857` and `EPSG:4326` CRS, only the last level of the tile matrices can be edited.

You can also set a *Minimum scale* for the tiles display.

If you enter a URL in the *Advertised URL* field, QGIS Server will advertise this specific URL in the WMTS GetCapabilities response.

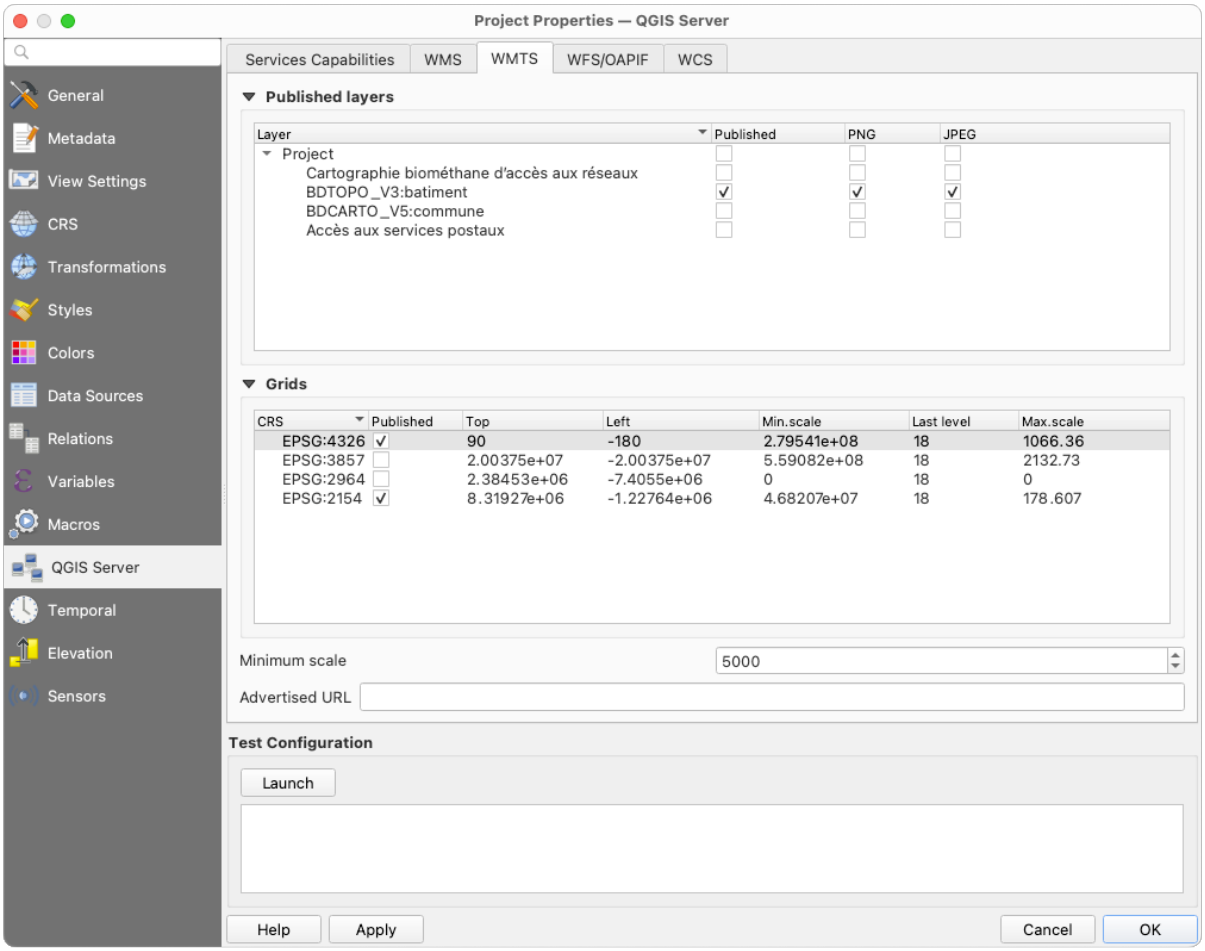


Fig. 2.4: Definitions in the WMTS tab

2.4.3 WFS/OAPIF capabilities

In the *WFS/OAPIF* tab, you can select the layers you want to publish as WFS or OAPIF, and specify if they will allow update, insert and delete operations.

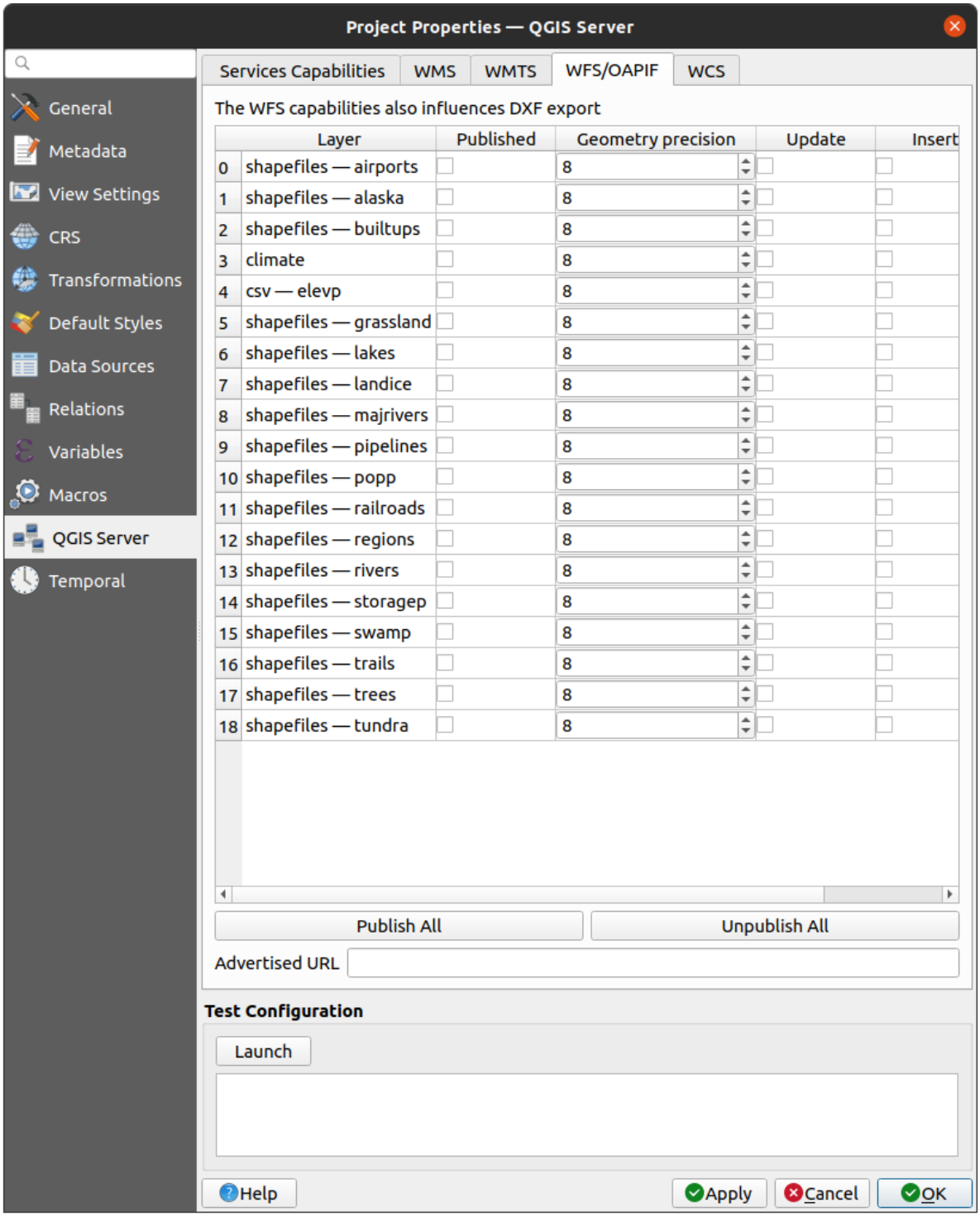


Fig. 2.5: Definitions in the WFS/OAPIF tab

If you enter a URL in the *Advertised URL* field, QGIS Server will advertise this specific URL in the WFS GetCapabilities response.

2.4.4 WCS capabilities

In the *WCS* tab, you can select the layers that you want to publish as WCS.

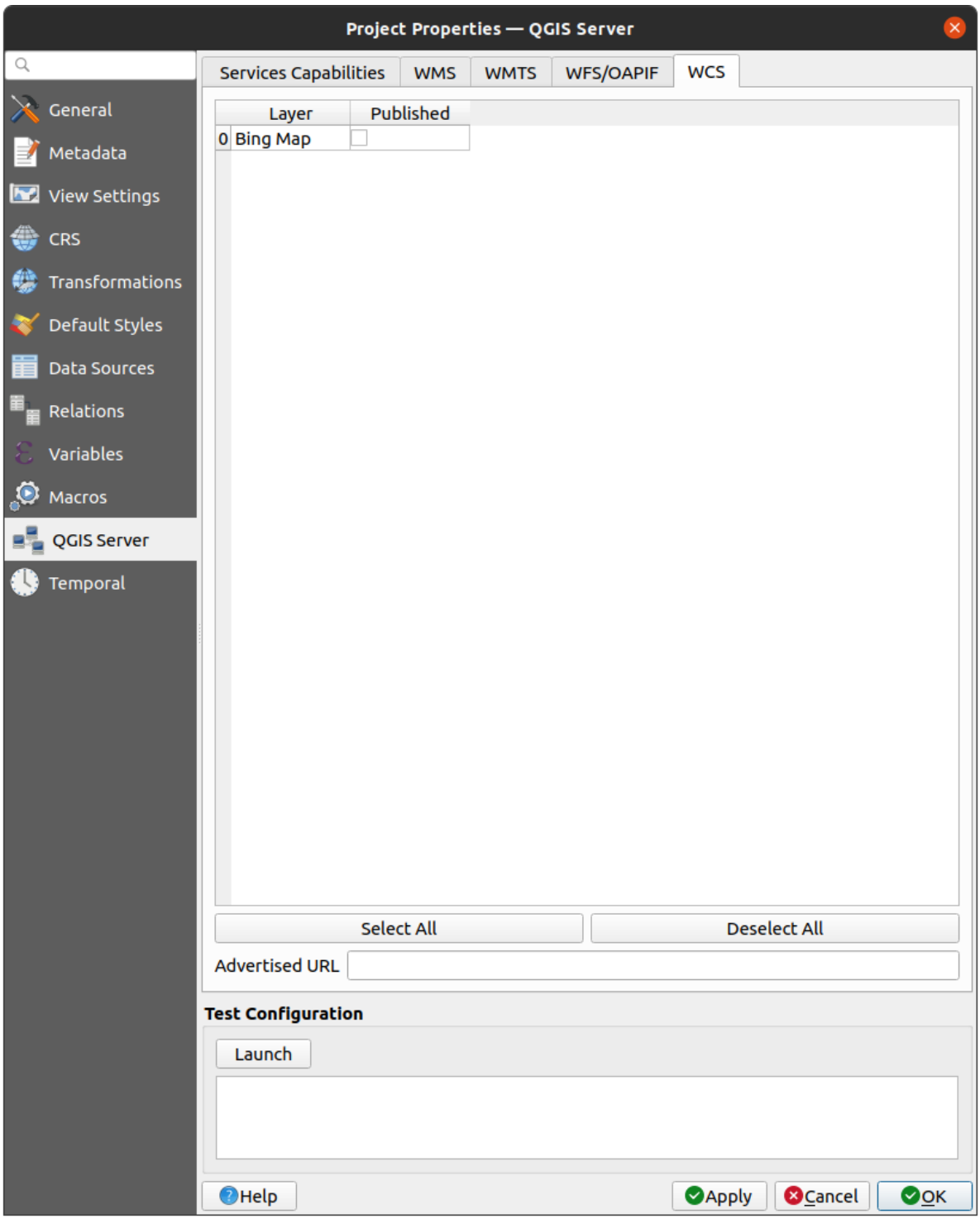


Fig. 2.6: Definitions in the WCS tab

If you enter a URL in the *Advertised URL* field of the *WCS capabilities* section, QGIS Server will advertise this specific URL in the WCS GetCapabilities response.

2.4.5 Reglarea fină a OWS-ului dvs.

For vector layers, the *Fields* tab of the *Layer* ► *Layer Properties* dialog allows you to define for each attribute if it will be published or not. By default, all the attributes are published by your WMS and WFS. If you don't want a specific attribute to be published, uncheck the appropriate checkbox in the *Configuration* column:

- *Do not expose in WFS*
- *Do not expose in WMS*

You can overlay watermarks over the maps produced by your WMS by adding text annotations or SVG annotations to the project file. See the sec_annotations section for instructions on creating annotations. For annotations to be displayed as watermarks on the WMS output, the *Fixed map position* checkbox in the *Annotation text* dialog must be unchecked. This can be accessed by double clicking the annotation while one of the annotation tools is active. For SVG annotations, you will need either to set the project to save absolute paths (in the *General* menu of the *Project* ► *Properties…* dialog) or to manually modify the path to the SVG image so that it represents a valid relative path.

2.5 Integration with third parties

QGIS Server provides standard OGC web services like WMS, WFS, etc. thus it can be used by a wide variety of end user tools.

2.5.1 Integration with QGIS Desktop

QGIS Desktop is the map designer where QGIS Server is the map server. The maps or QGIS projects will be served by the QGIS Server to provide OGC standards. These QGIS projects can either be files or entries in a database (by using *Project* ► *Save to* ► *PostgreSQL* in QGIS Desktop).

Furthermore, dedicated update workflow must be established to refresh a project used by a QGIS Server (ie. copy project files into server location and restart QGIS Server). For now, automated processes (as server reloading over message queue service) are not implemented yet.

2.5.2 Integration with MapProxy

MapProxy is a tile cache server and as it can read and serve any WMS/WMTS map server, it can be directly connected to QGIS server web services and improve end user experience.

2.5.3 Integration with QWC2

QWC2 is a responsive web application dedicated to QGIS Server. It helps you to build a highly customized map viewer with layer selection, feature info, etc.. Also many plugins are available like authentication or print service, the full list is available in this repository.

CAPITOLUL 3

Servicii

QGIS Server is able to serve data according to standard protocols as described by the **Open Geospatial Consortium (OGC)**:

- WMS 1.1.1 și 1.3.0
- WFS 1.0.0 și 1.1.0
- OGC API - Features (WFS3)
- WCS 1.0.0 și 1.1.1
- WMTS 1.0.0

Extra vendor parameters and requests are supported in addition to the original standard that greatly enhance the possibilities of customizing its behavior thanks to the QGIS rendering engine.

3.1 Basics

This section describes concepts and parameters mutually shared by services. Some of these are standard and defined in OGC specifications while others are very specific to QGIS Server.

Standard concepts:

Concept	Descriere
SERVICE	Name of the service
REQUEST	Name of the request

Vendor concepts:

Concept	Descriere
MAP	QGIS project file
FILE_NAME	File name of the downloaded file
Short name	Short name definition

3.1.1 SERVICE

This standard parameter allows to specify the name of the service to use for a specific *request* and has to be formed like `SERVICE=NAME`.

URL example for the **WMS** service:

```
http://localhost/qgisserver?
SERVICE=WMS
&...
```

> **Notă**
>
> Not available for REST based services like *WFS3 (OGC API Features)*.

3.1.2 REQUEST

This standard parameter allows to specify the name of the request to execute for a specific *service* and has to be formed like `REQUEST=RequestName`.

URL example for the **GetCapabilities** request:

```
http://localhost/qgisserver?
REQUEST=GetCapabilities
&...
```

> **Notă**
>
> Not available for REST based services like *WFS3 (OGC API Features)*.

3.1.3 MAP

This vendor parameter allows to define the QGIS project file to use. It may be an absolute path or a path relative to the location of the server executable `qgis_mapserv.fcgi`. `MAP` is mandatory by default because a request needs a QGIS project to actually work. However, the **QGIS_PROJECT_FILE** environment variable may be used to define a default QGIS project. In this specific case, `MAP` is no longer a required parameter. For further information you may refer to the *Configurare avansată* chapter.

URL example:

```
http://localhost/qgisserver?
MAP=/tmp/QGIS-Training-Data/exercise_data/qgis-server-tutorial-data/world.qgs
&...
```

3.1.4 FILE_NAME

If this vendor parameter is set, the server response will be sent to the client as a file attachment with the specified file name.

URL example to save an XML **GetCapabilities** document:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetCapabilities
```

(continues on next page)

(continuare din pagina precedentă)

```
&FILE_NAME=wms_capabilities.xml
&...
```

Notă

Not available for REST based services like *WFS3 (OGC API Features)*.

3.1.5 Short name

A number of elements have both a **short name** and a **title**. The short name is a text string used for machine-to-machine communication while the title is for the benefit of humans. For example, a dataset might have the descriptive title *"Maximum Atmospheric Temperature"* and be requested using the abbreviated short name *"ATMAX"*. You can set title, short name and abstract for:

- **Layers**: right-click on a layer and choose *Properties...* ► *QGIS Server* ► *Description*.
- **Groups**: right-click on a group and select *Set Group WMS data*
- **Project**: go to *Project* ► *Properties...* ► *QGIS Server* ► *Service Capabilities*.

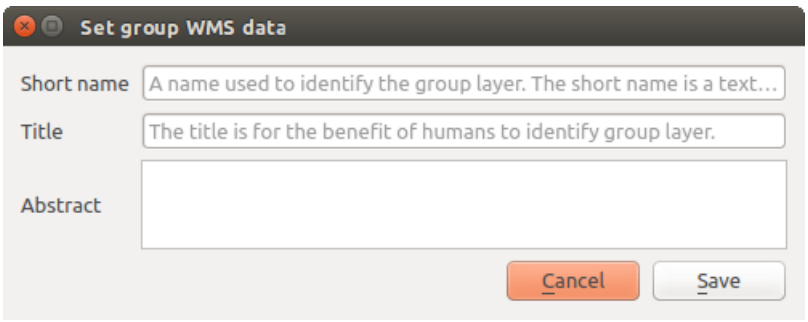


Fig. 3.1: Set group WMS data dialog

Thus, the short name may be used to identify these items when interacting with QGIS Server. For example with the standard `LAYERS` parameter:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=shortname1,shortname2
&...
```

3.2 Web Map Service (WMS)

The **1.1.1** and **1.3.0** WMS standards implemented in QGIS Server provide a HTTP interface to request map or legend images generated from a QGIS project. A typical WMS request defines the QGIS project to use, the layers to render as well as the image format to generate. Basic support is also available for **Styled Layer Descriptor (SLD)**.

