Implementación

En esta sección se detallan instrucciones para poder implementar la plataforma en producción.

Este documento apunta a *operadores y administradores de sistemas* que necesiten poner en funcionamiento la plataforma en un servidor, y por lo tanto se asumen conocimientos básciso de manejo de terminal, instalación de paquetes y configuración de servicios.

En los repositorios se incluyen ejemplos de archivos de configuración cuando es apropiado, pero siempre se deben tomar como ejemplos y ajustar según sea necesario a su criterio.

Requerimientos de SO

Se asume que la plataforma va a ejecutarse en un entorno con la distribución **Ubuntu 18.04** instalada. Para otros sistemas operativos, será necesario consultar la documentación correspondiente de las dependencias.

Dependencias

La plataforma consiste en un backend, implementado sobre Python, y un frontend implementado en Nodejs. Utiliza PostgreSQL como base de datos, y Redis como cache y base de datos para la cola de trabajos.

A continuación se detallan instrucciones para instalar y configurar estas dependencias.

PostgreSQL y extensiones (TimescaleDB y PostGIS)

Sólo para Ubuntu 18.04

Las instrucciones están pensadas para Ubuntu 18.04. Puede verificar la guía de instalación de TimescaleDB si está utilizando otro sistema operativo.

Agregue el siguiente repositorio de PostgreSQL para obtener la últimas versiones de los pquetes de PostgreSQL (esto es necesario para versiones de Ubuntu menores a 19.04).

```
# `lsb_release -c -s` should return the correct codename of your OS
echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -c -s)-pgdg
main" | sudo tee /etc/apt/sources.list.d/pgdg.list
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo
```

```
apt-key add -
sudo apt-get update
```

Agregue el repositorio de TimescaleDB y luego instale TimescaleDB, que al hacerlo automaticamente descargará e instalará también la versión correcta del servidor de PostgreSQL.

```
# Add TimescaleDBs PPA
sudo add-apt-repository -y ppa:timescale/timescaledb-ppa
sudo apt-get update

# Now install appropriate package for PG version
sudo apt install timescaledb-postgresql-11
```

Es recomendable que utilice la herramienta `timescaledb-tune` para ajustar y optimizar la base de datos para TimescaleDB.

```
sudo timescaledb-tune
```

Ahora, instale la extensión PostGIS 3 para esta versión de PostgreSQL.

```
sudo apt-get install -y postgresql-11-postgis-3
```

Finalmente, reinicie la instancia de PostgreSQL.

```
sudo service postgresql restart
```

Python, Redis, GDAL

Instale Python y otras dependencias como GDAL y el servidor de Redis.

```
sudo apt-get install -y \
  build-essential \
  git \
  gdal-bin \
  gettext \
  libgdal-dev \
  libpq-dev \
  libproj-dev \
  python3 \
  python3-dev \
  python3-pip \
  redis-server
```

Nginx

Nginx se utiliza para servir tanto el backend (API y administrador) como el frontend (sitio web de la plataforma).

Instale Nginx desde los repositorios de Ubuntu:

```
sudo apt install nginx
```

Backend

Instalación

Primero clone el repositorio del backend.

```
git clone https://github.com/undp/satlomas-back.git
cd satlomas-back/
```

Para instalar las dependencias correctas del backend, debe instalar el manejador de paquetes Pipenv.

```
sudo -H pip3 install -U pipenv
```

Luego, ejecute lo siguiente para instalar todas las dependencias necesarias. Esto creará un entorno virtual para este proyecto.

```
pipenv install
```

Una vez que finalize la instalación, estará listo para configurar el backend.

Configuración

Base de datos

Cree un rol de superusuario para el usuario que está utilizando actualmente. Si sabe que va a ejecutar PostgreSQL con otros usuario, por favor reemplaze la variable `$USER` como corresponde.

```
sudo -u postgres createuser -s $USER
```

Cree la base de datos.

```
createdb satlomas
```

Configure una contraseña para el usuario que acaba de crear. Ejecute el siguiente comando *reemplazando* `foobar` por una contraseña única:

> **Contraseña**
>
> Es muy importante que defina una contraseña única y dificil de adivinar.

```
psql satlomas -c "ALTER USER $USER WITH PASSWORD 'foobar'"
```

Agregue las extensiones TimescaleDB y PostGIS a la base de datos creada.

```
psql satlomas -c "CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE"
psql satlomas -c "CREATE EXTENSION IF NOT EXISTS postgis CASCADE"
```

Registración en Copernicus Open Access Hub

Para que la plataforma pueda descargar los productos del satélite Sentinel-1 y Sentinel-2 de la ESA, es necesario estar registrado en Copernicus Open Access Hub.

