Gobierno Bolivariano de Venezuela

Manual de Instalación

Sistema Automatizado de Planificación Integral Comunal

Noviembre, 2018

Centro Nacional de Desarrollo e Investigación en Tecnologías Libres

CENDITEL

Ente adscrito al Ministerio del Poder Popular para Educación Universitaria Ciencia y Tecnología

PUBLICADO POR EL CENTRO NACIONAL DE DESARROLLO E INVESTIGACIÓN EN TECNOLOGÍAS LIBRES

cenditel.gob.ve

Autor: Lully Troconis - ltroconis@cenditel.gob.ve

Índice general

I Parte I

1 Presentación 7
1.1 SAPIC: Sistema Automatizado de Planificación Integral Comunal 7

II Parte II

2 Instalación de requerimientos 11
2.1 Requerimientos previos 11
2.2 Control de versiones 11
2.3 Entorno Virtual 11
2.4 Instalación de requerimientos del proyecto 12
2.5 Base de datos y migración de modelos 12

III Parte III

3 Configuración motor de aplicación *uWSGI* 17
3.1 Descripción 17
3.2 Procedimiento 17

IV Parte IV

4 Configuración servidor web *Nginx* 21
4.1 Descripción 21
4.2 Procedimiento 21

V Parte V

5 Configuración fichero setttings.py 25
5.1 Descripción 25
5.2 Procedimiento 25

Parte I

1 Presentación 7
1.1 SAPIC: Sistema Automatizado de Planificación Integral Comunal

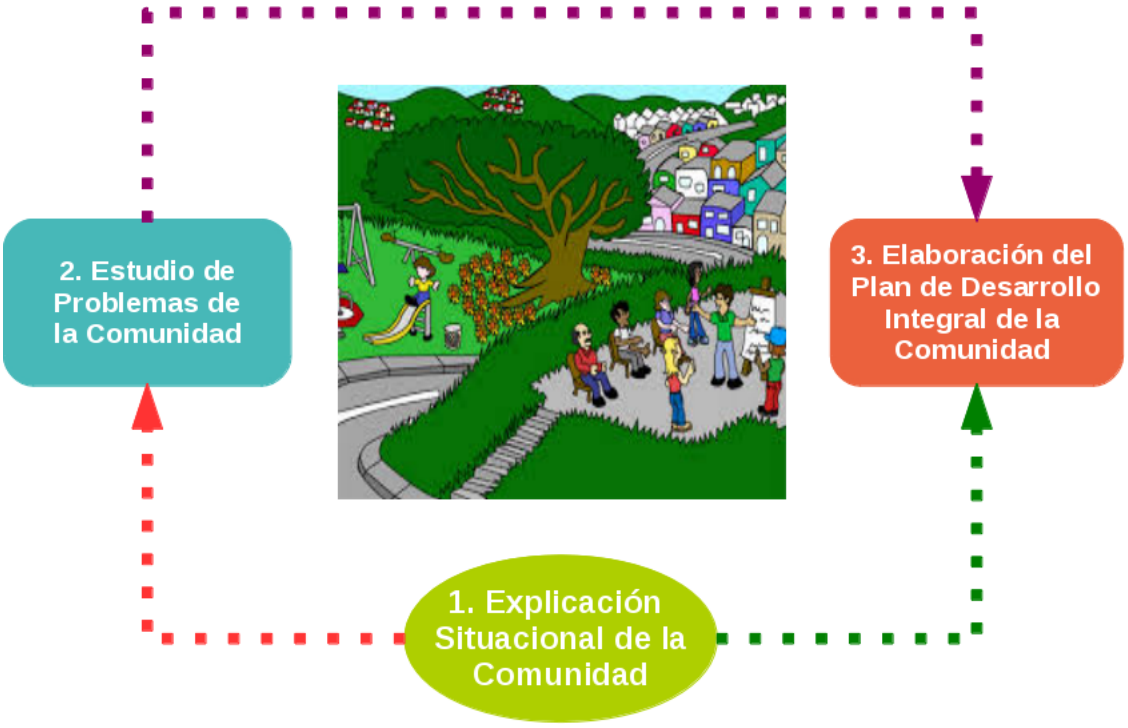
1. Presentación

1.1 SAPIC: Sistema Automatizado de Planificación Integral Comunal

Desde CENDITEL se abordó desde el punto de vista conceptual una propuesta metodológica para la planificación, para lo cual fue necesario realizar una investigación exploratoria de los métodos implementados por algunas organizaciones comunitarias, además de realizar una revisión del marco jurídico que sustenta el accionar de estas organizaciones, por último se sistematizaron los encuentros entre CENDITEL y diversas comunas y consejos comunales, estos elementos en conjunto con la experiencia acumulada del equipo de investigación adquirida por el desarrollo de la Metodología de Planificación Estratégica Situacional para el Sector Público, permitieron elaborar la Metodología de Planificación Estratégica Situacional para Espacios Comunales (MPESEC), en la cual se proponen un conjunto de actividades orientadas a entender la situación actual de las comunidades, en función de su entorno físico, económico, social y político, a fin de identificar problemas y potencialidades en base a las cuales se propongan acciones concretas para beneficio de estas comunidades.

Con base a la metodología diseñada desde CENDITEL y en función a las necesidades de las organizaciones sociales, se desarrolló una aplicación web denominada SAPIC, que tiene como objetivos automatizar y apoyar los procesos de planificación que llevan a cabo las distintas organizaciones sociales, centrando la atención en el análisis situacional y el estudio de los problemas de la organización.

El sistema está desarrollado en Python 3 utilizando como framework Django versión 1.11



Parte II

2 Instalación de requerimientos 11