Specifications:

- WMS 1.1.1
- WMS 1.3.0

- SLD 1.1.0 WMS profile

Standard requests provided by QGIS Server:

Request	Descriere
GetCapabilities	Returns XML metadata with information about the server
GetMap	Returns a map
GetFeatureInfo	Retrieves data (geometry and values) for a pixel location
GetLegendGraphic	Returns legend symbols
GetStyle(s)	Returns XML document with style description in SLD
DescribeLayer	Returns information about WFS and WCS availability respectively for vector and raster layers

Vendor requests provided by QGIS Server:

Request	Descriere
GetPrint	Returns a QGIS layout
GetProjectSettings	Returns specific information about QGIS Server
GetSchemaExtension	Returns XML metadata about optional extended capabilities

3.2.1 GetCapabilities

Standard parameters for the **GetCapabilities** request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetCapabilities**)
VERSION	Nu	Version of the service

The **GetCapabilities** request supports as well the following vendor parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetCapabilities
```

3.2.2 GetMap

Standard parameters for the **GetMap** request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetMap**)
VERSION	Da	Version of the service

continues on next page

Tabelul 3.1 – continuare din pagina precedentă

Parametru	Required	Descriere
LAYERS	Nu	Layers to display
STYLES	Nu	Layers» style
SRS / CRS	Da	Coordinate reference system
BBOX	Da	Map extent
WIDTH	Da	Width of the image in pixels
HEIGHT	Da	Height of the image in pixels
FORMAT	Nu	Image format
TRANSPARENT	Nu	Transparent background
SLD	Nu	URL of an SLD to be used for styling
SLD_BODY	Nu	In-line SLD (XML) to be used for styling

In addition to the standard ones, QGIS Server supports *redlining*, *external WMS layers* as well as the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file
BGCOLOR	Nu	Specify the background color
DPI	Nu	Specify the output resolution
IMAGE_QUALITY	Nu	JPEG compression
OPACITIES	Nu	Opacity for layer or group
FILTER	Nu	Subset of features
SELECTION	Nu	Highlight features
FILE_NAME	Nu	File name of the downloaded file Only for `FORMAT=application/dxf`
FORMAT_OPTIONS	Nu	Options of the specified file format Only for `FORMAT=application/dxf`
TILED	Nu	Working in *tiled mode*

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&MAP=/home/qgis/projects/world.qgs
&LAYERS=mylayer1,mylayer2,mylayer3
&STYLES=style1,default,style3
&OPACITIES=125,200,125
&CRS=EPSG:4326
&WIDTH=400
&HEIGHT=400
&FORMAT=image/png
&TRANSPARENT=TRUE
&DPI=300
&TILED=TRUE
```

VERSION

This parameter allows to specify the version of the service to use. Available values for the `VERSION` parameter are:

- `1.1.1`
- `1.3.0`

According to the version number, slight differences have to be expected as explained later for the next parameters:

- `CRS` / `SRS`
- `BBOX`

LAYERS

This parameter allows to specify the layers to display on the map. Names have to be separated by a comma.

In addition, QGIS Server introduced some options to select layers by:

- the layer id: the project option allowing to select layers by their id is in *QGIS Server* ► *WMS* tab of the *Project* ► *Properties...* dialog. Check the *Use layer ids as names* checkbox to activate this option.
- a *short name*

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayerid1,mylayerid2
&...
```

STYLES

This parameter can be used to specify a layer's style for the rendering step. Styles have to be separated by a comma. The name of the default style is `default`.

SRS / CRS

This parameter allows to indicate the map output Spatial Reference System in WMS **1.1.1** and has to be formed like `EPSG:XXXX`. Note that `CRS` is also supported if current version is **1.1.1**.

For WMS **1.3.0**, `CRS` parameter is preferable but `SRS` is also supported.

Note that if both `CRS` and `SRS` parameters are indicated in the request, then it's the current version indicated in `VERSION` parameter which is decisive.

In the next case, the `SRS` parameter is kept whatever the `VERSION` parameter because `CRS` is not indicated:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&SRS=EPSG:2854
&...
```

In the next case, the `SRS` parameter is kept instead of `CRS` because of the `VERSION` parameter:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.1.1
&CRS=EPSG:4326
```

(continues on next page)

(continuare din pagina precedentă)

```
&SRS=EPSG:2854
&...
```

In the next case, the `CRS` parameter is kept instead of `SRS` because of the `VERSION` parameter:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&CRS=EPSG:4326
&SRS=EPSG:2854
&...
```

BBOX

This parameter allows to specify the map extent with units according to the current CRS. Coordinates have to be separated by a comma.

The `BBOX` parameter is formed like `min_a,min_b,max_a,max_b` but `a` and `b` axis definition is different according to the current `VERSION` parameter:

- in WMS **1.1.1**, the axis ordering is always east/north
- in WMS **1.3.0**, the axis ordering depends on the CRS authority

For example in case of `EPSG:4326` and WMS **1.1.1**, `a` is the longitude (east) and `b` the latitude (north), leading to a request like:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.1.1
&SRS=epsg:4326
&BBOX=-180,-90,180,90
&...
```

But in case of WMS **1.3.0**, the axis ordering defined in the EPSG database is north/east so `a` is the latitude and `b` the longitude:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&CRS=epsg:4326
&BBOX=-90,-180,90,180
&...
```

WIDTH

This parameter allows to specify the width in pixels of the output image.

HEIGHT

This parameter allows to specify the height in pixels of the output image.

FORMAT

This parameter may be used to specify the format of map image. Available values are:

- `jpg`
- `jpeg`
- `image/jpeg`
- `image/png`
- `image/png; mode=1bit`
- `image/png; mode=8bit`
- `image/png; mode=16bit`
- `image/webp`
- `application/dxf`: only layers that have read access in the WFS service are exported in the DXF format
- `application/pdf`

  URL example:

```
http://localhost/qgisserver?
SERVICE=WMS&VERSION=1.3.0
&REQUEST=GetMap
&FORMAT=application/dxf
&LAYERS=Haltungen,Normschacht,Spezialbauwerke
&CRS=EPSG%3A21781
&BBOX=696136.28844801,245797.12108743,696318.91114315,245939.25832905
&WIDTH=1042
&HEIGHT=811
&FORMAT_OPTIONS=MODE:SYMBOLLAYERSYMBOLOGY;SCALE:250
&FILE_NAME=plan.dxf
```

TRANSPARENT

This boolean parameter can be used to specify the background transparency. Available values are (not case sensitive):

- `TRUE`
- `FALSE`

However, this parameter is ignored if the format of the image indicated with `FORMAT` is different from PNG.

BGCOLOR

This parameter allows to indicate a background color for the map image. However it cannot be combined with `TRANSPARENT` parameter in case of PNG images (transparency takes priority). The colour may be literal or in hexadecimal notation.

URL example with the literal notation:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&BGCOLOR=green
&...
```

URL example with the hexadecimal notation:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&BGCOLOR=0x00FF00
&...
```

DPI

This parameter can be used to specify the requested output resolution.

IMAGE_QUALITY

This parameter is only used for JPEG images. By default, the JPEG compression is `-1`.

You can change the default per QGIS project in the *OWS Server* ► *WMS capabilities* menu of the *Project* ► *Properties...* dialog. If you want to override it in a `GetMap` request you can do it using the `IMAGE_QUALITY` parameter.

OPACITIES

Comma separated list of opacity values. Opacity can be set on layer or group level. Allowed values range from `0` (fully transparent) to `255` (fully opaque).

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&WIDTH=400
&HEIGHT=200
&CRS=EPSG:4326
&LAYERS=countries,places
&BBOX=42,-6,52,15
&OPACITIES=255,0
```

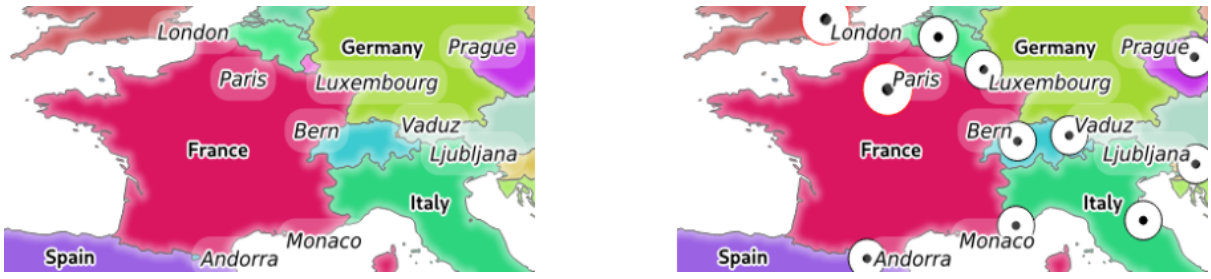


Fig. 3.2: To the left `OPACITIES=255,0` and to the right `OPACITIES=255,255`

FILTER

A subset of layers can be selected with the `FILTER` parameter. The syntax is basically the same as for the QGIS subset string. However, there are some restrictions to avoid SQL injections into databases via QGIS Server. If a dangerous string is found in the parameter, QGIS Server will return the next error:

```
<ServiceExceptionReport>
  <ServiceException code="Security">The filter string XXXXXXXXX has been rejected␣
↪because of security reasons.
  Note: Text strings have to be enclosed in single or double quotes. A space␣
↪between each word / special character is mandatory.
  Allowed Keywords and special characters are IS,NOT,NULL,AND,OR,IN,=,<,=<,>,>=,!=,
↪',',(,),DMETAPHONE,SOUNDEX.
  Not allowed are semicolons in the filter expression.</ServiceException>
</ServiceExceptionReport>
```

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&WIDTH=400
&HEIGHT=300
&CRS=EPSG:4326
&BBOX=41,-6,52,10
&LAYERS=countries_shapeburst,countries,places
&FILTER=countries_shapeburst,countries:"name" = 'France';places: "name" = 'Paris'
```

Fig. 3.3: Server response to a GetMap request with FILTER parameter

In this example, the same filter `"name" = 'France'` is applied to layers `countries` and `countries_shapeburst`, while the filter `"name" = 'Paris'` is only applied to `places`.

Notă

It is possible to make attribute searches via GetFeatureInfo and omit the X/Y parameter if a FILTER is there.

QGIS Server then returns info about the matching features and generates a combined bounding box in the XML output.

SELECTION

The `SELECTION` parameter can highlight features from one or more layers. Vector features can be selected by passing comma separated lists with feature ids.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayer1,mylayer2
&SELECTION=mylayer1:3,6,9;mylayer2:1,5,6
&...
```

The following image presents the response from a GetMap request using the `SELECTION` option e.g. `http://myserver.com/...&SELECTION=countries:171,65`.

As those features id's correspond in the source dataset to **France** and **Romania** they're highlighted in yellow.

Fig. 3.4: Server response to a GetMap request with SELECTION parameter

FORMAT_OPTIONS

This parameter can be used to specify options for the selected format. Only for `FORMAT=application/dxf` in GetMap request. Takes a list of key:value pairs separated by semicolon:

- SCALE: to be used for symbology rules, filters and styles (not actual scaling of the data - data remains in the original scale).
- MODE: corresponds to the export options offered in the QGIS Desktop DXF export dialog. Possible values are `NOSYMBOLOGY`, `FEATURESYMBOLOGY` and `SYMBOLLAYERSYMBOLOGY`.
- LAYERATTRIBUTES: specify a field or in case of many layers a comma separated list of fields that contains values for DXF layer names - if not specified, the original QGIS layer names are used.
- USE_TITLE_AS_LAYERNAME: if enabled, the title of the layer will be used as layer name.
- CODEC: specify a codec to be used for encoding. Default is `ISO-8859-1` check the QGIS desktop DXF export dialog for valid values.
- NO_MTEXT: Use TEXT instead of MTEXT for labels.
- FORCE_2D: Force 2D output. This is required for polyline width.

TILED

For performance reasons, QGIS Server can be used in tiled mode. In this mode, the client requests several small fixed size tiles, and assembles them to form the whole map. Doing this, symbols at or near the boundary between two tiles may appeared cut, because they are only present in one of the tile.

Set the `TILED` parameter to `TRUE` to tell QGIS Server to work in *tiled* mode, and to apply the *Tile buffer* configured in the QGIS project (see *Configure your project*).

When `TILED` is `TRUE` and when a non-zero Tile buffer is configured in the QGIS project, features outside the tile extent are drawn to avoid cut symbols at tile boundaries.

`TILED` defaults to `FALSE`.

3.2.3 GetFeatureInfo

Standard parameters for the **GetFeatureInfo** request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetFeatureInfo**)
VERSION	Nu	Version of the service
QUERY_LAYERS	Da	Layers to query
LAYERS	Da	Layers to display (identical to *QUERY_LAYERS*)
STYLES	Nu	Layers» style
SRS / CRS	Da	Coordinate reference system
BBOX	Nu	Map extent
WIDTH	Da	Width of the image in pixels
HEIGHT	Da	Height of the image in pixels
TRANSPARENT	Nu	Transparent background
INFO_FORMAT	Nu	Output format
FEATURE_COUNT	Nu	Maximum number of features to return
I	Nu	Pixel column of the point to query
X	Nu	Same as *I* parameter, but in WMS 1.1.1
J	Nu	Pixel row of the point to query
Y	Nu	Same as *J* parameter, but in WMS 1.1.1

continues on next page

Tabelul 3.3 – continuare din pagina precedentă

Parametru	Required	Descriere
WMS_PRECISION	Nu	The precision (number of digits) to be used when returning geometry (see *how to add geometry to feature response*). The default value is `-1` meaning that the precision defined in the project is used.

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file
FILTER	Nu	Subset of features
FI_POINT_TOLERANCE	Nu	Tolerance in pixels for point layers
FI_LINE_TOLERANCE	Nu	Tolerance in pixels for line layers
FI_POLYGON_TOLERANCE	Nu	Tolerance in pixels for polygon layers
FILTER_GEOM	Nu	Geometry filtering
WITH_DISPLAY_NAME	Nu	Add the feature display name to the output
WITH_MAPTIP	Nu	Add map tips to the output
WITH_GEOMETRY	Nu	Add geometry to the output

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&MAP=/home/qgis/projects/world.qgs
&LAYERS=mylayer1,mylayer2,mylayer3
&CRS=EPSG:4326
&WIDTH=400
&HEIGHT=400
&INFO_FORMAT=text/xml
&TRANSPARENT=TRUE
&QUERY_LAYERS=mylayer1
&FEATURE_COUNT=3
&I=250
&J=250
```

INFO_FORMAT

This parameter may be used to specify the format of the result. Available values are:

- `text/xml`
- `text/html`
- `text/plain`
- `application/vnd.ogc.gml`
- `application/json`

QUERY_LAYERS

This parameter specifies the layers to display on the map. Names are separated by a comma.

In addition, QGIS Server introduces options to select layers by:

- short name
- layer id

See the `LAYERS` parameter defined in *GetMap* for more information.

FEATURE_COUNT

This parameter specifies the maximum number of features per layer to return. For example if `QUERY_LAYERS` is set to `layer1,layer2` and `FEATURE_COUNT` is set to `3` then a maximum of 3 features from layer1 will be returned. Likewise a maximum of 3 features from layer2 will be returned.

By default, only 1 feature per layer is returned.

I

This parameter, defined in WMS 1.3.0, allows you to specify the pixel column of the query point.

X

Same parameter as `I`, but defined in WMS 1.1.1.

J

This parameter, defined in WMS 1.3.0, allows you to specify the pixel row of the query point.

Y

Same parameter as `J`, but defined in WMS 1.1.1.

FI_POINT_TOLERANCE

This parameter specifies the tolerance in pixels for point layers.

FI_LINE_TOLERANCE

This parameter specifies the tolerance in pixels for line layers.

FI_POLYGON_TOLERANCE

This parameter specifies the tolerance in pixels for polygon layers.

FILTER_GEOM

This parameter specifies a WKT geometry with which features have to intersect.

WITH_DISPLAY_NAME

This parameter specifies whether to add feature display name to the output.

Available values are (not case sensitive):

- `TRUE`
- `FALSE`

WITH_MAPTIP

This parameter specifies whether to add map tips to the output.

Available values are (not case sensitive):

- `TRUE`
- `FALSE`
- `HTML_FI_ONLY_MAPTIP`: like `TRUE`, with the difference that the HTML response to the feature info request only contains the maptip. This gives full control over the HTML response using e.g. the built-in layer maptip editor.

WITH_GEOMETRY

This parameter specifies whether to add geometries to the output. To use this feature you must first enable the *Add geometry to feature response* option in the QGIS project. See *Configure your project*.

Available values are (not case sensitive):

- `TRUE`
- `FALSE`

3.2.4 GetLegendGraphic

Standard parameters for the **GetLegendGraphic** request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetLegendGraphic**)
VERSION	Nu	Version of the service
LAYERS	Da	Layers to display
STYLES	Nu	Layers» style
SRS / CRS	Nu	Coordinate reference system
BBOX	Nu	Map extent
WIDTH	Nu	Width of the image in pixels
HEIGHT	Nu	Height of the image in pixels
FORMAT	Nu	Legend format
TRANSPARENT	Nu	Transparent background

In addition to the standard ones, QGIS Server supports extra parameters to change the size of the legend elements or the font properties for layer titles and item labels:

Parametru	Required	Descriere
MAP	Da	QGIS project file
SRCWIDTH	Nu	Map width
SRCHEIGHT	Nu	Map height
SHOWFEATURECOUNT	Nu	Add feature count of features
RULE	Nu	Rule symbol to render
RULELABEL	Nu	Item labels rendering
BOXSPACE	Nu	Space between legend frame and content (mm)
LAYERSPACE	Nu	Vertical space between layers (mm)
LAYERTITLESPACE	Nu	Vertical space between layer title and items (mm)
SYMBOLSPACE	Nu	Vertical space between symbol and items (mm)
ICONLABELSPACE	Nu	Horizontal space between symbol and label (mm)
SYMBOLWIDTH	Nu	Width of the symbol preview (mm)
SYMBOLHEIGHT	Nu	Height of the symbol preview (mm)
LAYERTITLE	Nu	Layer title rendering
LAYERFONTFAMILY	Nu	Layer font family
LAYERFONTBOLD	Nu	Layer title bold rendering
LAYERFONTSIZE	Nu	Layer title font size (pt)
LAYERFONTITALIC	Nu	Layer title italic rendering
LAYERFONTCOLOR	Nu	Layer title color
ITEMFONTFAMILY	Nu	Item font family
ITEMFONTBOLD	Nu	Item label bold rendering
ITEMFONTSIZE	Nu	Item label font size (pt)
ITEMFONTITALIC	Nu	Item label italic rendering
ITEMFONTCOLOR	Nu	Item label color
SHOWRULEDETAILS	Nu	Adds the rule text to JSON output
ADDLAYERGROUPS	Nu	Adds the layer groups to JSON output

BBOX

This parameter can be used to specify the geographical area for which the legend should be built (its format is described *here*) but cannot be combined with the `RULE` parameter. The `SRS`/`CRS` parameter becomes mandatory when using the `BBOX` parameter.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries,airports
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
```

Notă

When the `BBOX` parameter is defined, the legend is referred to as a *content based legend*.

WIDTH

This parameter is not used by default but becomes mandatory when the `RULE` parameter is set. In this case it allows to specify the width in pixels of the output image.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=getlegendgraphic
&LAYER=testlayer%20%C3%A8%C3%A9
&RULE=rule1
&WIDTH=30
&HEIGHT=30
```

HEIGHT

This parameter is not used by default but becomes mandatory when the `RULE` parameter is set. In this case it allows to specify the height in pixels of the output image.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYER=testlayer%20%C3%A8%C3%A9
&RULE=rule1
&WIDTH=30
&HEIGHT=30
```

FORMAT

This parameter may be used to specify the format of legend image. Available values are:

- `image/jpeg`
- `image/png`
- `application/json`

For JSON, symbols are encoded with Base64 and most other options related to layout or fonts are not taken into account because the legend must be built on the client side. The `RULE` parameter cannot be combined with this format.

URL example with the corresponding JSON output:

```
http://localhost/qgisserver?
SERVICE=WMS&
REQUEST=GetLegendGraphic&
LAYERS=airports&
FORMAT=application/json
```

And the corresponding JSON output:

