2019

# Manual Técnico

*TAPEYTY*

UNIVERSIDAD NACIONAL DE ASUNCIÓN – FACULTAD POLITÉCNICA | San Lorenzo - Paraguay

# I. PRESENTACIÓN

El presente documento fue elaborado por alumnos de la Facultad Politécnica de la Universidad Nacional de Asunción dentro del marco de Trabajo Final de Grado. Contiene un conjunto de procedimientos para la instalación y configuración del Sistema *TapeYty*. El documento está orientado a usuarios con conocimientos de sistemas operativos, administración de bases de datos y administración de servicios.

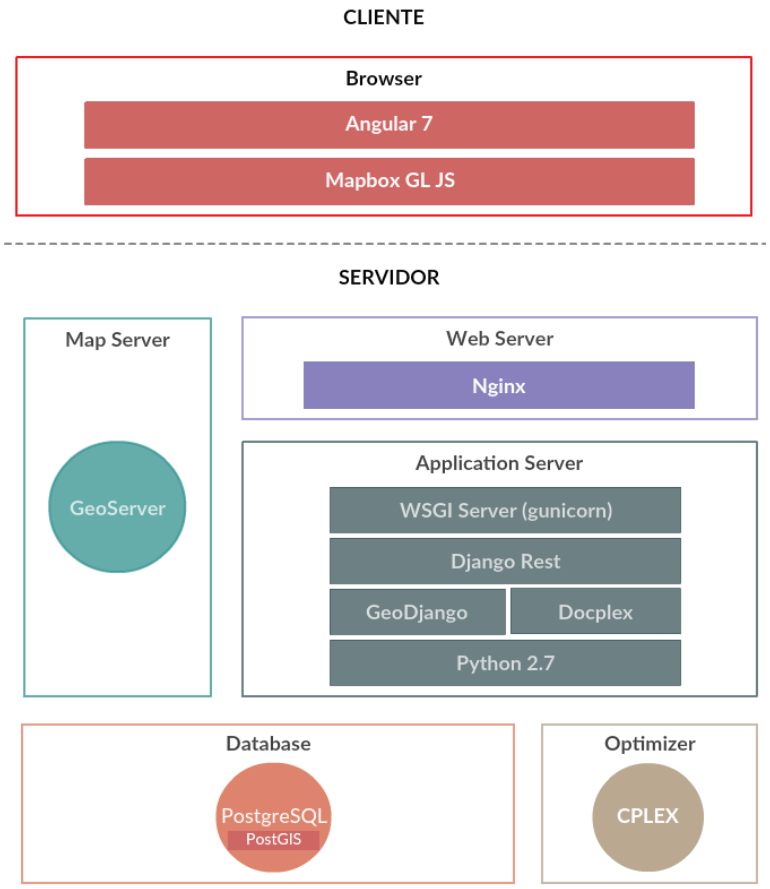
## Contenido

# II. ARQUITECTURA

A continuación se presenta un esquema por niveles de los servicios implementados como solución del Sistema. El lado Servidor es implementado en un único servidor de producción. Una alternativa podría ser tener un servidor dedicado para los mapas (GeoServer) y otro para la aplicación (Nginx). Los usuarios se conectan a través de la intranet al lado Servidor mediante un navegador web.

El servidor web recibe las peticiones del usuario y las envía al de aplicaciones. Los servicios REST reciben la solicitud y validan las credenciales del usuario, en este caso se autentica contra usuarios creados en la base de datos. Durante las siguientes transacciones la comunicación se realiza directamente entre el servidor de aplicaciones y el servidor de base de datos o entre el servidor de aplicaciones y el motor de optimización CPLEX.

En la imagen 1 se esquematiza el contexto general de la solución implementada, con sus componentes internos y externos.


CLIENTE
Browser
Angular 7
Mapbox GL JS
SERVIDOR
Map Server
GeoServer
Web Server
Nginx
Application Server
WSGI Server (gunicorn)
Django Rest
GeoDjango
Docplex
Python 2.7
Database
PostgreSQL
PostGIS
Optimizer
CPLEX

# III. REQUERIMIENTOS

Los requerimientos básicos para un buen desempeño de la aplicación son los siguientes:

- Entorno de producción con Sistema Operativo Linux 64 bits (preferentemente).
- Memoria RAM: 8Gb o más.
- Procesador con 4 núcleos o más.
- Almacenamiento: 100 Mb o más. Preferentemente (250 Mb).