2.1 Requerimientos previos
2.2 Control de versiones
2.3 Entorno Virtual
2.4 Instalación de requerimientos del proyecto
2.5 Base de datos y migración de modelos

2. Instalación de requerimientos

2.1 Requerimientos previos

Requisitos del sistema

- Debian ≥ 8
- Django v.1.11
- Python v.3.x
- PostgreSQL v9.x
- Postgis

2.2 Control de versiones

El control de versiones utilizado para gestionar el proyecto será git, por lo que se debe proceder a instalarlo en caso de que no lo este.

2.2.1. Ejecutar como superusuario:
aptitude install git

2.2.2. Copiar el repositorio del proyecto, como el repositorio cuenta con certificados autofirmados se necesita remover la comprobación de los mismos. Ejecutar como usuario regular:
$ export GIT_SSL_NO_VERIFY=1

2.2.3. Clonar el repositorio mediante el comando git clone con la url siguiente:
$ git clone https://planificacion.cenditel.gob.ve/scm/git/sapic.git

2.3 Entorno Virtual

El proyecto está desarrollado con el lenguaje de programación *Python v.3*, se debe instalar *Python v3.4.2* (*Python v3.5* en Debian Stretch) ya que por defecto la versión utilizada es la 2.7, teniendo un entorno directamente en la versión 3 facilitará la instalación de paquetes sin que éstos afecten el sistema operativo en general.

2.3.1. Clonar el repositorio mediante el comando git clone con la url siguiente:
$ git clone https://planificacion.cenditel.gob.ve/scm/git/sapic.git

2.3.2. Con los siguientes comandos como usuario root instalar *Python* y *pip*.
Debian Jessie:
aptitude install python3.4 python3-pip python3.4-dev python3-setuptools
aptitude install python3-virtualenv virtualenvwrapper

Debian Stretch:
aptitude install python3.5 python3-pip python3.5-dev python3-setuptools
aptitude install python3-virtualenv virtualenvwrapper

2.3.3. Salir del modo root y como usuario regular crear el entorno virtual:
$ virtualenv - -python=/usr/bin/python3 sapic

2.4 Instalación de requerimientos del proyecto

2.4.1. Para activar el entorno virtual creado ejecute el comando a continuación:
$ source ruta/entorno/virtual/bin/activate

2.4.2. El entorno activo se verá como se muestra en la línea siguiente:
(sapic) usuario@Usuario: $

2.4.3. Entrar en la carpeta raíz del proyecto:
(sapic) usuario@Usuario: $ cd /ruta/sapic
(sapic) usuario@Usuario: /ruta/sapic$

2.4.4. Instalar los requerimientos del proyecto con el siguiente comando:
(sapic) usuario@Usuario: /ruta/sapic$ pip install -r requerimientos.txt

Nota:Si hay problemas en la instalación del paquete lxml==3.6.0 descrito en el fichero requerimientos.txt es necesario instalar los siguientes paquetes como usuario root:
aptitude install python3-lxml libxml2-dev libxslt-dev python-dev

Instalar las dependencias de compilación mediante:
apt-get build-dep python3-lxml

Se deberá replicar la instalación de los requerimientos. Una vez estén todos los requerimientos instalados crear la base de datos.