```
{
  "nodes":[
    {
      "icon":"<base64 icon>",
      "title":"airports",
      "type":"layer"
    }
  ],
  "title":""
}
```

SRCWIDTH

This parameter may de defined when the `RULE` parameter is set. In this case, the `SRCWIDTH` value is forwarded to the underlying `GetMap` request as the `WIDTH` parameter while the `WIDTH` parameter of `GetLegendGraphic` is used for the image legend size.

SRCHEIGHT

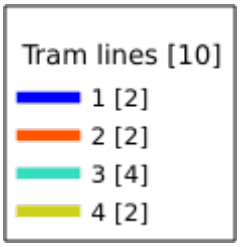
This parameter may de defined when the `RULE` parameter is set. In this case, the `SRCHEIGHT` value is forwarded to the underlying `GetMap` request as the `HEIGHT` parameter while the `HEIGHT` parameter of `GetLegendGraphic` is used for the image legend size.

SHOWFEATURECOUNT

This parameter can be used to activate feature count in the legend. Available values are (not case sensitive):

- `TRUE`
- `FALSE`

For example:



RULE

This parameter is available on layers with *Rule-based* rendering and allows to build a legend with only the named rule symbol. It cannot be combined with `BBOX` parameter. `HEIGHT` and `WIDTH` must be specified.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=mylayer,
&RULE=myrulename,
&WIDTH=20,
&HEIGHT=20
```

RULELABEL

This parameter allows to control the item label rendering. Available values are (not case sensitive):

- `TRUE`: display item label
- `FALSE`: hide item label
- `AUTO`: hide item label for layers with *Single symbol* rendering

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries,airports
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&RULELABEL=AUTO
```

Fig. 3.5: Legend rendering without label for single symbol layers

BOXSPACE

This parameter allows to specify the space between legend frame and content in millimeters. By default, the space value is 2 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&BOXSPACE=0
```

Fig. 3.6: To the left `BOXSPACE=0` and to the right `BOXSPACE=15`

LAYERSPACE

This parameter allows to specify the vertical space between layers in millimeters. By default, the space value is `3` mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERSPACE=0
```

Fig. 3.7: To the left `LAYERSPACE=0` and to the right `LAYERSPACE=10`

LAYERTITLESPACE

This parameter allows to specify the vertical space between layer title and items following in millimeters. By default the space value is `3` mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
```

(continues on next page)

(continuare din pagina precedentă)

```
&TRANSPARENT=TRUE
&LAYERTITLESPACE=0
```

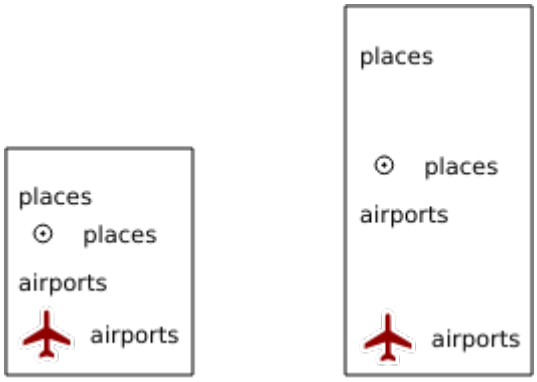


Fig. 3.8: To the left `LAYERTITLESPACE=0` and to the right `LAYERTITLESPACE=10`

SYMBOLSPACE

This parameter allows to specify the vertical space between symbol and item following in millimeters. By default the space value is `2` mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&SYMBOLSPACE=0
```

Fig. 3.9: To the left `SYMBOLSPACE=0` and to the right `SYMBOLSPACE=5`

ICONLABELSPACE

This parameter allows to specify the horizontal space between symbol and label text in millimeters. By default the space value is 2 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=getlegendgraphic
&LAYERS=countries,
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ICONLABELSPACE=0
```

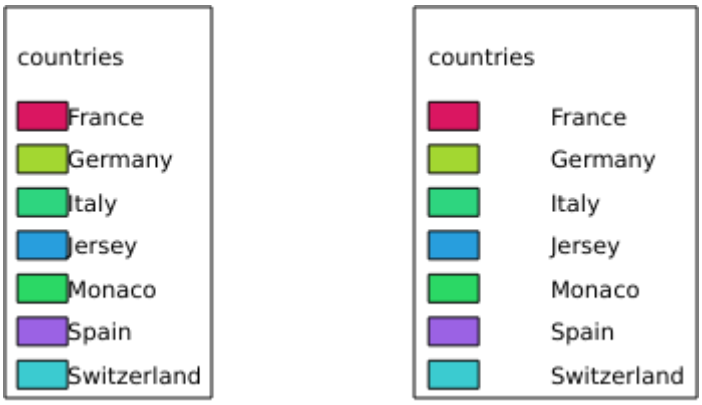


Fig. 3.10: To the left `ICONLABELSPACE=0` and to the right `ICONLABELSPACE=10`

SYMBOLWIDTH

This parameter allows to specify the width of the symbol preview in millimeters. By default the width value is 7 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries,
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&SYMBOLWIDTH=2
```

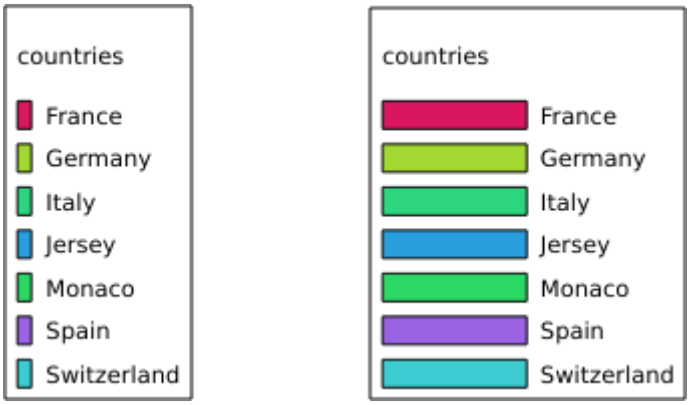


Fig. 3.11: To the left `SYMBOLWIDTH=2` and to the right `SYMBOLWIDTH=20`

SYMBOLHEIGHT

This parameter allows to specify the height of the symbol preview in millimeters. By default the height value is `4` mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries,
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&SYMBOLHEIGHT=2
```

Fig. 3.12: To the left `SYMBOLHEIGHT=2` and to the right `SYMBOLHEIGHT=6`

LAYERTITLE

This parameter specifies whether to render layer title.

Available values are (not case sensitive):

- `TRUE` (default value)
- `FALSE`

LAYERFONTFAMILY

This parameter specifies the font family to use for rendering layer title.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries
&LAYERFONTFAMILY=monospace
```

LAYERFONTBOLD

This parameter specifies whether the layer title is rendered in bold. Available values are (not case sensitive):

- `TRUE`
- `FALSE`

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTBOLD=TRUE
```

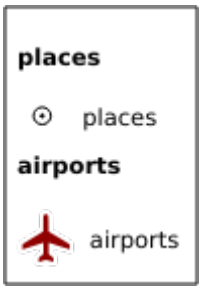


Fig. 3.13: Legendă cu `LAYERFONTBOLD=TRUE`

LAYERFONTSIZE

This parameter specifies the font size for rendering layer title in point.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTSIZE=20
```

Fig. 3.14: Legendă cu `LAYERFONTSIZE=20`

LAYERFONTITALIC

This parameter specifies whether the layer title is rendered in italic. Available values are (not case sensitive):

- `TRUE`
- `FALSE`

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTITALIC=TRUE
```

Fig. 3.15: Legendă cu `LAYERFONTITALIC=TRUE`

LAYERFONTCOLOR

This parameter specifies the layer title color. The color may be literal (`red`, `green`, ..) or in hexadecimal notation (`0xFF0000`, `0x00FF00`, ...).

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTCOLOR=0x5f9930
```

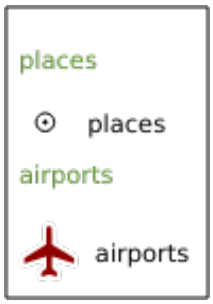


Fig. 3.16: Legendă cu `LAYERFONTCOLOR=0x5f9930`

ITEMFONTFAMILY

This parameter specifies the font family to use for rendering item label.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries
&ITEMFONTFAMILY=monospace
```

ITEMFONTBOLD

This parameter specifies whether the item label is rendered in bold. Available values are (not case sensitive):

- `TRUE`
- `FALSE`

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONTBOLD=TRUE
```

Fig. 3.17: Legendă cu `ITEMFONTBOLD=TRUE`

ITEMFONTSIZE

This parameter specifies the font size for rendering layer title in point.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONTSIZE=20
```

Fig. 3.18: Legendă cu `ITEMFONTSIZE=30`

ITEMFONTITALIC

This parameter specifies whether the item label is rendered in italic. Available values are (not case sensitive):

- `TRUE`
- `FALSE`

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONTITALIC=TRUE
```

Fig. 3.19: Legendă cu `ITEMFONTITALIC=TRUE`

ITEMFONTCOLOR

This parameter specifies the item label color. The color may be literal (`red`, `green`, ..) or in hexadecimal notation (`0xFF0000`, `0x00FF00`, ...).

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONTCOLOR=0x5f9930
```

Fig. 3.20: Legendă cu `ITEMFONTCOLOR=0x5f9930`

SHOWRULEDETAILS

This parameter specifies if the JSON output will also contain the details about the rule that generated the legend entry. This parameter only has effect when the renderer is rule-based or categorized.

URL example with the corresponding JSON output:

```
http://localhost/qgisserver?
SERVICE=WMS&
REQUEST=GetLegendGraphic&
LAYERS=airports&
FORMAT=application/json&
SHOWRULEDETAILS=TRUE
```

And the corresponding JSON output:

```
{
  "nodes":[
    {
      "icon":"<base64 icon>",
      "title":"airports",
      "type":"layer",
      "rule": "type = 'airport'"
    }
  ],
  "title":""
}
```

ADDLAYERGROUPS

This parameter specifies if the JSON output will also display the names of the layers groups (and subgroups) in the legend entry. Possible values are:

- `TRUE`: display the groups labels
- `FALSE` (default): hide the groups labels

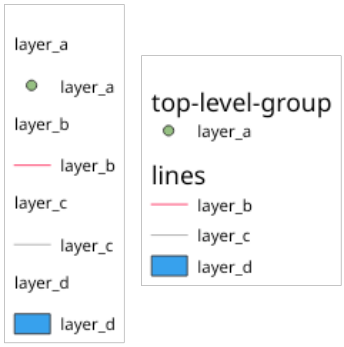


Fig. 3.21: Legend without (left) and with (right) layer groups labels display

The corresponding JSON output showing groups name would look like:

```
{ "nodes":
  [{ "nodes":
     [{ "icon":
↪"iVBORw0KGgoAAAANSUhEUgAAABQAAAAUCAYAAACNiR0NAAAACXBIWXMAABYlAAAWJQFJUiTwAAAAUklEQVQ4jWNgGAXDHz
↪37l28evBKKisr/
↪0+a7IMiNi93C15DcUpgM4wYQ5nwuZAcMIQNvHv3LuO83C0kG0hysoFZRrJNo2AYAQC87BpkGQj1fwAAAABJRU5ErkJggg==
↪",
        "title":"layer_a", "type":"layer"
     },
     { "nodes":
       [{ "icon":
↪"iVBORw0KGgoAAAANSUhEUgAAABMAAAAUCAYAAABvVQZ0AAAACXBIWXMAABYlAAAWJQFJUiTwAAAAHUlEQVQ4jWNgGAWjYC
↪",
          "title":"layer_b", "type":"layer"
        },
        { "icon":
↪"iVBORw0KGgoAAAANSUhEUgAAABMAAAAUCAYAAABvVQZ0AAAACXBIWXMAABYlAAAWJQFJUiTwAAAAHUlEQVQ4jWNgGAWjYC
↪8aaEeMglEwYAAAaIoCzTtn5XoAAAAASUVORK5CYII=",
          "title":"layer_c", "type":"layer"
       }],
       "title":"lines", "type":"group"
     },
```

(continues on next page)

(continuare din pagina precedentă)

```
    { "icon":
↪"iVBORw0KGgoAAAANSUhEUgAAABMAAAATCAYAAAByUDbMAAAACXBIWXMAABYlAAAWJQFJUiTwAAAAKklEQVQ4jWNUVlX/
↪z0AlwMLAwMDAnXmQKoYxUcWUUcNGDRs1bNSwYWYYACXDAsvQaTuVAAAAAElFTkSuQmCC",
      "title":"layer_d", "type":"layer"
    }],
    "title":"top-level-group", "type":"group"
  }],
  "title":""
}
```

3.2.5 GetStyle(s)

Standard parameters for the **GetStyle** (or **GetStyles**) request according to the OGC WMS 1.1.1 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetStyle** or **GetStyles**)
LAYERS	Da	Layers to query

The **GetStyle** request supports as well the following vendor parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetStyles
&LAYERS=mylayer1,mylayer2
```

3.2.6 DescribeLayer

Standard parameters for the **DescribeLayer** request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**DescribeLayer**)
LAYERS	Da	Layers to describe
SLD_VERSION	Da	SLD version

The **DescribeLayer** request supports as well the following vendor parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=DescribeLayer
&SLD_VERSION=1.1.0
&LAYERS=mylayer1
```

The XML document looks like:

```
<DescribeLayerResponse xmlns="http://www.opengis.net/sld" xmlns:xsi="http://www.w3.
↪org/2001/XMLSchema-instance" xmlns:ows="http://www.opengis.net/ows" xmlns:xlink=
↪"http://www.w3.org/1999/xlink" xmlns:se="http://www.opengis.net/se"␣
↪xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.net/sld/1.
↪1.0/DescribeLayer.xsd">
    <Version>1.1.0</Version>
    <LayerDescription>
        <owsType>wfs</owsType>
        <se:OnlineResource xlink:href="http://localhost/qgisserver" xlink:type=
↪"simple"/>
        <TypeName>
            <se:FeatureTypeName>my_vector_layer</se:FeatureTypeName>
        </TypeName>
    </LayerDescription>
    <LayerDescription>
        <owsType>wcs</owsType>
        <se:OnlineResource xlink:href="http://localhost/qgisserver" xlink:type=
↪"simple"/>
        <TypeName>
            <se:FeatureTypeName>my_raster_layer</se:FeatureTypeName>
        </TypeName>
    </LayerDescription>
</DescribeLayerResponse>
```

SLD_VERSION

This parameter allows to specify the version of SLD. Only the value `1.1.0` is available.

3.2.7 GetPrint

QGIS Server has the capability to create print layout output in pdf or pixel format. Print layout windows in the published project are used as templates. In the **GetPrint** request, the client has the possibility to specify parameters of the contained layout maps and labels.

The **GetPrint** request supports *redlining*, *external WMS layers* as well as the following parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetPrint**)
VERSION	Nu	Version of the service
LAYERS	Nu	Layers to display
TEMPLATE	Da	Layout template to use
SRS / CRS	Da	Coordinate reference system
FORMAT	Nu	Output format
FORMAT_OPTIONS	Nu	Options of the specified file format Only for `FORMAT=application/pdf`
ATLAS_PK	Nu	Atlas features

continues on next page

Tabelul 3.4 – continuare din pagina precedentă

Parametru	Required	Descriere
STYLES	Nu	Layers» style
TRANSPARENT	Nu	Transparent background
OPACITIES	Nu	Opacity for layer or group
SELECTION	Nu	Highlight features
mapX:EXTENT	Nu	Extent of the map «X»
mapX:LAYERS	Nu	Layers of the map «X»
mapX:STYLES	Nu	Layers» style of the map «X»
mapX:SCALE	Nu	Layers» scale of the map «X»
mapX:ROTATION	Nu	Rotation of the map «X»
mapX:GRID_INTERVAL_X	Nu	Grid interval on x axis of the map «X»
mapX:GRID_INTERVAL_Y	Nu	Grid interval on y axis of the map «X»

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetPrint
&MAP=/home/qgis/projects/world.qgs
&CRS=EPSG:4326
&FORMAT=png
&TEMPLATE=Layout%201
&map0:EXTENT=-180,-90,180,90
&map0:LAYERS=mylayer1,mylayer2,mylayer3
&map0:OPACITIES=125,200,125
&map0:ROTATION=45
```

Note that the layout template may contain more than one map. In this way, if you want to configure a specific map, you have to use `mapX:` parameters where `X` is a positive number that you can retrieve thanks to the **GetProjectSettings** request.

For example:

```
<WMS_Capabilities>
...
<ComposerTemplates xsi:type="wms:_ExtendedCapabilities">
<ComposerTemplate width="297" height="210" name="Druckzusammenstellung 1">
<ComposerMap width="171" height="133" name="map0"/>
<ComposerMap width="49" height="46" name="map1"/></ComposerTemplate>
</ComposerTemplates>
...
</WMS_Capabilities>
```

TEMPLATE

This parameter can be used to specify the name of a layout template to use for printing.

FORMAT

This parameter specifies the format of map image. Available values are:

- `png` (default value)
- `image/png`
- `jpg`
- `jpeg`
- `image/jpeg`
- `svg`
- `image/svg`
- `image/svg+xml`
- `pdf`
- `application/pdf`

If the `FORMAT` parameter is different from one of these values, then an exception is returned.

FORMAT_OPTIONS

This parameter can be used to specify options for the selected format. Only for `FORMAT=application/pdf` in GetPrint requests. Takes a list of key:value pairs separated by semicolon:

- `RASTERIZE_WHOLE_IMAGE`: whether the whole pdf should be exported as an image. Default: false.
- `FORCE_VECTOR_OUTPUT`: whether pdf should be exported as vector. Default: false.
- `APPEND_GEOREFERENCE`: whether georeference info shall be added to the pdf. Default: true.
- `EXPORT_METADATA`: whether metadata shall be added to the pdf. Default: true.
- `TEXT_RENDER_FORMAT`: sets the text render format for pdf export. It can be `AlwaysOutlines` (default) or `AlwaysText`.
- `SIMPLIFY_GEOMETRY`: whether features geometries shall be simplified. Default: true.
- `WRITE_GEO_PDF`: whether a Geospatial PDF shall be exported. Default: false.
- `USE_ISO_32000_EXTENSION_FORMAT_GEOREFERENCING`: whether Iso32000 georeferencing shall be used. Default: false.
- `USE_OGC_BEST_PRACTICE_FORMAT_GEOREFERENCING`: whether OGC best practice georeferencing shall be used. Default: false.
- `EXPORT_THEMES`: a comma separated list of map themes to use for a Geospatial PDF export
- `PREDEFINED_MAP_SCALES`: a comma separated list of map scales to render the map
- `LOSSLESS_IMAGE_COMPRESSION`: whether images embedded in pdf must be compressed using a lossless algorithm. Default: false.
- `DISABLE_TILED_RASTER_RENDERING`: whether rasters shall be untiled in the pdf. Default: false.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetPrint
&MAP=/home/qgis/projects/world.qgs
&CRS=EPSG:4326
&FORMAT=pdf
&TEMPLATE=Layout%201
&FORMAT_OPTIONS=FORCE_VECTOR_OUTPUT:TRUE;TEXT_RENDER_FORMAT:AlwaysOutlines;
↪PREDEFINED_MAP_SCALES:250
```

ATLAS_PK

This parameter allows activation of Atlas rendering by indicating which features we want to print. In order to retrieve an atlas with all features, the `*` symbol may be used (according to the maximum number of features allowed in the project configuration).

When `FORMAT` is `pdf`, a single PDF document combining the feature pages is returned. For all other formats, a single page is returned.

mapX:EXTENT

This parameter specifies the extent for a layout map item as xmin,ymin,xmax,ymax.

mapX:ROTATION

This parameter specifies the map rotation in degrees.

mapX:GRID_INTERVAL_X

This parameter specifies the grid line density in the X direction.

mapX:GRID_INTERVAL_Y

This parameter specifies the grid line density in the Y direction.

mapX:SCALE

This parameter specifies the map scale for a layout map item. This is useful to ensure scale based visibility of layers and labels even if client and server may have different algorithms to calculate the scale denominator.

mapX:LAYERS

This parameter specifies the layers for a layout map item. See *GetMap Layers* for more information on this parameter.

mapX:STYLES

This parameter specifies the layers» styles defined in a specific layout map item. See *GetMap Styles* for more information on this parameter.