## A. Especificaciones

La aplicación requiere ciertos programas, librerías y utilidades, y que éstos sean de una versión específica. A continuación, se listan cada uno de los programas con sus respectivas versiones:

- Python v2.7.12
- Django v1.11.10
- Postgresql v10
- PostGIS v2.4
- IBM ILOG CPLEX Optimization Studio Community Edition v12.8
- GeoServer v2.13 con su extensión vectortiles-plugin
- Java SE Development Kit 8
- Nginx v1.16.1
- Gunicorn v19.9.0
- Angular v7

# IV. GUÍAS DE INSTALACIÓN

## A. Base de datos Postgresql – PostGIS

a) Instalar por línea de comandos los paquetes de Postgresql
   - *sudo apt-get install python-pip python-dev libpq-dev postgresql-10 postgresql-10-contrib curl*

b) Instalar por línea de comandos los paquetes de PostGIS
   - *sudo apt-get install postgis postgresql-10-postgis-2.4 postgresql-10-postgis-2.4-scripts*

c) Configurar contraseña del usuario de base de datos
   - *sudo su postgres*
   - *psql*
   - *ALTER USER postgres with PASSWORD '[contraseña]';*

d) Crear base de datos
   - *CREATE DATABASE tapeyty;*

e) Importar la base de datos
   - Ir al directorio donde se encuentra el backup. Ej. *cd /tmp*
   - Si se usó pg_dump para el backup: *psql [nombre_tabla] < [archivo_backup]*
   - Si se usa tar o custom para el backup: *pg_restore -d [nombre_tabla] [archivo_backup]*

f) Para iniciar el servicio, reiniciar o parar el servicio de BD respectivamente hacer:
   - *sudo systemctl start postgresql*
   - *sudo systemctl restart postgresql*
   - *sudo systemctl stop postgresql*

## B. IBM ILOG CPLEX Optimization Studio Community Edition

a) Descargar el instalador de CPLEX
   - Descargar instalador IBM ILOG CPLEX Optimization Studio Community Edition. Versión 12.7

b) Ejecutar en la línea de comande en el directorio de CPLEX (se requiere de Python)
   - *sudo ./cplex_studio128.linux-x86-64.bin*

c) Seleccionar el idioma, presionar "1" para aceptar el acuerdo de Licencia

d) Seleccionar la ubicación de instalación por defecto.

**Nota:** Con esto el CPLEX ya se encuentra instalado en el servidor, se deberá posteriormente enlazar agregar al entorno de Python a utilizar en el proyecto.

e) Ir al directorio de instalación de CPLEX e ingresar en Python:
   - *cd /opt/ibm/ILOG/CPLEX_Studio128/cplex/python/2.7/x86-64_linux*

f) Ejecutar en el entorno de Python (o virtualenv utilizado)
   - *sudo python setup.py install*

## C. GeoServer

Es un proyecto abierto que permite brindar los servicios de mapa al Sistema.

a) Descomprimir el GeoServer de TapeYty en el servidor
   - *sudo unzip /tmp/geoserver.zip -d [directorio_servidor]*

b) Comprobar instalación de Java
   - *javac –versión*. Si se despliega algo de Java 8, saltar al paso c)
   - *sudo apt-get install openjdk-8-jdk-headless*

c) Modificar permisos al directorio GeoServer
   - *sudo chown -R [usuario].[ usuario] [directorio_servidor]/geoserver/*

d) Iniciar y probar el servicio
   - *cd [directorio_servidor]/geoserver/bin*
   - *./startup.sh &*

e) Verificar configuración de las capas, fuente de datos en GeoServer.

f) Configurar como servicio del Sistema Operativo
   - Descargar el script desde: https://docs.geoserver.org/latest/en/user/production/linuxscript.html
   - En /etc/default/geoserver

     *USER=geoserver*

     *GEOSERVER_DATA_DIR=/home/$USER/Documentos/geoserver/data_dir*

     *GEOSERVER_HOME=/home/$USER/Documentos/geoserver*

     *JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64*

     *JAVA_OPTS="-Xms128m -Xmx512m"*

g) Cambiar permisos en /etc/default

- *sudo chmod 755 geoserver*

h) Comprobar que funciona el script

- */etc/init.d/geoserver start*

i) Para iniciar el servicio, reiniciar o parar el servicio

- *sudo systemctl start geoserver*
- *sudo systemctl status geoserver*
- *sudo systemctl enable geoserver*