Para registrase, ingrese aquí. Tome nota del **nombre de usuario** y **contraseña** dado que será necesario al momento de configurar el backend.

Registración en NASA EarthData

Para descargar los productos de MODIS VI, es necesario estar registrado en NASA EarthData.

Puede registrarse ingresando aquí. Tome nota del **nombre de usuario** y **contraseña** dado que será necesario al momento de configurar el backend.

Variables de entorno

Copie `env.sample` y guarde un archivo nuevo llamada `.env`.

```
cp env.sample .env
```

Ahora edite `.env` según sus necesidades. El archivo `env.sample` es un ejemplo con valores por defecto para algunas variables, pero algunas de ellas debe completarlas. A continuación se describen las variables que deben ser configuradas para el funcionamiento de la plataforma.

Variable	Descripción
`SECRET_KEY`	String único de caracteres alfanuméricos, utilizado para firmas criptográficas. Puede generar uno aquí.

Variable	Descripción
`ALLOWED_HOSTS`	Lista de *hosts* habilitados. Debería ingresar el dominio y/o IP pública del servidor
`DB_USER`	Usuario de la BD (debería ser el nombre del usuario actual)
`DB_PASSWORD`	Contraseña de la BD (la contraseña que definió antes)
`SCIHUB_USER`	Usuario de SciHub
`SCIHUB_PASS`	Contraseña de SciHub
`MODIS_USER`	Usuario de EarthData
`MODIS_PASS`	Contraseña de EarthData

Inicialización

Ingrese al entorno virtual creado por Pipenv, y proceda con la inicialización de la base de datos.

```
pipenv shell
./manage.py migrate   # Correr las migraciones
```

Antes de continuar, debería generar los archivos estáticos, necesarios para el panel de administrador:

```
./manage.py collectstatic
```

Finalmente, cree un *superusuario* para SatLomas. Este será el primer usuario administrador, con el que podrá registrar nuevos administradores o usuarios.

```
./manage.py createsuperuser
```

Luego de seguir las instrucciones del comando, tendrá un usuario y contraseña para ingresar al administrador del backend.

Frontend

Instalación

Primero debe instalar Node v12. Ejecute los siguientes comandos:

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Luego, clone el repositorio del frontend:

```
git clone https://github.com/undp/satlomas-front.git
cd satlomas-front/
```

Instale todas las dependencias con `npm`:

```
npm install
```

Finalmente, ejecute el siguiente comando para generar un *build* del sitio web:

```
npm run build
```

Configuración de servicios

Más información

Esta sección de la documentación está basada en la guía How To Set Up Django with Postgres, Nginx, and Gunicorn on Ubuntu 18.04, de Digital Ocean. Para más información para depurar los servicios, puede consultarla.

Backend

Comience creando un nuevo archivo para el socket del servicio del backend:

```
sudo nano /etc/systemd/system/gunicorn.socket
```

Copie lo siguiente dentro del archivo:

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock
```

```
[Install]
WantedBy=sockets.target
```

Guarde y ciérrelo.

Para el siguiente paso es necesario que verifique la ruta del entorno virtual del backend. Desde el directorio del código del backend, ejecute el siguiente comando:

```
pipenv --venv
```

Tome nota de la ruta que devuelve el comando.

Ahora, cree un nuevo archivo para el servicio. El nombre de archivo del servicio debería coincidir con el del socket, a excepción de la extensión:

```
sudo nano /etc/systemd/system/gunicorn.service
```

Copie el siguiente contenido:

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target

[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/satlomas-back
ExecStart=/home/ubuntu/.local/share/virtualenvs/satlomas-back-XVGhZdP0/bin/
gunicorn \
          --access-logfile - \
          --timeout 600 \
          --workers 3 \
          --bind unix:/run/gunicorn.sock \
          satlomas.wsgi:application

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

Ajuste `WorkingDirectory` a la ruta *absoluta* del repositorio clonado del backend. Luego, ajuste la ruta absoluta de `ExecStart` a la del entorno virtual de Pipenv, agregando al final `/bin/gunicorn`. También será necesario que defina el `User` y `Group` con el que se ejecutará el proceso. Este usuario debería tener permisos para poder acceder al repositorio del backend.

Puede ajustar también la cantidad de *workers* y el timeout permitido, o agregar otras opciones de Gunicorn si es necesario.

Finalmente, puede iniciar y activar el servicio. Esto creará un archivo `/run/gunicorn.sock` y se iniciará el servidor. Cuando se realiza una conexión al socket, systemd automaticamente inicia el servicio `gunicorn.service` para manejar el pedido.