3.2.8 GetProjectSettings

This request type works similar to *GetCapabilities*, but it is more specific to QGIS Server and allows a client to read additional information which are not available in the *GetCapabilities* output:

- vizibilitatea inițială a straturilor
- information about vector attributes and their edit types
- information about layer order and drawing order
- list of layers published in WFS
- show if a group in the layer tree is mutually exclusive

The **GetProjectSettings** request supports the following parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetProjectSettings**)

3.2.9 GetSchemaExtension

The **GetSchemaExtension** request allows to retrieve optional extended capabilities and operations of the WMS service such as implemented by QGIS Server.

Parametru	Required	Descriere
MAP	Da	QGIS project file
SERVICE	Da	Name of the service (**WMS**)
REQUEST	Da	Name of the request (**GetSchemaExtension**)

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetSchemaExtension
```

The XML document looks like:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:wms="http://www.opengis.net/
↪wms" xmlns:qgs="http://www.qgis.org/wms" targetNamespace="http://www.qgis.org/wms
↪" elementFormDefault="qualified" version="1.0.0">
    <import namespace="http://www.opengis.net/wms" schemaLocation="http://schemas.
↪opengis.net/wms/1.3.0/capabilities_1_3_0.xsd"/>
    <element name="GetPrint" type="wms:OperationType" substitutionGroup="wms:_
↪ExtendedOperation"/>
    <element name="GetStyles" type="wms:OperationType" substitutionGroup="wms:_
↪ExtendedOperation"/>
</schema>
```

3.2.10 External WMS layers

QGIS Server allows including layers from external WMS servers in WMS *GetMap* and WMS *GetPrint* requests. This is especially useful if a web client uses an external background layer in the web map. For performance reasons, such layers should be directly requested by the web client (not cascaded via QGIS server). For printing however, these layers should be cascaded via QGIS server in order to appear in the printed map.

External layers can be added to the LAYERS parameter as EXTERNAL_WMS:<layername>. The parameters for the external WMS layers (e.g. url, format, dpiMode, crs, layers, styles) can later be given as service parameters <layername>:<parameter>. In a *GetMap* request, this might look like this:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=EXTERNAL_WMS:basemap,layer1,layer2
&OPACITIES=100,200,255
&STYLES=,,
&basemap:url=http://externalserver.com/wms.fcgi
&basemap:format=image/jpeg
&basemap:dpiMode=7
&basemap:crs=EPSG:2056
&basemap:layers=orthofoto
&basemap:styles=default
```

Similarly, external layers can be used in *GetPrint* requests:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetPrint
&TEMPLATE=A4
&map0:layers=EXTERNAL_WMS:basemap,layer1,layer2
&map0:EXTENT=<minx,miny,maxx,maxy>
&OPACITIES=100,200,255
&basemap:url=http://externalserver.com/wms.fcgi
&basemap:format=image/jpeg
&basemap:dpiMode=7
&basemap:crs=EPSG:2056
&basemap:layers=orthofoto
&basemap:styles=default
```

3.2.11 Redlining

This feature is available and can be used with *GetMap* and *GetPrint* requests.

The redlining feature can be used to pass geometries and labels in the request which are overlapped by the server over the standard returned image (map). This permits the user to put emphasis or maybe add some comments (labels) to some areas, locations etc. that are not in the standard map.

The *GetMap* request is in the format:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&HIGHLIGHT_GEOM=POLYGON((590000 5647000, 590000 6110620, 2500000 6110620, 2500000␣
↪5647000, 590000 5647000))
&HIGHLIGHT_SYMBOL=<StyledLayerDescriptor><UserStyle><Name>Highlight</Name>
↪<FeatureTypeStyle><Rule><Name>Symbol</Name><LineSymbolizer><Stroke><SvgParameter␣
↪name="stroke">%23ea1173</SvgParameter><SvgParameter name="stroke-opacity">1</
↪SvgParameter><SvgParameter name="stroke-width">1.6</SvgParameter></Stroke></
↪LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></StyledLayerDescriptor>
```

(continues on next page)

(continuare din pagina precedentă)

```
&HIGHLIGHT_LABELSTRING=Write label here
&HIGHLIGHT_LABELSIZE=16
&HIGHLIGHT_LABELCOLOR=%23000000
&HIGHLIGHT_LABELBUFFERCOLOR=%23FFFFFF
&HIGHLIGHT_LABELBUFFERSIZE=1.5
```

The *GetPrint* equivalent is in the format (note that `mapX:` parameter is added to tell which map has redlining):

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetPrint
&map0:HIGHLIGHT_GEOM=POLYGON((590000 5647000, 590000 6110620, 2500000 6110620,␣
↪2500000 5647000, 590000 5647000))
&map0:HIGHLIGHT_SYMBOL=<StyledLayerDescriptor><UserStyle><Name>Highlight</Name>
↪<FeatureTypeStyle><Rule><Name>Symbol</Name><LineSymbolizer><Stroke><SvgParameter␣
↪name="stroke">%23ea1173</SvgParameter><SvgParameter name="stroke-opacity">1</
↪SvgParameter><SvgParameter name="stroke-width">1.6</SvgParameter></Stroke></
↪LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></StyledLayerDescriptor>
&map0:HIGHLIGHT_LABELSTRING=Write label here
&map0:HIGHLIGHT_LABELSIZE=16
&map0:HIGHLIGHT_LABELCOLOR=%23000000
&map0:HIGHLIGHT_LABELBUFFERCOLOR=%23FFFFFF
&map0:HIGHLIGHT_LABELBUFFERSIZE=1.5
```

Here is the image outputed by the above request in which a polygon and a label are drawn on top of the normal map:

Fig. 3.22: Server response to a GetMap request with redlining parameters

You can see there are several parameters in this request to control the redlining feature. The full list includes:

- **HIGHLIGHT_GEOM**: You can add POINT, MULTILINESTRING, POLYGON etc. It supports multipart geometries. Here is an example: `HIGHLIGHT_GEOM=MULTILINESTRING((0 0, 0 1, 1 1))`. The coordinates should be in the CRS of the GetMap/GetPrint request.
- **HIGHLIGHT_LABELBUFFERCOLOR**: This parameter controls the label buffer color.
- **HIGHLIGHT_LABELBUFFERSIZE**: This parameter controls the label buffer size.
- **HIGHLIGHT_LABELCOLOR**: This parameter controls the label color.
- **HIGHLIGHT_LABEL_DISTANCE**: controls the distance between feature (e.g. point or line) and the label in mm
- **HIGHLIGHT_LABELFONT**: This parameter controls the font of the label (e.g. Arial)
- **HIGHLIGHT_LABEL_HORIZONTAL_ALIGNMENT**: places the label horizontally on a point using the specified alignment (e.g. «left», «center», «right»)
- **HIGHLIGHT_LABEL_ROTATION**: controls the label rotation in degrees
- **HIGHLIGHT_LABELSIZE**: This parameter controls the size of the label.
- **HIGHLIGHT_LABELSTRING**: You can pass your labeling text to this parameter.
- **HIGHLIGHT_LABEL_VERTICAL_ALIGNMENT**: places the label vertically on a point using the specified alignment (e.g. «top», «half», «bottom»)

- **HIGHLIGHT_SYMBOL**: This controls how the geometry is outlined and you can change the stroke width, color and opacity.

3.3 Web Feature Service (WFS)

The **1.0.0** and **1.1.0** WFS standards implemented in QGIS Server provide a HTTP interface to query geographic features from a QGIS project. A typical WFS request defines the QGIS project to use and the layer to query.

Specifications document according to the version number of the service:

- WFS 1.0.0
- WFS 1.1.0

Standard requests provided by QGIS Server:

Request	Descriere
GetCapabilities	Returns XML metadata with information about the server
GetFeature	Returns a selection of features
DescribeFeatureType	Returns a description of feature types and properties
Transaction	Allows features to be inserted, updated or deleted

3.3.1 GetCapabilities

Standard parameters for the **GetCapabilities** request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WFS**)
REQUEST	Da	Name of the request (**GetCapabilities**)
VERSION	Nu	Version of the service

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

VERSION

This parameter allows to specify the version of the service to use. Available values for the `VERSION` parameter are:

- `1.0.0`
- `1.1.0`

If no version is indicated in the request, then `1.1.0` is used by default.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&...
```

3.3.2 GetFeature

Standard parameters for the **GetFeature** request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WFS**)
REQUEST	Da	Name of the request (**GetFeature**)
VERSION	Nu	Version of the service
TYPENAME	Nu	Name of layers
FEATUREID	Nu	Filter the features by ids
OUTPUTFORMAT	Nu	Output Format
RESULTTYPE	Nu	Type of the result
PROPERTYNAME	Nu	Name of properties to return
MAXFEATURES	Nu	Maximum number of features to return
SRSNAME	Nu	Coordinate reference system
FILTER	Nu	OGC Filter Encoding
BBOX	Nu	Map Extent
SORTBY	Nu	Sort the results

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file
STARTINDEX	Nu	Paging
GEOMETRYNAME	Nu	Type of geometry to return
EXP_FILTER	Nu	Expression filtering

TYPENAME

This parameter allows to specify layer names and is mandatory if `FEATUREID` is not set.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&TYPENAME=countries
```

FEATUREID

This parameter allows to specify the ID of a specific feature and is formed like `typename.fid,typename.fid,...`.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&FEATUREID=countries.0,places.1
```

XML response:

```
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://
↪www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://
↪www.opengis.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:qgs="http:/
↪/www.qgis.org/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"␣
↪xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.
↪1.0/wfs.xsd http://www.qgis.org/gml http://192.168.1.15/qgisserver?SERVICE=WFS&
↪VERSION=1.1.0&REQUEST=DescribeFeatureType&TYPENAME=countries,places&
↪OUTPUTFORMAT=text/xml; subtype%3Dgml/3.1.1">
  <gml:boundedBy>
    ...
  </gml:boundedBy>
  <gml:featureMember>
    <qgs:countries gml:id="countries.1">
      ...
    </qgs:countries>
  </gml:featureMember>
  <gml:featureMember>
    <qgs:places gml:id="places.1">
      ...
    </qgs:places>
  </gml:featureMember>
</wfs:FeatureCollection>
```

OUTPUTFORMAT

This parameter may be used to specify the format of the response. If `VERSION` is greater or equal than `1.1.0`, GML3 is the default format. Otherwise GML2 is used.

Available values are:

- `gml2`
- `text/xml; subtype=gml/2.1.2`
- `gml3`
- `text/xml; subtype=gml/3.1.1`
- `geojson`
- `application/vnd.geo+json`,
- `application/vnd.geo json`
- `application/geo+json`
- `application/geo json`
- `application/json`

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&FEATUREID=countries.0
&OUTPUTFORMAT=geojson
```

GeoJSON response:

```
{
    "type":"FeatureCollection",
    "bbox":[
        -180,
```

(continues on next page)

(continuare din pagina precedentă)


```
        -90,
        180,
        83.6236
    ],
    "features":[
        {
            "bbox":[
                -61.891113,
                16.989719,
                -61.666389,
                17.724998
            ],
            "geometry":{
                "coordinates":[
                    "..."
                ],
                "type":"MultiPolygon"
            },
            "id":"countries.1",
            "properties":{
                "id":1,
                "name":"Antigua and Barbuda"
            },
            "type":"Feature"
        }
    ]
}
```


RESULTTYPE

This parameter may be used to specify the kind of result to return. Available values are:

- `results`: the default behavior
- `hits`: returns only a feature count

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&RESULTTYPE=hits
&...
```

PROPERTYNAME

This parameter may be used to specify a specific property to return. A property needs to be mapped with a `TYPENAME` or a `FEATUREID`:

Valid URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&PROPERTYNAME=name
&TYPENAME=places
```

On the contrary, the next URL will return an exception:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&PROPERTYNAME=name
&TYPENAME=places,countries
```

```
<ServiceExceptionReport xmlns="http://www.opengis.net/ogc" version="1.2.0">
    <ServiceException code="RequestNotWellFormed">There has to be a 1:1 mapping␣
↪between each element in a TYPENAME and the PROPERTYNAME list</ServiceException>
</ServiceExceptionReport>
```

MAXFEATURES

This parameter allows to limit the number of features returned by the request.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places
&MAXFEATURES=1000
```

Notă

This parameter may be useful to improve performances when underlying vector layers are heavy.

SRSNAME

This parameter allows to indicate the response output Spatial Reference System as well as the `BBOX` CRS and has to be formed like `EPSG:XXXX`.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places
&SRSNAME=EPSG:32620
```

FILTER

This parameter allows to filter the response with the **Filter Encoding** language defined by the OGC Filter Encoding standard. For example:

```
http://localhost/qgisserver?
SERVICE=WFS&
REQUEST=GetFeature&
TYPENAME=places&
FILTER=<Filter><PropertyIsEqualTo><PropertyName>name</PropertyName><Literal>Paris</
↪Literal></PropertyIsEqualTo></Filter>
```

In case of multiple typenames, filters have to be enclosed in parentheses:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places,countries
&FILTER=(<Filter><PropertyIsEqualTo><PropertyName>name</PropertyName><Literal>Paris
↪</Literal></PropertyIsEqualTo></Filter>)(<Filter><PropertyIsEqualTo>
↪<PropertyName>name</PropertyName><Literal>France</Literal></PropertyIsEqualTo></
↪Filter>)
```

Filter features that intersect with a polygon:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&VERSION=1.1.0
&TYPENAME=places
&FILTER=<Filter xmlns="http://www.opengis.net/ogc">
            <Intersects>
                <PropertyName>geometry</PropertyName>
                <Polygon xmlns="http://www.opengis.net/gml" srsName="EPSG:4326">
                    <exterior>
                        <LinearRing>
                            <posList>
                                -0.6389 42.5922
                                10.2683 51.9106
                                14.5196 41.0320
                                -0.6389 42.5922
                            </posList>
                        </LinearRing>
                    </exterior>
                </Polygon>
            </Intersects>
        </Filter>
```

BBOX

This parameter allows to specify the map extent with units according to the current CRS. Coordinates have to be separated by a comma.

The `SRSNAME` parameter may specify the CRS of the extent. If not specified, the CRS of the layer is used.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places
&BBOX=-11.84,42.53,8.46,50.98
```

The `FEATUREID` parameter cannot be used with the `BBOX`. Any attempt will result in an exception:

```
<ServiceExceptionReport xmlns="http://www.opengis.net/ogc" version="1.2.0">
  <ServiceException code="RequestNotWellFormed">FEATUREID FILTER and BBOX␣
↪parameters are mutually exclusive</ServiceException>
</ServiceExceptionReport>
```

SORTBY

This parameter allows to sort resulting features according to property values and has to be formed like `propertyname SORTRULE`.

Available values for `SORTRULE` in case of descending sorting:

- `D`
- `+D`
- `DESC`
- `+DESC`

Available values for `SORTRULE` in case of ascending sorting:

- `A`
- `+A`
- `ASC`
- `+ASC`

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places
&PROPERTYNAME=name
&MAXFEATURES=3
&SORTBY=name DESC
```

The corresponding result:

```
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://
↪www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://
↪www.opengis.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:qgs="http:/
↪/www.qgis.org/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"␣
↪xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.
↪1.0/wfs.xsd http://www.qgis.org/gml http://192.168.1.15/qgisserver?SERVICE=WFS&
↪VERSION=1.1.0&REQUEST=DescribeFeatureType&TYPENAME=places&OUTPUTFORMAT=text/xml;␣
↪subtype%3Dgml/3.1.1">
    <gml:boundedBy>
        ...
    </gml:boundedBy>
    <gml:featureMember>
        <qgs:places gml:id="places.90">
            <qgs:name>Zagreb</qgs:name>
        </qgs:places>
    </gml:featureMember>
    <gml:featureMember>
        <qgs:places gml:id="places.113">
            <qgs:name>Yerevan</qgs:name>
        </qgs:places>
    </gml:featureMember>
    <gml:featureMember>
        <qgs:places gml:id="places.111">
            <qgs:name>Yaounde</qgs:name>
        </qgs:places>
    </gml:featureMember>
</wfs:FeatureCollection>
```

GEOMETRYNAME

Acest parametru poate fi utilizat pentru a specifica tipul de geometrie de returnat de către entități. Valorile disponibile sunt:

- `extent`
- `centroid`
- `none`

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&GEOMETRYNAME=centroid
&...
```

STARTINDEX

This parameter is standard in WFS 2.0, but it's an extension for WFS 1.0.0.

Actually, it can be used to skip some features in the result set and in combination with `MAXFEATURES`, it provides the ability to page through results.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&STARTINDEX=2
&...
```

EXP_FILTER

This parameter allows to filter the response with QGIS expressions. The `;` character is used to separate filters in case of multiple typenames.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places,countries
&EXP_FILTER="name"='Paris';"name"='France'
```

3.3.3 DescribeFeatureType

Standard parameters for the **DescribeFeatureType** request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WFS**)
REQUEST	Da	Name of the request (**DescribeFeatureType**)
VERSION	Nu	Version of the service
OUTPUTFORMAT	Nu	Format of the response
TYPENAME	Nu	Name of layers

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=DescribeFeatureType
&TYPENAME=countries
```

Output response:

```
<schema xmlns:ogc="http://www.opengis.net/ogc" xmlns:xsd="http://www.w3.org/2001/
↪XMLSchema" xmlns="http://www.w3.org/2001/XMLSchema" xmlns:qgs="http://www.qgis.
↪org/gml" xmlns:gml="http://www.opengis.net/gml" targetNamespace="http://www.qgis.
↪org/gml" version="1.0" elementFormDefault="qualified">
  <import schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"␣
↪namespace="http://www.opengis.net/gml"/>
  <element type="qgs:countriesType" substitutionGroup="gml:_Feature" name=
↪"countries"/>
  <complexType name="countriesType">
    <complexContent>
      <extension base="gml:AbstractFeatureType">
        <sequence>
          <element minOccurs="0" type="gml:MultiPolygonPropertyType" maxOccurs="1"␣
↪name="geometry"/>
          <element type="long" name="id"/>
          <element nillable="true" type="string" name="name"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</schema>
```

3.3.4 Tranzacție

This request allows to update, delete or add one or several features thanks to a XML document. The *delete* action may be achieved with a POST request as well as with the *OPERATION* parameter while the *add* and the *update* operations may be achieved through POST request only.

Standard parameters for the **Transaction** request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WFS**)
REQUEST	Da	Name of the request (**Transaction**)
VERSION	Nu	Version of the service
FILTER	Nu	OGC Filter Encoding
BBOX	Nu	Map Extent
FEATUREID	Nu	Filter the features by ids
TYPENAME	Nu	Name of layers

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file
OPERATION	Nu	Specify the operation
EXP_FILTER	Nu	Expression filtering

OPERATION

This parameter allows to delete a feature without using a POST request with a dedicated XML document.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=Transaction
&OPERATION=DELETE
&FEATUREID=24
```

Notă

`FEATUREID`, `BBOX` and `FILTER` parameters are mutually exclusive and prioritized in this order.