## D. Backend – GeoDjango

a) Instalar como requerimiento de GeoDjango:
   - *sudo apt-get install binutils libproj-dev gdal-bin*

b) Ir al directorio del código Fuente del Proyecto backend y verificar sus configuraciones
   - *cd [directorio_backend]/tapeYty*
   - *vi settings.py*

c) Configurar un entorno virtual
   - *sudo -H pip install virtualenv*
   - *cd [directorio_backend]*
   - *virtualenv tapeytyenv*

d) Instalar paquetes y librerías del proyecto
   - *source tapeytyenv/bin/actívate*
   - *pip install -r requirements.txt*

e) Ejecutar aplicación
   - *cd [directorio_backend]*
   - *python manage.py collectstatic*
   - *python manage.py runserver 0.0.0.0:8000*
   - *gunicorn --bind 0.0.0.0:8000 tapeYty.wsgi*
   - *sudo nano /etc/systemd/system/gunicorn.socket*

     *[Unit]*

     *Description=gunicorn socket*

*[Socket]*

*ListenStream=/run/gunicorn.sock*

*[Install]*

*WantedBy=sockets.target*

- *sudo nano /etc/systemd/system/gunicorn.service*
- En el archivo /etc/systemd/system/gunicorn.service

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target
[Service]
User=[usuario]
Group=www-data
WorkingDirectory= [directorio_backend]
ExecStart=/home/dsu/Documentos/backend/tapeYty-App/tapeytyenv/bin/gunicorn \
          --access-logfile - \
          --workers 9 \
          --bind unix:/run/gunicorn.sock \
          --timeout 300 \
          tapeYty.wsgi:application
[Install]
WantedBy=multi-user.target
```

- Habilitar socket

f) Probar y activar Gateway gunicorn

- *sudo systemctl start gunicorn.socket*
- *sudo systemctl enable gunicorn.socket*

- *sudo systemctl status gunicorn.socket*
- *sudo systemctl status gunicorn*
- *curl --unix-socket /run/gunicorn.sock localhost*
- *sudo systemctl status gunicorn*
- *sudo systemctl daemon-reload*
- *sudo systemctl restart gunicorn*

**Notas**: Para más detalle ver el siguiente enlace: https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-18-04.

## E. Servidor Web – Nginx

a) Instalar nginx:

- *sudo apt update*
- *sudo apt install nginx*
- *systemctl status nginx*
- Verificar y habilitar IP y puertos 80/443

b) Agregar a la propiedad http:

- *proxy_connect_timeout 300s;*
- *proxy_read_timeout 300s;*

c) Configurar proxy de aplicación backend

- Agregar al archivo /etc/nginx/sites-available

```
location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
}
```

d) Reiniciar nginx

- *sudo systemctl restart nginx*

## F. Frontend – Angular

A partir del código Fuente seguir los siguientes pasos:

a) Generar el código compilado del proyecto Frontend Angular:
   - *ng build --prod tapeyty*

b) Subir a nginx
   - Comprimir dist/tapeyty
   - *sudo rm -rf /var/www/html/**
   - *sudo apt-get install unzip*
   - *sudo unzip /tmp/tapeyty.zip -d /var/www/html/*
   - *cd /var/www/html/*
   - *sudo cp -R tapeyty/* .En /etc/nginx/nginx.conf*. Esto ya que los servicios de optimización pueden tener una duración mayor a la predefinida.

**Notas:** Para soportar mayor cantidad de usuarios concurrentes en la aplicación, se deben realizar ajustes en el servidor:

1. En el archivo */etc/systemd/system/gunicorn.service*
   a. *--workers 9.* El valor debe ajustarse al número de núcleos del servidor. Se recomiendo colocar como valor el resultado de (2 * {cant_núcleos}) + 1.
2. En el archivo */proc/sys/net/core/somaxconn*
   a. Modificar el valor máximo de conexiones de sockets mayor. Por defecto es 128.
3. En el archivo */etc/nginx/nginx.conf*
   a. *worker_processes 9*
   b. *events {*
      *worker_connections 8192*
      *}*