```
sudo systemctl start gunicorn.socket
sudo systemctl enable gunicorn.socket
```

Frontend

De manera similar, para el servidor web del frontend será administrador por systemd.

Cree un nuevo archivo para el servicio del frontend:

```
sudo nano /etc/systemd/system/satlomas-front.service
```

y copie el siguiente contenido:

```
[Unit]
Description=Platform frontend next.js app
After=network.target

[Service]
Environment=NODE_ENV=production
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/satlomas-front
ExecStart=/usr/bin/node server.js

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

Ajuste `WorkingDirectory` a la ruta *absoluta* del repositorio clonado del frontend. También será necesario que defina el `User` y `Group` con el que se ejecutará el proceso. Este usuario debería tener permisos para poder acceder al repositorio del backend.

Finalmente, puede iniciar y activar el servicio.

```
sudo systemctl start satlomas-front.service
sudo systemctl enable satlomas-front.service
```

Nginx

A modo ejemplo, se presentan dos archivos de configuración para Nginx, para servir el backend y frontend respectivamente.

En este ejemplo, se asume que se tienen registrado el dominio `satlomas.com`, y dos registros A para los subdominios:

1. `api.satlomas.com`: Backend
2. `app.satlomas.com`: Frontend

Cree un nuevo archivo `/etc/nginx/sites-available/satlomas-back`

```
sudo nano /etc/nginx/sites-available/satlomas-back
```

y copie dentro de éste el siguiente contenido:

```
server {
  server_name api.satlomas.com;

  location = /favicon.ico { access_log off; log_not_found off; }
  location /static/ {
    root /home/ubuntu/satlomas-back;
  }

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

    proxy_connect_timeout       600;
    proxy_send_timeout          600;
    proxy_read_timeout          600;
    send_timeout                600;
  }
}
```

Ajuste `server_name` al subdominio que corresponda al backend, y `root` a la ruta absoluta del repositorio del backend (en este caso, el repositorio fue clonado dentro del directorio *home* del usuario `ubuntu`).

Luego cree otro archivo `/etc/nginx/sites-available/satlomas-front`

```
sudo nano /etc/nginx/sites-available/satlomas-front
```

y copie lo siguiente:

```
server {
    server_name app.satlomas.com;

    location /static/ {
        root /home/ubuntu/satlomas-front/public;
    }

    location /_next/static/ {
        alias /home/ubuntu/satlomas-front/.next/static/;
    }
```

```
    location / {
      # default port, could be changed if you use next with custom server
      proxy_pass http://localhost:3000;

      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection 'upgrade';
      proxy_set_header Host $host;
      proxy_cache_bypass $http_upgrade;

      # if you have try_files like this, remove it from our block
      # otherwise next app will not work properly
      # try_files $uri $uri/ =404;
    }
}
```

Nuevamente, ajuste `server_name` al subdominio correspndiente para el frontend, y las ocurrencias de `/home/ubuntu/satlomas-front` a la ruta absoluta del repositorio del frontend.

Finalmente, haga enlaces simbólicos de estos dos archivos al directorio `/etc/nginx/sites-enabled/` para habilitar los sitios:

```
sudo ln -s /etc/nginx/sites-available/satlomas-back /etc/nginx/sites-enabled/
sudo ln -s /etc/nginx/sites-available/satlomas-front /etc/nginx/sites-enabled/
```

y reinicie Nginx:

```
sudo systemctl restart nginx
```

Con esto concluye la implementación de la plataforma. Podrá acceder al backend desde `http://api.satlomas.com/`, y al frontend desde `http://app.satlomas.com/` (suponiendo que configuró los sitios con esos subdominios).

Configuración SSL

Es posible configurar SSL a través de los certificados gratuitos de Let's Encrypt.

> **Instrucciones para Ubuntu 18.04**
>
> Las siguientes instrucciones están basadas en la guía de Certbot para la distribución de Ubuntu 18.04 y Nginx. Si desea ver más información puede acceder aquí.

Deberá agregar el PPA de Certbot a la lista de repositorios. Para esto, ejecute los siguientes comandos:

```
sudo apt-get update
sudo apt-get install -y software-properties-common
```

```
sudo add-apt-repository -y universe
sudo add-apt-repository -y ppa:certbot/certbot
sudo apt-get update
```

Ejecute este comando para instalar Certbot:

```
sudo apt-get install -y certbot python3-certbot-nginx
```

Ahora ejecute este comando para obtener un certificado y hacer que Certbot edite los archivos de configuración de Nginx de los sitios habilitados automáticamente.

```
sudo certbot --nginx
```

El certificado se debería actualizar automaticamente a través del servicio de Certbot.