Add features

POST request example:

```
wget --post-file=add.xml "http://localhost/qgisserver?SERVICE=WFS&
↪REQUEST=Transaction"
```

with the *add.xml* document:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:Transaction service="WFS" version="1.0.0" xmlns:wfs="http://www.opengis.net/
↪wfs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ogc="http://www.
↪opengis.net/ogc" xmlns="http://www.opengis.net/wfs" updateSequence="0"␣
↪xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.
↪opengis.net/wfs http://schemas.opengis.net/wfs/1.0.0/WFS-capabilities.xsd"␣
↪xmlns:gml="http://www.opengis.net/gml"  xmlns:ows="http://www.opengis.net/ows">
  <wfs:Insert idgen="GenerateNew">
    <qgs:places>
      <qgs:geometry>
        <gml:Point srsDimension="2" srsName="http://www.opengis.net/def/crs/EPSG/0/
↪4326">
```

(continues on next page)

(continuare din pagina precedentă)

```
          <gml:coordinates decimal="." cs="," ts=" ">-4.6167,48.3833</
↪gml:coordinates>
        </gml:Point>
      </qgs:geometry>
      <qgs:name>Locmaria-Plouzané</qgs:name>
    </qgs:places>
  </wfs:Insert>
</wfs:Transaction>
```

Update features

POST request example:

```
wget --post-file=update.xml "http://localhost/qgisserver?SERVICE=WFS&
↪REQUEST=Transaction"
```

with the *update.xml* document:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:Transaction service="WFS" version="1.0.0" xmlns:wfs="http://www.opengis.net/
↪wfs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ogc="http://www.
↪opengis.net/ogc" xmlns="http://www.opengis.net/wfs" updateSequence="0"␣
↪xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.
↪opengis.net/wfs http://schemas.opengis.net/wfs/1.0.0/WFS-capabilities.xsd"␣
↪xmlns:gml="http://www.opengis.net/gml"  xmlns:ows="http://www.opengis.net/ows">
    <wfs:Update typeName="places">
      <wfs:Property>
          <wfs:Name>name</wfs:Name>
          <wfs:Value>Lutece</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
          <ogc:FeatureId fid="24"/>
      </ogc:Filter>
    </wfs:Update>
</wfs:Transaction>
```

Delete features

POST request example:

```
wget --post-file=delete.xml "http://localhost/qgisserver?SERVICE=WFS&
↪REQUEST=Transaction"
```

with the *delete.xml* document:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:Transaction service="WFS" version="1.0.0" xmlns:wfs="http://www.opengis.net/
↪wfs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ogc="http://www.
↪opengis.net/ogc" xmlns="http://www.opengis.net/wfs" updateSequence="0"␣
↪xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.
↪opengis.net/wfs http://schemas.opengis.net/wfs/1.0.0/WFS-capabilities.xsd"␣
↪xmlns:gml="http://www.opengis.net/gml"  xmlns:ows="http://www.opengis.net/ows">
    <wfs:Delete typeName="places">
        <ogc:Filter>
            <ogc:FeatureId fid="24"/>
        </ogc:Filter>
    </wfs:Delete>
</wfs:Transaction>
```

3.4 Web Coverage Service (WCS)

The **1.0.0** and **1.1.1** WCS standards implemented in QGIS Server provide a HTTP interface to access raster data, referred to as *coverage*, coming from a QGIS project.

Specifications:

- WCS 1.0.0
- WCS 1.1.1

Standard requests provided by QGIS Server:

Request	Descriere
GetCapabilities	Returns XML metadata with information about the server
DescribeCoverage	Retrieves XML document about additional information about coverages
GetCoverage	Retrieves coverage

3.4.1 GetCapabilities

Standard parameters for the **GetCapabilities** request according to the OGC WCS 1.1.1 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WCS**)
REQUEST	Da	Name of the request (**GetCapabilities**)
VERSION	Nu	Version of the service

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WCS
&VERSION=1.1.1
&REQUEST=GetCapabilities
```

XML document example when a single raster layer (named `T20QPD_20171123T144719_TCI`) is published in the QGIS project for the WCS service:

```
<WCS_Capabilities xmlns="http://www.opengis.net/wcs" xmlns:xlink="http://www.w3.
↪org/1999/xlink" xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.
↪org/2001/XMLSchema-instance" version="1.0.0" updateSequence="0"␣
↪xsi:schemaLocation="http://www.opengis.net/wcs http://schemas.opengis.net/wcs/1.
↪0.0/wcsCapabilities.xsd">
  <Service>
    ...
  </Service>
  <Capability>
    ...
```

(continues on next page)

(continuare din pagina precedentă)

```
  </Capability>
  <ContentMetadata>
    <CoverageOfferingBrief>
      <name>T20QPD_20171123T144719_TCI</name>
      <label>T20QPD_20171123T144719_TCI</label>
      <lonLatEnvelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
        <gml:pos>-61.585973 16.331189</gml:pos>
        <gml:pos>-61.52537 16.400376</gml:pos>
      </lonLatEnvelope>
    </CoverageOfferingBrief>
  </ContentMetadata>
</WCS_Capabilities>
```

VERSION

This parameter allows to specify the version of the service to use. Currently, the version values is not internally used and always fallback to `1.1.1`.

3.4.2 DescribeCoverage

This request allows to retrieve additional information about coverages like the format of the underlying datasource, the number of bands, … Standard parameters for the **DescribeCoverage** request according to the OGC WCS 1.1.1 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WCS**)
REQUEST	Da	Name of the request (**DescribeCoverage**)
VERSION	Nu	Version of the service
COVERAGE	Nu	Specify coverage layers (WCS 1.0.0)
IDENTIFIER	Nu	Specify coverage layers (WCS 1.1.1)

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

The XML document for a 3 bands GeoTIFF raster layer looks like:

```
<CoverageDescription xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.
↪opengis.net/wcs" xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.
↪org/2001/XMLSchema-instance" updateSequence="0" version="1.0.0"␣
↪xsi:schemaLocation="http://www.opengis.net/wcs http://schemas.opengis.net/wcs/1.
↪0.0/describeCoverage.xsd">
  <CoverageOffering>
    <name>T20QPD_20171123T144719_TCI</name>
    <label>T20QPD_20171123T144719_TCI</label>
    <lonLatEnvelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
      ...
    </lonLatEnvelope>
    <domainSet>
      ...
    </domainSet>
    <rangeSet>
      <RangeSet>
```

(continues on next page)

(continuare din pagina precedentă)

```
        <name>Bands</name>
        <label>Bands</label>
        <axisDescription>
          <AxisDescription>
            <name>bands</name>
            <label>bands</label>
            <values>
              <singleValue>1</singleValue>
              <singleValue>2</singleValue>
              <singleValue>3</singleValue>
            </values>
          </AxisDescription>
        </axisDescription>
      </RangeSet>
    </rangeSet>
    <supportedCRSs>
    ...
    </supportedCRSs>
    <supportedFormats nativeFormat="raw binary">
      <formats>GeoTIFF</formats>
    </supportedFormats>
  </CoverageOffering>
</CoverageDescription>
```

COVERAGE

This parameter, defined in WCS `1.0.0`, allows to specify the layers to query for additional information. Names have to be separated by a comma.

In addition, QGIS Server introduced an option to select layers by its short name. The short name of a layer may be configured through *Properties* ► *Metadata* in layer menu. If the short name is defined, then it's used by default instead of the layer's name:

```
http://localhost/qgisserver?
SERVICE=WCS
&REQUEST=DescribeCoverage
&COVERAGE=mylayer1name,mylayer2shortname
```

Notă

`COVERAGE` is mandatory if `IDENTIFIER` is not set.

IDENTIFIER

This parameter replaces the *COVERAGE* parameter in WCS `1.1.1`. But QGIS Server does not filter according to the `VERSION` parameter so `IDENTIFIER` and `COVERAGE` have the same effect.

Notă

`IDENTIFIER` is mandatory if `COVERAGE` is not set. If both `IDENTIFIER` and `COVERAGE` parameters are defined, `COVERAGE` is always used in priority.

3.4.3 GetCoverage

This request allows to retrieve the coverage according to specific constraints like the extent or the CRS. Standard parameters for the **DescribeCoverage** request according to the OGC WCS 1.1.1 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WCS**)
REQUEST	Da	Name of the request (**GetCoverage**)
VERSION	Nu	Version of the service
COVERAGE	Nu	Specify coverage layers (WCS 1.0.0)
IDENTIFIER	Nu	Specify coverage layers (WCS 1.1.1)
WIDTH	Da	Width of the response in pixels
HEIGHT	Da	Height of the response in pixels
BBOX	Da	Map extent in CRS units
CRS	Da	Coordinate reference system of the extent
RESPONSE_CRS	Nu	Coordinate reference system of the response

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

BBOX

This parameter allows to specify the map extent in the units of the current CRS. Coordinates have to be separated by a comma. The `BBOX` parameter is formed like `minx,miny,maxx,maxy`.

URL example:

```
http://localhost/qgisserver?
SERVICE=WCS
&REQUEST=GetCoverage
&IDENTIFIER=T20QPD_20171123T144719_TCI
&BBOX=647533,1805950,660987,1813940
&CRS=EPSG:32620
```

CRS

This parameter allows to indicate the Spatial Reference System of the `BBOX` parameter and has to be formed like `EPSG:XXXX`.

RESPONSE_CRS

This parameter allows to indicate the output response Spatial Reference System and has to be formed like `EPSG:XXXX`. The CRS of the corresponding coverage layer is used by default.

WIDTH

This parameter allows to specify the width in pixels of the output image. The resolution of the response image depends on this value.

HEIGHT

This parameter allows to specify the height in pixels of the output image. The resolution of the response image depends on this value.

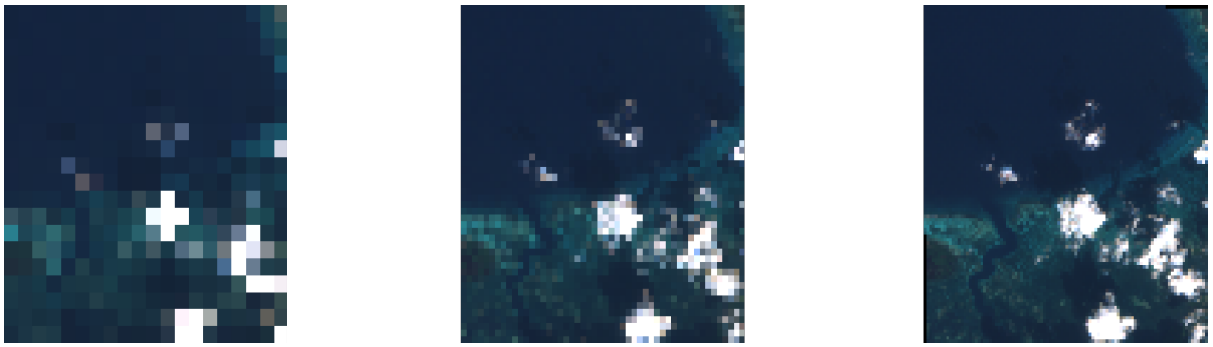


Fig. 3.23: From left to right: `WIDTH=20&HEIGHT=20`, `WIDTH=50&HEIGHT=50`, `WIDTH=100&HEIGHT=100`

3.5 Web Map Tile Service (WMTS)

The **1.0.0** WMTS standard implemented in QGIS Server provides a HTTP interface to request tiled map images generated from a QGIS project. A typical WMTS request defined the QGIS project to use, some WMS parameters like layers to render, as well as tile parameters.

Specifications document of the service:

- WMTS 1.0.0

Standard requests provided by QGIS Server:

Request	Descriere
GetCapabilities	Returns XML metadata with information about the server
GetTile	Returns a tile
GetFeatureInfo	Retrieves data (geometry and values) for a pixel location

3.5.1 GetCapabilities

Standard parameters for the **GetCapabilities** request according to the OGC WMTS 1.0.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMTS**)
REQUEST	Da	Name of the request (**GetCapabilities**)

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/?
SERVICE=WMTS
&REQUEST=GetCapabilities
&MAP=/home/qgis/projects/world.qgs
```

3.5.2 GetTile

Standard parameters for the **GetTile** request according to the OGC WMTS 1.0.0 specifications:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMTS**)
REQUEST	Da	Name of the request (**GetTile**)
LAYER	Da	Layer identifier
FORMAT	Da	Output format of the tile
TILEMATRIXSET	Da	Name of the pyramid
TILEMATRIX	Da	Meshing
TILEROW	Da	Row coordinate in the mesh
TILECOL	Da	Column coordinate in the mesh

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/?
SERVICE=WMTS
&REQUEST=GetTile
&MAP=/home/qgis/projects/world.qgs
&LAYER=mylayer
&FORMAT=image/png
&TILEMATRIXSET=EPSG:4326
&TILEROW=0
&TILECOL=0
```

FORMAT

This parameter may be used to specify the format of tile image. Available values are:

- `jpg`
- `jpeg`
- `image/jpeg`
- `image/png`

If the `FORMAT` parameter is different from one of these values, then the default format PNG is used instead.

TILEMATRIXSET

This parameter defines the CRS to use when computing the underlying pyramid. Format: `EPSG:XXXX`.

TILEMATRIX

This parameter allows to define the matrix to use for the output tile.

TILEROW

This parameter allows to select the row of the tile to get within the matrix.

TILECOL

This parameter allows to select the column of the tile to get within the matrix.

3.5.3 GetFeatureInfo

Standard parameters for the **GetFeatureInfo** request according to the OGC WMTS 1.0.0 specification:

Parametru	Required	Descriere
SERVICE	Da	Name of the service (**WMTS**)
REQUEST	Da	Name of the request (**GetFeatureInfo**)
LAYER	Da	Layer identifier
INFOFORMAT	Nu	Output format
I	Nu	X coordinate of a pixel
J	Nu	Y coordinate of a pixel
TILEMATRIXSET	Da	Name of the pyramid
TILEMATRIX	Meshing	
TILEROW	Da	Row coordinate in the mesh
TILECOL	Da	Column coordinate in the mesh

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parametru	Required	Descriere
MAP	Da	QGIS project file

URL example:

```
http://localhost/?
SERVICE=WMTS
&REQUEST=GetFeatureInfo
&MAP=/home/qgis/projects/world.qgs
&LAYER=mylayer
&INFOFORMAT=image/html
&I=10
&J=5
```

INFOFORMAT

This parameter allows to define the output format of the result. Available values are:

- `text/xml`
- `text/html`
- `text/plain`
- `application/vnd.ogc.gml`

The default value is `text/plain`.

I

This parameter allows to define the X coordinate of the pixel for which we want to retrieve underlying information.

J

This parameter allows to define the Y coordinate of the pixel for which we want to retrieve underlying information.

3.6 OGC API Features

OGC API Features (OAPIF) is the first implementation of the new generation of OGC protocols. It is described by the OGC API - Features - Part 1: Core document.

The API can be reached on typical installations via `http://localhost/qgisserver/wfs3`.

Here is a quick informal summary of the most important differences between the well known WFS protocol and OAPIF:

- OAPIF is based on a REST API
- OAPIF must follow the OPENAPI specifications
- OAPIF supports multiple output formats but it does not dictate any (only GeoJSON and HTML are currently available in QGIS OAPIF) and it uses content negotiation to determine which format is to be served to the client
- JSON and HTML are first class citizens in OAPIF
- OAPIF is self-documenting (through the `/api` endpoint)
- OAPIF is fully navigable (through links) and browsable

Important

While the OGC API Features implementation in QGIS can make use of the `MAP` parameter to specify the project file, no extra query parameters are allowed by the OPENAPI specification. For this reason it is strongly recommended that `MAP` is not exposed in the URL and the project file is specified in the environment by other means (i.e. setting `QGIS_PROJECT_FILE` in the environment through a web server rewrite rule).

Notă

The **API** endpoint provides comprehensive documentation of all supported parameters and output formats of your service. The following paragraphs will only describe the most important ones.

3.6.1 Resource representation

The implementation of OGC API Features in QGIS Server currently supports the following resource representation (output) formats:

- HTML
- JSON

The format that is actually served will depend on content negotiation, but a specific format can be explicitly requested by appending a format specifier to the endpoints.

Supported format specifier extensions are:

- `.json`
- `.html`

Additional format specifier aliases may be defined by specific endpoints:

- `.openapi`: alias for `.json` supported by the **API** endpoint
- `.geojson`: alias for `.json` supported by the **Features** and **Feature** endpoints

3.6.2 Endpoints

The API provides a list of endpoints that the clients can retrieve. The system is designed in such a way that every response provides a set of links to navigate through all the provided resources.

Endpoints points provided by the QGIS implementation are:

Nume	Path	Descriere
Landing Page	`/`	General information about the service and provides links to all available endpoints
Conformance	`/conformance`	Information about the conformance of the service to the standards
API	`/api`	Full description of the endpoints provided by the service and the returned documents structure
Collections	`/collections`	List of all collections (i.e. «vector layers») provided by the service
Collection	`/collections/{collectionId}`	Information about a collection (name, metadata, extent etc.)
Funcțiuni	`/collections/{collectionId}/items`	List of the features provided by the collection
Feature	`/collections/{collectionId}/items/{featureId}`	Information about a single feature

Similar to WFS-T (transactional Web Feature Service), it is possible to add, update and delete features (CRUD). The respective requests are described on „`/api`”.

Landing Page

The main endpoint is the **Landing Page**. From that page it is possible to navigate to all the available service endpoints. The **Landing Page** must provide links to

- the API definition (path `/api` link relations `service-desc` and `service-doc`),
- the Conformance declaration (path `/conformance`, link relation `conformance`), and
- the Collections (path `/collections`, link relation `data`).

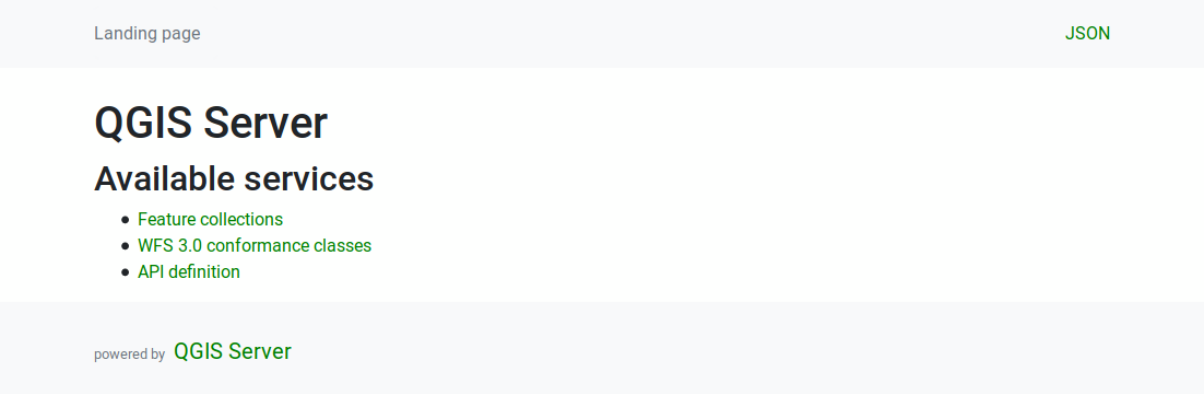


Fig. 3.24: Server OAPIF landing page

API Definition

The **API Definition** is an OPENAPI-compliant description of the API provided by the service. In its HTML representation it is a browsable page where all the endpoints and their response formats are accurately listed and documented. The path of this endpoint is `/api`.

The API definition provides a comprehensive and authoritative documentation of the service, including all supported parameters and returned formats.

> **Notă**
>
> This endpoint is analogue to WFS's `GetCapabilities`

Collections list

The collections endpoint provides a list of all the collections available in the service. Since the service „serves" a single QGIS project the collections are the vector layers from the current project (if they were published as WFS in the project properties). The path of this endpoint is `/collections/`.

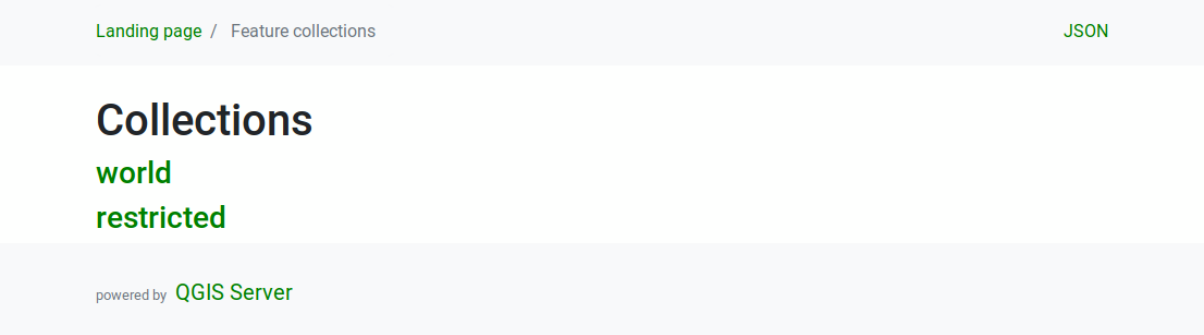


Fig. 3.25: Server OAPIF collections list page

Collection detail

While the collections endpoint does not provide detailed information about each available collection, that information is available in the `/collections/{collectionId}` endpoints. Typical information includes the extent, a description, CRSs and other metadata.

The HTML representation also provides a browsable map with the available features.

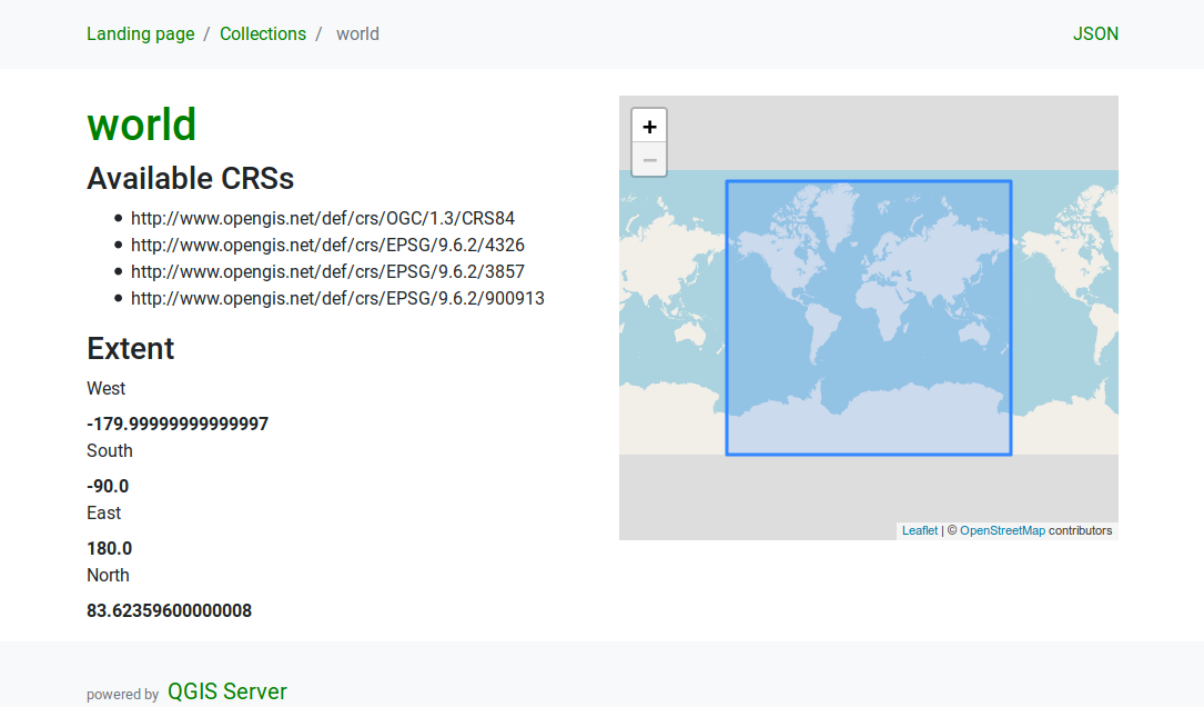


Fig. 3.26: Server OAPIF collection detail page

Features list

This endpoint provides a list of all features in a collection knowing the collection ID. The path of this endpoint is `/collections/{collectionId}/items`.

The HTML representation also provides a browsable map with the available features.

> **Notă**
>
> This endpoint is analogue to `GetFeature` in WFS 1 and WFS 2.

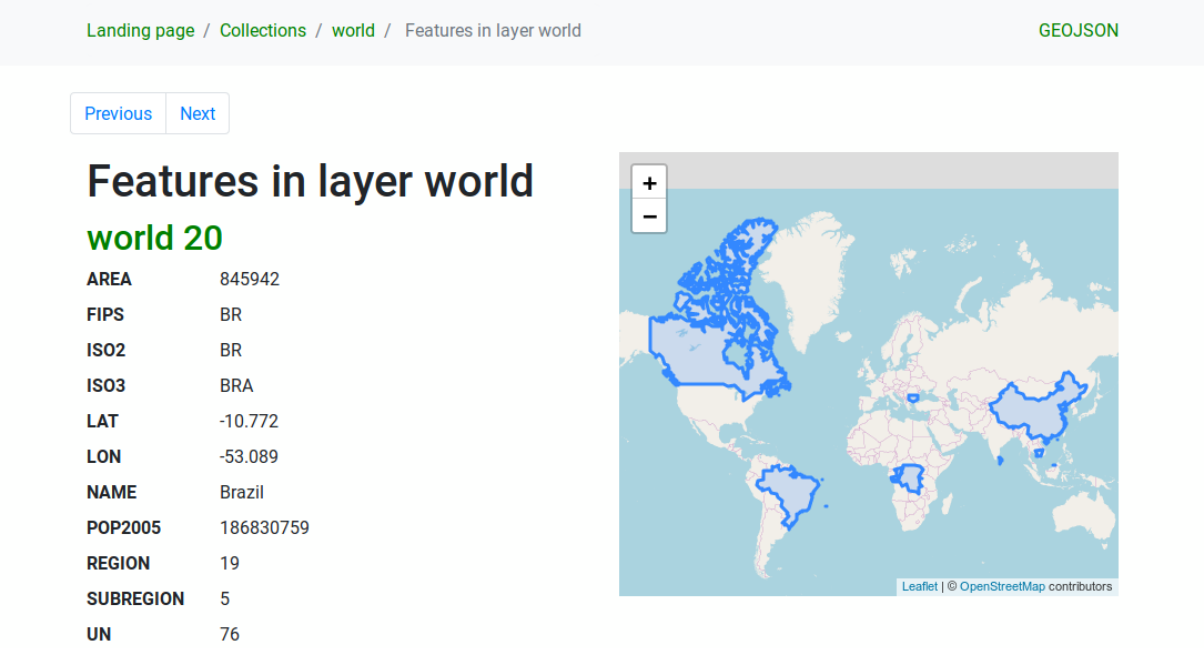


Fig. 3.27: Server OAPIF features list page

Feature detail

This endpoint provides all the available information about a single feature, including the feature attributes and its geometry. The path of this endpoint is `/collections/{collectionId}/items/{itemId}`.

The HTML representation also provides a browsable map with the feature geometry.

Fig. 3.28: Server OAPIF feature detail page

3.6.3 Pagination

Pagination of a long list of features is implemented in the OGC API through `next` and `prev` links, QGIS server constructs these links by appending `limit` and `offset` as query string parameters.

URL example:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?offset=10&limit=10
```

Notă

The maximum acceptable value for `limit` can be configured with the `QGIS_SERVER_API_WFS3_MAX_LIMIT` server configuration setting (see: *Variabile de mediu*).

3.6.4 Filtrarea entităților

The features available in a collection can be filtered/searched by specifying one or more filters.

Date and time filter

Collections with date and/or datetime attributes can be filtered by specifying a `datetime` argument in the query string. By default the first date/datetime field is used for filtering. This behavior can be configured by setting a „Date” or „Time” dimension in the *QGIS Server* ► *Dimension* section of the layer properties dialog.

The date and time filtering syntax is fully described in the *API Definition* and also supports ranges (begin and end values are included) in addition to single values.

URL examples:

Returns only the features with date dimension matching `2019-01-01`

```
http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-01
```

Returns only the features with datetime dimension matching `2019-01-01T01:01:01`

```
http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-
↪01T01:01:01
```

Returns only the features with datetime dimension in the range `2019-01-01T01:01:01 - 2019-01-01T12:00:00`

```
http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-
↪01T01:01:01/2019-01-01T12:00:00
```

Bounding box filter

A bounding box spatial filter can be specified with the `bbox` parameter:

The order of the comma separated elements is:

- Lower left corner, WGS 84 longitude
- Lower left corner, WGS 84 latitude
- Upper right corner, WGS 84 longitude
- Upper right corner, WGS 84 latitude

Notă

The OGC specifications also allow a 6 item bbox specifier where the third and sixth items are the Z components, this is not yet supported by QGIS server.

URL example:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?bbox=-180,-90,180,90
```

If the *CRS* of the bounding box is not WGS 84, a different CRS can be specified by using the optional parameter `bbox-crs`. The CRS format identifier must be in the OGC URI format:

URL example:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?bbox=913191,5606014,
↪913234,5606029&bbox-crs=http://www.opengis.net/def/crs/EPSG/9.6.2/3857
```

Attribute filters

Attribute filters can be combined with the bounding box filter and they are in the general form: `<attribute name>=<attribute value>`. Multiple filters can be combined using the `AND` operator.

URL example:

filters all features where attribute `name` equals „my value”

```
http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=my%20value
```

Partial matches are also supported by using a `*` („star”) operator:

URL example:

filters all features where attribute `name` ends with „value”

```
http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=*value
```

3.6.5 Feature sorting

It is possible to order the result set by field value using the `sortby` query parameter.

The results are sorted in ascending order by default. To sort the results in descending order, a boolean flag (`sortdesc`) can be set:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?sortby=name&sortdesc=1
```

3.6.6 Attribute selection

The feature attributes returned by a *Features list* call can be limited by adding a comma separated list of attribute names in the optional `properties` query string argument.

URL example:

returns only the `name` attribute

```
http://localhost/qgisserver/wfs3/collection_one/items.json?properties=name
```

3.6.7 Customize the HTML pages

The HTML representation uses a set of HTML templates to generate the response. The template is parsed by a template engine called inja. The templates can be customized by overriding them (see: *Template overrides*). The template has access to the same data that are available to the `JSON` representation and a few additional functions are available to the template:

Custom template functions

- `path_append( path )`: appends a directory path to the current url
- `path_chomp( n )`: removes the specified number „n” of directory components from the current url path
- `json_dump( )`: prints the JSON data passed to the template
- `static( path )`: returns the full URL to the specified static path. For example: „static(„/style/black.css”)” with a root path „http://localhost/qgisserver/wfs3” will return „http://localhost/qgisserver/wfs3/static/style/black.css”.
- `links_filter( links, key, value )`: Returns filtered links from a link list
- `content_type_name( content_type )`: Returns a short name from a content type, for example „text/html” will return „HTML”
- `nl2br( text )`: Returns the input text with all newlines replaced by „
” tags
- `starts_with( string, prefix )`: returns true if a string begins with the provided string prefix, false otherwise

Template overrides

Templates and static assets are stored in subdirectories of the QGIS server default API resource directory (`/usr/share/qgis/resources/server/api/` on a Linux system), the base directory can be customized by changing the environment variable `QGIS_SERVER_API_RESOURCES_DIRECTORY`.

A typical Linux installation will have the following directory tree:

```
/usr/share/qgis/resources/server/api/
└── ogc
    ├── schema.json
    ├── static
    │   ├── jsonFormatter.min.css
    │   ├── jsonFormatter.min.js
    │   └── style.css
    └── templates
        └── wfs3
            ├── describeCollection.html
            ├── describeCollections.html
            ├── footer.html
            ├── getApiDescription.html
            ├── getFeature.html
            ├── getFeatures.html
            ├── getLandingPage.html
            ├── getRequirementClasses.html
            ├── header.html
            ├── leaflet_map.html
            └── links.html
```

To override the templates you can copy the whole tree to another location and point `QGIS_SERVER_API_RESOURCES_DIRECTORY` to the new location.

CAPITOLUL 4

Catalog

The QGIS Server Catalog is a simple catalog that shows the list of QGIS projects served by the QGIS Server. It provides a user-friendly fully browsable website with basic mapping capabilities to quickly browse the datasets exposed through those QGIS projects. The catalog can be configured using the environment variables starting with `QGIS_SERVER_LANDING_PAGE` (see *Variabile de mediu*).

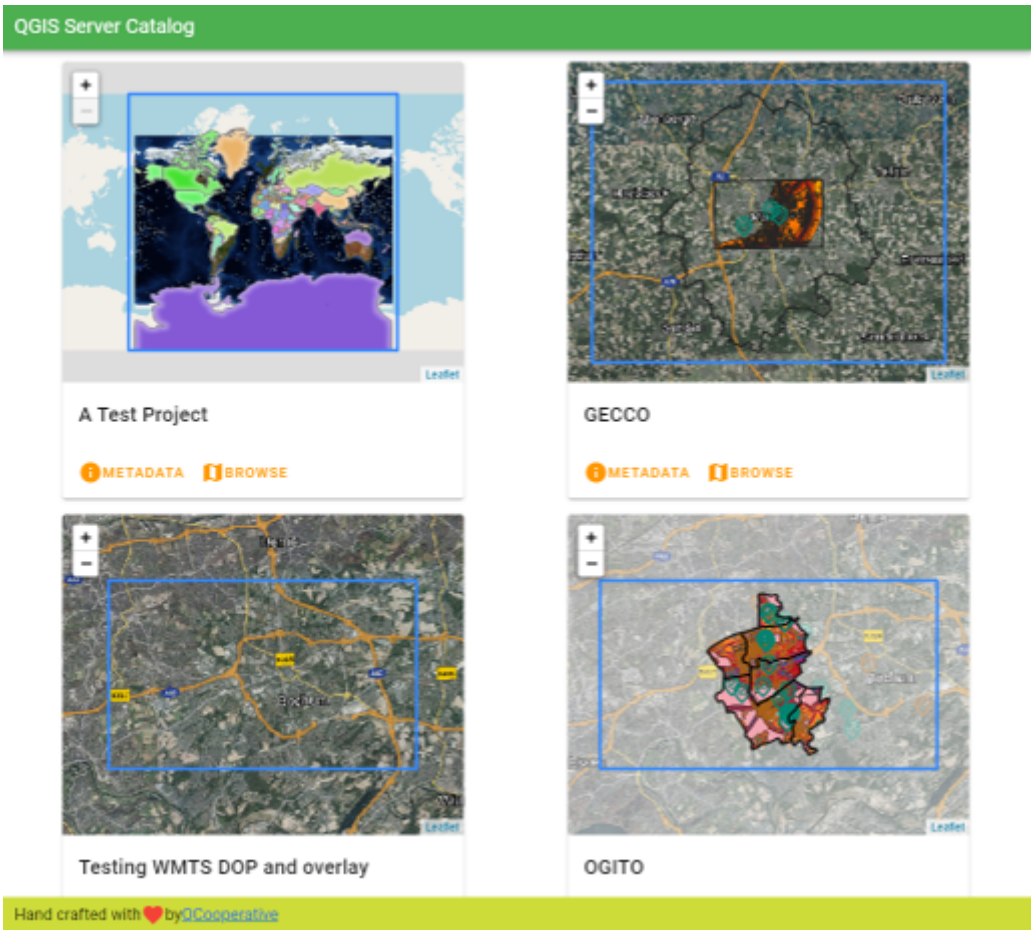


Fig. 4.1: Server Catalog project list page

You can consult the metadata associated to a project and the services that it provides. Links to those services are also given.

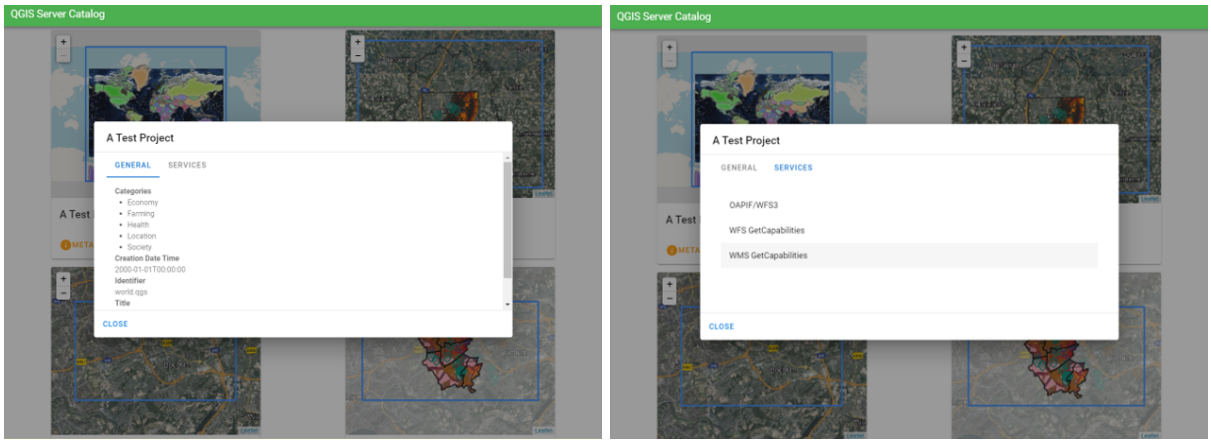


Fig. 4.2: Server Catalog, metadata associated to a project and services (links to) that it provides.

By browsing a project, it is listed the dataset that it serves.

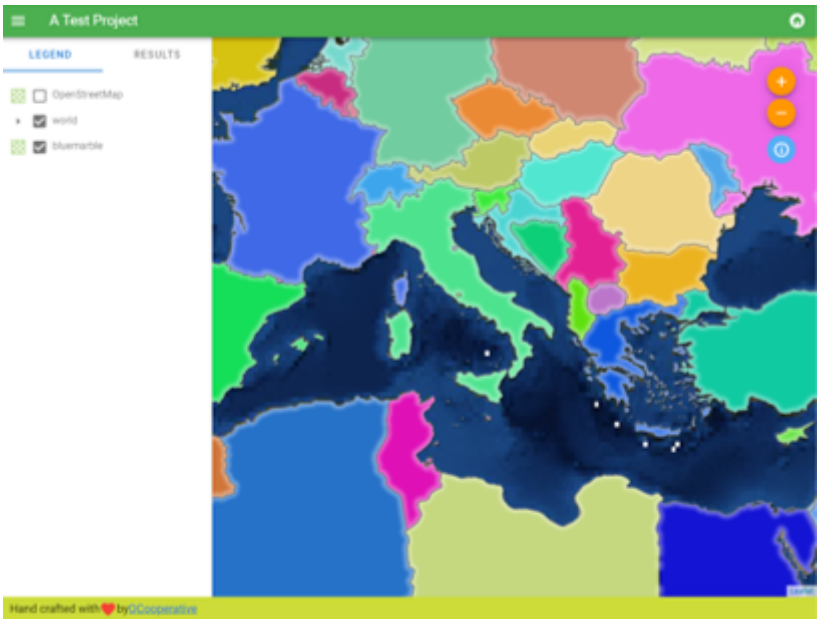


Fig. 4.3: Browsing a dataset served by a project in the Server Catalog

Use Right click on a layer to display the attribute table associated to it.

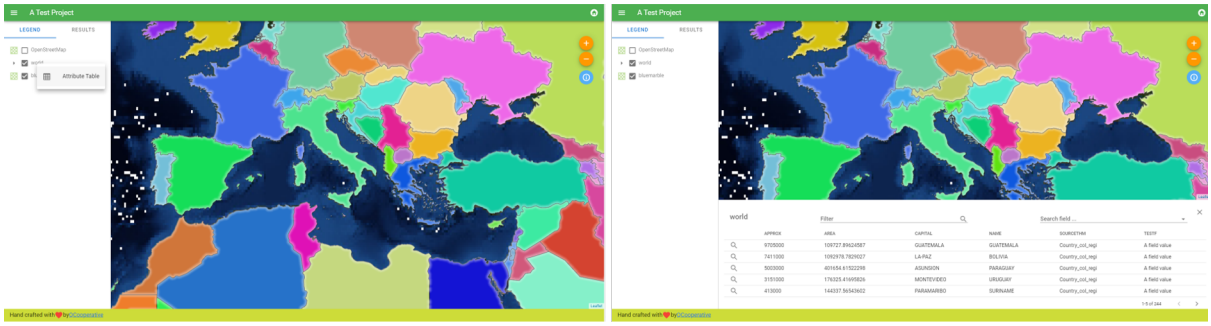


Fig. 4.4: Attribute table associated to a layer

It is possible to consult information of the elements in the map as shown in the image below:

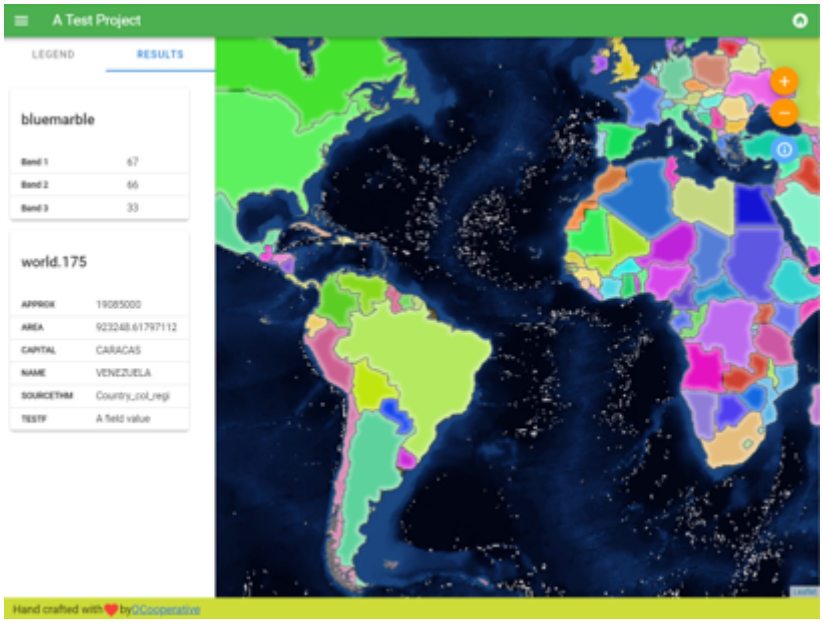


Fig. 4.5: Consulting information of a map element

CAPITOLUL 5

Plugin-uri

5.1 Lista plugin-urilor

Plugins can also be installed on QGIS Server.

Some plugins designed for server can be found on the official QGIS repository.

Install only plugins you need for your own purpose. On QGIS server, plugins are like hooks into QGIS server, they can alter inputs or outputs of QGIS server. They can produce unexpected result if you don't know how the plugin works. Please refer to their respective documentation or the application that needs QGIS server plugins to know which plugin can be useful for you.

5.2 Location of plugins

By default, on Debian based systems, QGIS Server will look for plugins located in `/usr/lib/qgis/plugins`. The default value is displayed when QGIS Server is starting, in the logs. It's possible to set a custom path by defining the environment variable `QGIS_PLUGINPATH` in the web server configuration.

5.3 Instalarea

5.3.1 Manually with a ZIP

As an example, to install the **HelloWorld** plugin for testing the server, using a specific folder, you first have to create a directory to hold server plugins. This will be specified in the virtual host configuration and passed on to the server through an environment variable:

```
mkdir -p /var/www/qgis-server/plugins
cd /var/www/qgis-server/plugins
wget https://github.com/elpaso/qgis-helloserver/archive/master.zip
unzip master.zip
mv qgis-helloserver-master HelloServer
```

> **Atenționare**
>
> According to its description, HelloServer plugin is designed for development and demonstration purposes. Do not keep this plugin for production if you don't need it.

5.3.2 Cu un instrument în linie de comandă

If you need to install and regularly upgrade plugins which are stored in the QGIS plugin repository, you may use the QGIS-Plugin-Manager. It's a tool to help you manage plugins from the command line.

The installation is using pip. Installing in a virtual environment is a good practice but not required:

```
pip3 install qgis-plugin-manager
```

Pentru a actualiza instrumentul:

```
pip3 install --upgrade qgis-plugin-manager
```

Then, you can use the `qgis-plugin-manager` executable from the command line:

```
cd /var/www/qgis-server/plugins
qgis-plugin-manager list

QGIS server version 3.19.0
List all plugins in /var/www/qgis-server/plugins

--------------------------------------------------------------------------------
↪-----------------------------------------------------------------------
|  Folder                    |  Name              |  Version  |  Experimental  | ␣
↪QGIS min  |  QGIS max  |  Author                    |  Action ⍰          |
--------------------------------------------------------------------------------
↪-----------------------------------------------------------------------
|wfsOutputExtension          |wfsOutputExtension  |1.6.2      |                |3.
↪0         |            |3Liz                        |                    |
|qgis_server_render_geojson  |GeoJson Renderer    |v0.4       |                |3.
↪4         |            |Matthias Kuhn (OPENGIS.ch)  |                    |
|DataPlotly                  |Data Plotly         |3.7.1      |                |3.
↪4         |3.98        |Matteo Ghetta (Faunalia)    |Upgrade to 3.8.1    |
--------------------------------------------------------------------------------
↪-----------------------------------------------------------------------
```

We suggest you to read the full documentation in the readme file to know how to install or upgrade plugins with this tool.

5.4 Configurarea serverului HTTP

5.4.1 Apache

To be able to use server plugins, FastCGI needs to know where to look. So, we have to modify the Apache configuration file to indicate the `QGIS_PLUGINPATH` environment variable to FastCGI:

```
FcgidInitialEnv QGIS_PLUGINPATH "/var/www/qgis-server/plugins"
```

Moreover, a basic HTTP authorization is necessary to play with the HelloWorld plugin previously introduced. So we have to update the Apache configuration file a last time:

```
# Needed for QGIS HelloServer plugin HTTP BASIC auth
<IfModule mod_fcgid.c>
    RewriteEngine on
    RewriteCond %{HTTP:Authorization} .
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

Apoi, restartați Apache:

```
systemctl restart apache2
```

5.5 Cum se utilizează un plugin

Testați serverul cu plugin-ul HelloWorld:

```
wget -q -O - "http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=HELLO"
 HelloServer!
```

You can have a look at the default GetCapabilities of the QGIS server at:

```
http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&
↪REQUEST=GetCapabilities
```

CAPITOLUL 6

Configurare avansată

6.1 Jurnalizarea

To log requests sent to the server, you have to set the following environment variable:

- *QGIS_SERVER_LOG_STDERR*

With the following variables the logging can be further customized:

- *QGIS_SERVER_LOG_LEVEL*
- *QGIS_SERVER_LOG_PROFILE*

6.2 Variabile de mediu

You can configure some aspects of QGIS Server by setting **environment variables**.

According to the HTTP server and how you run QGIS Server, there are several ways to define these variables. This is fully described in *Server HTTP Apache*.

Name & Description	Implicit	Servicii
QGIS_OPTIONS_PATH Specifies the path to the directory with settings. It works the same way as QGIS application `--optionspath` option. It is looking for settings file in `<QGIS_OPTIONS_PATH>/QGIS/QGIS3.ini`.	«»	All
QGIS_PLUGINPATH Useful if you are using Python plugins for the server, this sets the folder that is searched for Python plugins.	«»	All

continues on next page

Tabelul 6.1 – continuare din pagina precedentă

Name & Description	Implicit	Servicii
QGIS_PROJECT_FILE The `.qgs` or `.qgz` project file, normally passed as a parameter in the query string (with *MAP*), you can also set it as an environment variable (for example by using `mod_rewrite` Apache module). **Notă** You may also indicate a project stored in PostgreSQL, e.g. `postgresql://localhost:5432?sslmode=disable&dbname=mydb&schema=myschema&pr` or inside a geopackage file, e.g. `geopackage:/path/to/geopackage/file.gpkg?projectName=myProjectName`.	«»	All
QGIS_SERVER_ALLOWED_EXTRA_SQL_TOKENS Comma separated list of strings that represent the allowed extra SQL tokens accepted as components of a feature filter.	«»	WMS
QGIS_SERVER_API_RESOURCES_DIRECTORY Base directory for all OGC API (such as OAPIF/WFS3) static resources (HTML templates, CSS, JS, …)	depends on packaging	OAPIF/WFS3
QGIS_SERVER_APPLICATION_NAME Name of the application to be used, for instance when connecting to a database to identify the QGIS server instance connected	QGIS3 server	All
QGIS_SERVER_API_WFS3_MAX_LIMIT Maximum value for `limit` in a OAPIF/WFS3 features request.	10000	OAPIF/WFS3
QGIS_SERVER_CACHE_DIRECTORY Specifies the network cache directory on the filesystem.	`cache` in profile directory	All
QGIS_SERVER_CACHE_SIZE Sets the network cache size in MB.	50 MB	All
QGIS_SERVER_CAPABILITIES_CACHE_SIZE The maximum number of project capabilities to cache.	40	All
QGIS_SERVER_DISABLE_GETPRINT This is an option at the project level to improve project read time by disabling loading of layouts. Activating this option disables the QGIS WMS GetPrint request. Set this QGIS project flag to not load layouts.	false	WMS

continues on next page

Tabelul 6.1 – continuare din pagina precedentă

Name & Description	Implicit	Servicii
QGIS_SERVER_FORCE_READONLY_LAYERS Force QGIS Server to open all layers in a read only mode	false	All
QGIS_SERVER_IGNORE_BAD_LAYERS „Bad" layers are layers that cannot be loaded. The default behavior of QGIS Server is to consider the project as not available if it contains a bad layer. The default behavior can be overridden by setting this variable to `1` or `true`. In this case, „bad" layers will just be ignored, and the project will be considered valid and available.	false	All
QGIS_SERVER_LANDING_PAGE_PREFIX Prefix of the path component of the landing page base URL	„"	All
QGIS_SERVER_LANDING_PAGE_PROJECTS_DIRECTORIES Directories used by the landing page service to find .qgs and .qgz projects	„"	All
QGIS_SERVER_LANDING_PAGE_PROJECTS_PG_CONNECTIONS PostgreSQL connection strings used by the landing page service to find projects	„"	All
QGIS_SERVER_LOG_FILE Specify path and filename. Make sure that server has proper permissions for writing to file. File should be created automatically, just send some requests to server. If it's not there, check permissions. **Atenţionare** QGIS_SERVER_LOG_FILE is deprecated since QGIS 3.4, use QGIS_SERVER_LOG_STDERR instead. File logging support will be removed in QGIS 4.0.	«»	All
QGIS_SERVER_LOG_LEVEL Specify desired log level. Available values are: • `0` or `INFO` (log all requests) • `1` or `WARNING` • `2` or `CRITICAL` (log just critical errors, suitable for production purposes)	0	All
QGIS_SERVER_LOG_PROFILE Add detailed profile information to the logs, only effective when QGIS_SERVER_LOG_LEVEL=0	false	All

continues on next page

Tabelul 6.1 – continuare din pagina precedentă

Name & Description	Implicit	Servicii
QGIS_SERVER_LOG_STDERR Activate logging to stderr. This variable has no effect when `QGIS_SERVER_LOG_FILE` is set. • `0` or `false` (case insensitive) • `1` or `true` (case insensitive)	false	All
QGIS_SERVER_MAX_THREADS Number of threads to use when parallel rendering is activated. If value is `-1` it uses the number of processor cores.	-1	All
QGIS_SERVER_OVERRIDE_SYSTEM_LOCALE Sets LOCALE to be used by QGIS server. The default value is empty (no override). Example: `de_CH.utf8`	«»	All
QGIS_SERVER_PARALLEL_RENDERING Activates parallel rendering for WMS GetMap requests. It's disabled (`false`) by default. Available values are: • `0` or `false` (case insensitive) • `1` or `true` (case insensitive)	false	WMS
QGIS_SERVER_PROJECT_CACHE_CHECK_INTERVAL Controls the periodic strategy interval for cache invalidation, in milliseconds. Defaults to 0 which selects the legacy File system watcher.		All
QGIS_SERVER_PROJECT_CACHE_STRATEGY Defines method for invalidating the project cache. Available strategies are: • `filesystem`: uses the file system watcher strategy • `periodic`: uses the last modified value of a project for checking changes on project configuration. Convenient on atypical file systems, such as NFS, or when the project file is stored in a database system like PostgreSQL. • `off`: disables internal cache invalidation completely	filesystem	All

continues on next page

Tabelul 6.1 – continuare din pagina precedentă

Name & Description	Implicit	Servicii
QGIS_SERVER_SERVICE_URL This is an option to set the service URL if it is not present in the project. The service URL is defined from (in order of precedence): • Value defined in the project per service • The `QGIS_SERVER_<service>_SERVICE_URL` environment variable • The `QGIS_SERVER_SERVICE_URL` environment variable • The `X-Qgis-<service>-Service-Url` header • The `X-Qgis-Service-Url` header • Build from the `Forwarded` header • Build from the `X-Forwarded-Host` and `X-Forwarded-Proto` headers • Build from the `Host` header and the server protocol • Build from the server name and the server protocol. In the last four cases, the resulting Service URL is based on the `MAP` parameter provided in the query string and on the incoming path request.	«»	All
QGIS_SERVER_SHOW_GROUP_SEPARATOR Defines whether a group separator (e.g. thousand separator) should be used for numeric values (e.g. in GetFeatureInfo responses). The default value is `0`. • `0` or `false` (case insensitive) • `1` or `true` (case insensitive)	false	WMS
QGIS_SERVER_TRUST_LAYER_METADATA This is an option at the project level to improve project read time by using the vector layer extents defined in the project metadata and disabling the check for PostgreSQL/PostGIS layer primary key uniqueness. Trusting layer metadata can be forced by setting this variable to `1` or `true`. The vector layer's extent will then be the one defined in the project, and the PostgreSQL/PostGIS layer's primary key defined in the data source is considered as unique without a check. Do not use it if layers» extent is not fixed during the project's use.	false	All
QGIS_SERVER_WCS_SERVICE_URL This is an option to set the service URL if it is not present in the project. See *QGIS_SERVER_SERVICE_URL* for more information.	«»	WCS
QGIS_SERVER_WFS_SERVICE_URL This is an option to set the service URL if it is not present in the project. See *QGIS_SERVER_SERVICE_URL* for more information.	«»	WFS
QGIS_SERVER_WMS_MAX_HEIGHT / QGIS_SERVER_WMS_MAX_WIDTH Maximum height/width for a WMS request. The most conservative between this and the project one is used. If the value is `-1`, it means that there is no maximum set.	-1	WMS

continues on next page

Tabelul 6.1 – continuare din pagina precedentă

Name & Description	Implicit	Servicii
QGIS_SERVER_WMS_SERVICE_URL This is an option to set the service URL if it is not present in the project. See *QGIS_SERVER_SERVICE_URL* for more information.	«»	WMS
QGIS_SERVER_WMTS_SERVICE_URL This is an option to set the service URL if it is not present in the project. See *QGIS_SERVER_SERVICE_URL* for more information.	«»	WMTS
QUERY_STRING The query string, normally passed by the web server. This variable can be useful while testing QGIS server binary from the command line. For example for testing a GetCapabilities request on the command line to a project that also requires a PostgreSQL connection defined in a pg_service.conf file: `PGSERVICEFILE=/etc/pg_service.conf \ QUERY_STRING="MAP=/home/projects/world.qgs& ↪SERVICE=WMS&REQUEST=GetCapabilities" \ /usr/lib/cgi-bin/qgis_mapserv.fcgi` The result should be either the content of the GetCapabilities response or, if something is wrong, an error message.	«»	All

6.3 Settings summary

When QGIS Server is starting, you have a summary of all configurable parameters thanks to environment variables. Moreover, the value currently used and the origin is also displayed.

For example with spawn-fcgi:

```
export QGIS_OPTIONS_PATH=/home/user/.local/share/QGIS/QGIS3/profiles/default/
export QGIS_SERVER_LOG_STDERR=1
export QGIS_SERVER_LOG_LEVEL=2
spawn-fcgi -f /usr/lib/cgi-bin/qgis_mapserv.fcgi -s /tmp/qgisserver.sock -U www-
↪data -G www-data -n

 QGIS Server Settings:

   - QGIS_OPTIONS_PATH / '' (Override the default path for user configuration): '/
↪home/user/.local/share/QGIS/QGIS3/profiles/default/' (read from ENVIRONMENT_
↪VARIABLE)

   - QGIS_SERVER_PARALLEL_RENDERING / '/qgis/parallel_rendering' (Activate/
↪Deactivate parallel rendering for WMS getMap request): 'true' (read from INI_
↪FILE)

   - QGIS_SERVER_MAX_THREADS / '/qgis/max_threads' (Number of threads to use when␣
↪parallel rendering is activated): '4' (read from INI_FILE)

   - QGIS_SERVER_LOG_LEVEL / '' (Log level): '2' (read from ENVIRONMENT_VARIABLE)

   - QGIS_SERVER_LOG_STDERR / '' (Activate/Deactivate logging to stderr): '1'␣
↪(read from ENVIRONMENT_VARIABLE)
```

(continues on next page)

(continuare din pagina precedentă)

```
  - QGIS_PROJECT_FILE / '' (QGIS project file): '' (read from DEFAULT_VALUE)

  - MAX_CACHE_LAYERS / '' (Specify the maximum number of cached layers): '100'␣
↪(read from DEFAULT_VALUE)

  - QGIS_SERVER_CACHE_DIRECTORY / '/cache/directory' (Specify the cache␣
↪directory): '/root/.local/share/QGIS/QGIS3/profiles/default/cache' (read from␣
↪DEFAULT_VALUE)

  - QGIS_SERVER_CACHE_SIZE / '/cache/size' (Specify the cache size): '52428800'␣
↪(read from INI_FILE)

 Ini file used to initialize settings: /home/user/.local/share/QGIS/QGIS3/profiles/
↪default/QGIS/QGIS3.ini
```

In this particular case, we know that **QGIS_SERVER_MAX_THREADS** and **QGIS_SERVER_PARALLEL_RENDERING** values are read from the ini file found in **QGIS_OPTIONS_PATH** directory (which is defined through an environment variable). The corresponding entries in the ini file are **/qgis/max_threads** and **/qgis/parallel_rendering** and their values are **true** and **4** threads.

6.4 Connection to service file

In order to make apache aware of the PostgreSQL service file (see the pg-service-file section) you need to make your `*.conf` file look like:

```
SetEnv PGSERVICEFILE /home/web/.pg_service.conf

<Directory "/home/web/apps2/bin/">
  AllowOverride None
.....
```

6.5 Add fonts to your linux server

Keep in mind that you may use QGIS projects that point to fonts that may not exist by default on other machines. This means that if you share the project, it may look different on other machines (if the fonts don't exist on the target machine).

In order to ensure this does not happen you just need to install the missing fonts on the target machine. Doing this on desktop systems is usually trivial (double clicking the fonts).

For linux, if you don't have a desktop environment installed (or you prefer the command line) you need to:

- On Debian based systems:

```
sudo su
mkdir -p /usr/local/share/fonts/truetype/myfonts && cd /usr/local/share/fonts/
↪truetype/myfonts

# copy the fonts from their location
cp /fonts_location/* .

chown root *
cd .. && fc-cache -f -v
```

- On Fedora based systems:

```
sudo su
mkdir /usr/share/fonts/myfonts && cd /usr/share/fonts/myfonts

# copy the fonts from their location
cp /fonts_location/* .

chown root *
cd .. && fc-cache -f -v
```

CAPITOLUL 7

Development Server

A production installation and deployment of QGIS Server usually involves setting up a web server component (e.g. Apache or Nginx) that can forward the HTTP requests coming from the clients to the QGIS Server FastCGI binary application.

If you want to quickly test QGIS Server on your local machine without configuring and installing a full web server stack you can use the QGIS Development Standalone server.

This is an independent application that provides a very simple web server ready to serve your project files.

Atenționare

The Standalone Development Server has not been developed with the purpose of being used in production, it was not checked for security vulnerabilities or for other stress conditions that normally will occur on a publicly exposed server.

To launch the server:

```
$ qgis_mapserver
```

The default port the Development Server listens to is `8000`. Example output:

```
QGIS Development Server listening on http://localhost:8000
CTRL+C to exit
127.0.0.1 [lun gen 20 15:16:41 2020] 5140 103ms "GET /wfs3/?MAP=/tests/testdata/
↪qgis_server/test_project.qgs HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 3298 2ms "GET /wfs3/static/jsonFormatter.min.
↪js HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 1678 3ms "GET /wfs3/static/jsonFormatter.min.
↪css HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 1310 5ms "GET /wfs3/static/style.css HTTP/1.1
↪" 200
127.0.0.1 [lun gen 20 15:16:43 2020] 4285 13ms "GET /wfs3/collections?MAP=/tests/
↪testdata/qgis_server/test_project.qgs HTTP/1.1" 200
```

The server has a few options that can be passed as command line arguments. You can see them all by invoking the server with `-h`.

```
Usage: qgis_mapserver [options] [address:port]
QGIS Development Server

Options:
-h, --help        Displays this help.
-v, --version     Displays version information.
-l <logLevel>     Sets log level (default: 0)
                    0: INFO
                    1: WARNING
                    2: CRITICAL
-p <projectPath>  Path to a QGIS project file (*.qgs or *.qgz),
                    if specified it will override the query string MAP argument
                    and the QGIS_PROJECT_FILE environment variable

Arguments:
addressAndPort    Listen to address and port (default: "localhost:8000")
                    address and port can also be specified with the environment
                    variables QGIS_SERVER_ADDRESS and QGIS_SERVER_PORT
```

CAPITOLUL 8

Containerized deployment

There are many ways to use containerized application, from the most simple (simple Docker images) to sophisticated (Kubernetes and so on).

Notă

This kind of deployment needs the docker application to be installed and running. Check this tutorial.

Sugestie

Docker run pre packaged application (aka images) which can be retrieved as sources (Dockerfile and resources) to build or already built from registries (private or public).

Notă

QGIS Debian-Ubuntu package downloads need a valid gpg authentication key. Please refer to the installation pages to update the following Dockerfile.

8.1 Simple docker images

As the docker image does not exist in a public registry. you will need to build it. To do so create a directory `qgis-server` and within its directory:

- create a file `Dockerfile` with this content:

```
FROM debian:bookworm-slim

ENV LANG=en_EN.UTF-8


RUN apt-get update \
    && apt-get install --no-install-recommends --no-install-suggests --allow-
```

(continues on next page)

(continuare din pagina precedentă)

```
↪unauthenticated -y \
        gnupg \
        ca-certificates \
        wget \
        locales \
    && localedef -i en_US -f UTF-8 en_US.UTF-8 \
    # Add the current key for package downloading
    # Please refer to QGIS install documentation (https://www.qgis.org/fr/site/
↪forusers/alldownloads.html#debian-ubuntu)
    && mkdir -m755 -p /etc/apt/keyrings \
    && wget -O /etc/apt/keyrings/qgis-archive-keyring.gpg https://qgis.org/
↪download//downloads/qgis-archive-keyring.gpg \
    # Add repository for latest version of qgis-server
    # Please refer to QGIS repositories documentation if you want other version␣
↪(https://qgis.org/resources/installation-guide/#repositories)
    && echo "deb [signed-by=/etc/apt/keyrings/qgis-archive-keyring.gpg] https://
↪qgis.org/debian bookworm main" | tee /etc/apt/sources.list.d/qgis.list \
    && apt-get update \
    && apt-get install --no-install-recommends --no-install-suggests --allow-
↪unauthenticated -y \
        qgis-server \
        spawn-fcgi \
        xauth \
        xvfb \
    && apt-get remove --purge -y \
        gnupg \
        wget \
    && rm -rf /var/lib/apt/lists/*

RUN useradd -m qgis

ENV TINI_VERSION v0.19.0
ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/tini /tini
RUN chmod +x /tini

ENV QGIS_PREFIX_PATH /usr
ENV QGIS_SERVER_LOG_STDERR 1
ENV QGIS_SERVER_LOG_LEVEL 2

COPY cmd.sh /home/qgis/cmd.sh
RUN chmod -R 777 /home/qgis/cmd.sh
RUN chown qgis:qgis /home/qgis/cmd.sh

USER qgis
WORKDIR /home/qgis

ENTRYPOINT ["/tini", "--"]

CMD ["/home/qgis/cmd.sh"]
```

- create a file `cmd.sh` with this content:

```
#!/bin/bash

[[ $DEBUG == "1" ]] && env

exec /usr/bin/xvfb-run --auto-servernum --server-num=1 /usr/bin/spawn-fcgi -p 5555␣
↪-n -d /home/qgis -- /usr/lib/cgi-bin/qgis_mapserv.fcgi
```

- build the image with:

```
docker build -f Dockerfile -t qgis-server ./
```

8.1.1 First run

To run the server you will need a QGIS project file. You can use one of yours or pick this sample.

To do so, create a directory `data` within the directory `qgis-server` and copy your file in it. To comply with the following explanations, rename it to `osm.qgs`.

> **Notă**
>
> You may need to add advertised URLs under the *QGIS Server* tab of the *Project* ► *Properties* if the GetCapabilites are broken. For example if your server is exposed on port 8080, you will put this for advertised URL `http://localhost:8080/qgis-server/`. More information available in section *Configure your project* and subsequent.

Now, you can run the server with:

```
docker network create qgis
docker run -d --rm --name qgis-server --net=qgis --hostname=qgis-server \
              -v $(pwd)/data:/data:ro -p 5555:5555 \
              -e "QGIS_PROJECT_FILE=/data/osm.qgs" \
              qgis-server
```

Options used:

- **-d**: run in the background
- **–rm**: remove the container when it is stopped
- **–name**: name of the container to be created
- **–net**: (previously created) sub network
- **–hostname**: container hostname, for later referencing
- **-v**: local data directory to be mounted in the container
- **-p**: host/container port mapping
- **-e**: environment variable to be used in the container

To check, type `docker ps | grep qgis-server` and you should see a line with **qgis-server**:

```
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS       ␣
↪  PORTS                      NAMES
4de8192da76e   qgis-server    "/tini -- /home/qgis…"   3 seconds ago    Up 2 seconds␣
↪  0.0.0.0:5555->5555/tcp     qgis-server
```

8.1.2 Usable sample

As the server is only accepting fastcgi connections, you need an HTTP server that handles this protocol. To do so we have to create a simple Nginx configuration file and start a Nginx image.

Create a file `nginx.conf` in the current directory with this content:

```
server {
  listen 80;
  server_name _;
  location / {
```

(continues on next page)

(continuare din pagina precedentă)

```
    root  /usr/share/nginx/html;
    index index.html index.htm;
  }
  location /qgis-server {
    proxy_buffers 16 16k;
    proxy_buffer_size 16k;
    gzip off;
    include fastcgi_params;
    fastcgi_pass qgis-server:5555;
  }
}
```

And type this command:

```
docker run -d --rm --name nginx --net=qgis --hostname=nginx \
              -v $(pwd)/nginx.conf:/etc/nginx/conf.d/default.conf:ro -p 8080:80 \
              nginx:1.13
```

To check capabilities availability, type in a browser http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities

8.1.3 Cleanup

To cleanup the running images, type:

```
docker stop qgis-server nginx
```

8.2 Docker stacks

The previous method is scriptable, but not easily packageable nor standardized or easily manageable.

To work with a docker image set you could use a docker stack managed by an orchestrator. In a stack, the images are working in the same private network, and you can start / stop the whole stack or deploy the stack to other workers. There are many orchestrators, for example Swarm, Kubernetes and Mesos.

In the following, we will present simple configurations for testing purposes. They are not suitable for production.

8.2.1 Swarm/docker-compose

Docker now has its own orchestrator: Swarm (compatible with docker-compose files). You have to enable it (the Mac version will also work with Linux).

Stack description

Now that you have Swarm working, create the service file (see Deploy to Swarm) `qgis-stack.yaml`:

```
version: '3.7'

services:
  qgis-server:
    # Should use version with utf-8 locale support:
    image: qgis-server:latest
    volumes:
      - REPLACE_WITH_FULL_PATH/data:/data:ro
    environment:
```

(continues on next page)

(continuare din pagina precedentă)

```
      - LANG=en_EN.UTF-8
      - QGIS_PROJECT_FILE=/data/osm.qgs
      - QGIS_SERVER_LOG_LEVEL=0   # INFO (log all requests)
      - DEBUG=1                   # display env before spawning QGIS Server

  nginx:
    image: nginx:1.13
    ports:
      - 8080:80
    volumes:
      - REPLACE_WITH_FULL_PATH/nginx.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
      - qgis-server
```

To deploy (or update) the stack, type:

```
docker stack deploy -c qgis-stack.yaml qgis-stack
```

Check the stack deployment status until you obtain **1/1** in the **replicas** column:

```
docker stack services qgis-stack
```

Something like:

```
ID                  NAME                MODE                REPLICAS            ␣
↪IMAGE                 PORTS
gmx7ewlvwsqt        qgis_nginx          replicated          1/1                 ␣
↪nginx:1.13            *:8080->80/tcp
l0v2e7cl43u3        qgis_qgis-server    replicated          1/1                 ␣
↪qgis-server:latest
```

To check WMS capabilities, type in a web browser http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities

Cleanup

To cleanup, type:

```
docker stack rm qgis-stack
```

8.2.2 Kubernetes

Instalarea

If you have a **Docker Desktop** installation, using Kubernetes (aka k8s) is pretty straight forward: enable k8s.

If not, follow the minikube tutorial or microk8s for Ubuntu.

As Kubernetes installation can be really complex, we will only focus on aspects used by this demo. For further / deeper information, check the official documentation.

microk8s

microk8s needs extra steps: you have to enable the registry and tag the qgis-server image in order to have Kubernetes to find the created images.

First, enable the registry:

```
microk8s enable dashboard dns registry
```

Then, tag and push the image to your newly created registry:

```
docker tag qgis-server 127.0.0.1:32000/qgis-server && docker push 127.0.0.1:32000/
↪qgis-server
```

Finally, add or complete the `/etc/docker/daemon.json` to have your registry **127.0.0.1:32000** listed in the **insecure-registries** field:

```
{
  "insecure-registries": ["127.0.0.1:32000"]
}
```

Creating manifests

Kubernetes describes the objects to deploy in yaml manifests. There are many different kinds, but we will only use deployments (handle pods, i.e. docker images) and services to expose the deployments to internal or external purposes.

Deployment manifests

Create a file `deployments.yaml` with this content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: qgis-server
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      myLabel: qgis-server
  template:
    metadata:
      labels:
        myLabel: qgis-server
    spec:
      containers:
        - name: qgis-server
          image: localhost:32000/qgis-server:latest
          imagePullPolicy: Always
          env:
            - name: LANG
              value: en_EN.UTF-8
            - name: QGIS_PROJECT_FILE
              value: /data/osm.qgs
            - name: QGIS_SERVER_LOG_LEVEL
              value: "0"
            - name: DEBUG
              value: "1"
```

(continues on next page)

(continuare din pagina precedentă)

```
          ports:
            - containerPort: 5555
          volumeMounts:
            - name: qgis-data
              mountPath: /data/
      volumes:
        - name: qgis-data
          hostPath:
            path: REPLACE_WITH_FULL_PATH/data

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: qgis-nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      myLabel: qgis-nginx
  template:
    metadata:
      labels:
        myLabel: qgis-nginx
    spec:
      containers:
        - name: qgis-nginx
          image: nginx:1.13
          ports:
            - containerPort: 80
          volumeMounts:
            - name: nginx-conf
              mountPath: /etc/nginx/conf.d/
      volumes:
        - name: nginx-conf
          configMap:
            name: nginx-configuration

---
kind: ConfigMap
apiVersion: v1
metadata:
  name: nginx-configuration
data:
  nginx.conf: |
    server {
      listen 80;
      server_name _;
      location / {
        root  /usr/share/nginx/html;
        index index.html index.htm;
      }
      location /qgis-server {
        proxy_buffers 16 16k;
        proxy_buffer_size 16k;
        gzip off;
        include fastcgi_params;
        fastcgi_pass qgis-server:5555;
        }
      }
```

Service manifests

Create a file `services.yaml` with this content:

```
apiVersion: v1
kind: Service
metadata:
  name: qgis-server
  namespace: default
spec:
  type: ClusterIP
  selector:
    myLabel: qgis-server
  ports:
    - port: 5555
      targetPort: 5555
---
apiVersion: v1
kind: Service
metadata:
  name: qgis-nginx
  namespace: default
spec:
  type: NodePort
  selector:
    myLabel: qgis-nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
```

Deploying manifests

To deploy the images and services in Kubernetes, one can use the dashboard (click on the **+** on the upper right) or the command line.

> **Notă**
>
> When using the command line with microk8s you will have to prefix each command with *microk8s*.

To deploy or update your manifests:

```
kubectl apply -f ./
```

To check what is currently deployed:

```
kubectl get pods,services,deployment
```

You should obtain something like:

```
NAME                                READY   STATUS    RESTARTS   AGE
pod/qgis-nginx-54845ff6f6-8skp9     1/1     Running   0          27m
pod/qgis-server-75df8ddd89-c7t7s    1/1     Running   0          27m

NAME                         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)      ␣
↪  AGE
service/Kubernetes           ClusterIP   10.152.183.1     <none>        443/TCP      ␣
↪  5h51m
service/qgis-exec-server     ClusterIP   10.152.183.218   <none>        5555/TCP     ␣
```

(continues on next page)

(continuare din pagina precedentă)

```
↪  35m
service/qgis-nginx          NodePort    10.152.183.234   <none>        80:30080/TCP␣
↪  27m
service/qgis-server         ClusterIP   10.152.183.132   <none>        5555/TCP    ␣
↪  27m

NAME                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/qgis-nginx       1/1     1            1           27m
deployment.apps/qgis-server      1/1     1            1           27m
```

To read nginx/qgis logs, type:

```
kubectl logs -f POD_NAME
```

To check WMS capabilities, type in a web browser http://localhost:30080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities

Cleanup

To clean up, type:

```
kubectl delete service/qgis-server service/qgis-nginx deployment/qgis-nginx␣
↪deployment/qgis-server configmap/nginx-configuration
```

8.3 Cloud deployment

Managing your own cluster of servers to handle the deployment of containerized applications, is a complex job. You have to handle multiple issues, such as hardware, bandwidth and security at different levels.

Cloud deployment solutions can be a good alternative when you do not want to focus on infrastructure management.

A cloud deployment may use proprietary mechanisms, but they are also compatible with the stages explained previously (*docker images* and *stack management*).

8.3.1 AWS usecase

With Amazon AWS, through ECS (Elastic Container Service) functionalities, you can use docker-compose or Kubernetes compatible wrappers to manage your stack. You will have to create an image registry for your custom images to be accessible.

To use docker-compose alike functionalities, you need to install the **ecs-cli** client and have proper permissions / roles. Then, with the help of the ecs-cli compose commands, you can reuse the *stack description*.

To use Kubernetes, you can use the AWS web console or the command line tool eksctl and have the proper permissions / roles. Then with a well configured kubectl environment, you can reuse the *Kubernetes manifests*.

CAPITOLUL 9

Frequently Asked Question

- *What are the differences between QGIS Desktop and QGIS Server?*

  QGIS Desktop has a graphical user interface and allows you to create and modify maps. QGIS Server is a server application serving your QGIS project files to end user applications via OGC web services like WMS, WFS, etc..

- *What is OGC?*

  The OGC (Open Geospatial Consortium) is an international not for profit organization committed to making quality open standards for the global geospatial community.

- *Name some other web mapping servers?*

  ArcGIS server, Geoserver, Mapserver, Mapnik etc.

- *How to compare QGIS server to other web mapping servers? (2021/01/01)*

Funcțiuni	QGIS Server	GeoServer	ArcGIS Server
Începând cu	2006	2001	1999
Licență	GPL	GPL	comercial
Suport comercial	Multiple companii	Multiple companii	ESRI și rețeaua sa de furnizori
Technologie	C++/python	Java	C++
Înmagazinare plăcuțe	da	da (via GeoWebCache)	da
3D	Nu	Nu	Da
Querying	FES (2.0) and OGC (1.0) filters	CQL and OGC filters	OGC filters
Report generation	da	da	da
Server administration	yes via third parties (LizMap, QWC2, etc.)	web + API REST	web + API REST
GIS project Layer/symbology edition	complete via dedicated GUI	simple via web interface	complete via dedicated GUI

- *What are the OGC specification versions implemented in QGIS server compared to other web mapping servers? (2021/01/01)*

OGC standards	QGIS Server	GeoServer	ArcGIS Server
WMS (Web Map Service)	1.3.0 - 1.1.1	1.3.0 - 1.1.1	1.3.0 - 1.1.1
WFS (Web Feature Service)	1.1.0 - 1.0.0	2.0.0 - 1.0.0	2.0.0 - 1.0.0
OAPIF (aka WFS3)	1.0.0	nu	nu
WMTS (Web Map Tile Service)	1.0.0	1.0.0	1.0.0
WCS (Web Coverage Service)	1.0.0	2.0.1 - 1.0.0	2.0.1 - 1.0.0
WPS (Web Processing Service)	nu	1.0.0	1.0.0
CSW (Catalogue Service for the Web)	nu	2.0.2	nu
SLD (Styled Layer Descriptor)	da	da	da

- *What is a tile cache?*

  Maps are often static. As most mapping clients render WMS (Web Map Service) data every time they are queried, this can result in unnecessary processing and increased wait times.

  The tile cache optimizes this experience by saving (caching) map images, or tiles, as they are requested, in effect acting as a proxy between client (such as OpenLayers or Google Maps) and server (any WMS-compliant server). As new maps and tiles are requested, QGIS server intercepts these calls and returns pre-rendered tiles if stored, or calls the QGIS engine to render new tiles as necessary. Thus, once tiles are stored, the speed of map rendering increases by many times, creating a much improved user experience.

- *What is PostgreSQL?*

  PostgreSQL is a powerful, open source object-relational database companion for QGIS.

- *What is PostGIS?*

  PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.