2.5 Base de datos y migración de modelos

El manejador de base de datos utilizado en el sistema es *postgreSQL*, por lo tanto se debe instalar y crear la base de datos de la siguiente manera si se usa desde la consola de postgres. Instalar además *postgis* para el uso de una base de datos georeferenciada:

2.5.1. Instalar:
aptitude install postgresql-x.x-postgis-x.x postgis

2.5.2. Iniciar como usuario postgres, crear el usuario y la base de datos:
su postgres

2.5.3. Ejecutar para iniciar la consola de postgres:
su postgres
postgres@Usuario: $ psql

Se verá de la siguiente forma:
postgres=#

2.5.4. Crear el usuario, la base de datos y crear la extensión de *postgis* en la base de datos ya creada:
postgres=# CREATE USER sapic WITH ENCRYPTED PASSWORD 'password' CREATEDB;
postgres=# CREATE DATABASE sapic OWNER=sapic ENCODING='UTF-8';
postgres=# \q

postgres@Usuario: $ psql sapic
sapic=# CREATE EXTENSION postgis;

2.5.5. Copiar el fichero a uno nuevo y editar:
cp /ruta/sapic/sapic/settings_default.py/ruta/sapic/sapic/settings.py

Modificar los datos referidos a la base en el fichero settings.py
vim /ruta/sapic/sapic/settings.py

El archivo lucirá similar a:

```
# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'sapic',
        'USER': 'sapic',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '5432',
        'ATOMIC_REQUESTS': True,
    }
}
```

2.5.6. Ya se tiene la base de datos completamente lista por lo tanto se procede a migrar los modelos del proyecto ejecutando los siguientes comandos en el entorno virtual y dentro del directorio del proyecto:
(sapic)$ python manage.py makemigrations explicacion_situacional organizaciones users utils
(sapic)$ python manage.py makemigrations
(sapic)$ python manage.py migrate

2.5.7. Cargar la data inicial del proyecto:
Asegúrese de que los modelos se hayan migrado en base de datos y ejecute los siguientes comandos para cargar la data inicial del proyecto:

Carga los grupos de usuarios, permisos de los usuarios y el superusuario:
(sapic)$ python manage.py loaddata fixtures/initial_data_auth.json

Carga los datos de los estados, municipios, parroquias y utilidades de la aplicación:
(sapic)$ python manage.py loaddata fixtures/initial_data_utils.json

Carga los datos iniciales del usuario admin de la aplicación:
(sapic)$ python manage.py loaddata fixtures/initial_data_users.json

Carga los datos iniciales de las preguntas de las consulta relacionadas a la explicación situacional:
(sapic)$ python manage.py loaddata fixtures/initial_tipo_pregunta.json

Carga los datos iniciales de las características de las consulta relacionadas a la explicación situacional:

```
(sapic)$ python manage.py loaddata fixtures/initial_data_caracterizacion.json
```

Carga los datos iniciales de las consultas relacionadas a la explicación situacional:

```
(sapic)$ python manage.py loaddata fixtures/initial_data_consultas.json
```

Carga los datos iniciales de las preguntas relacionadas a las consultas de la explicación situacional:

```
(sapic)$ python manage.py loaddata fixtures/initial_data_preguntas.json
```

Carga los datos iniciales de las opciones de selección simple o múltiples relacionadas a las preguntas de las consultas:

```
(sapic)$ python manage.py loaddata fixtures/initial_data_opciones.json
```

Carga los datos iniciales del rol de un comité ejecutivo de una organización:

```
(sapic)$ python manage.py loaddata fixtures/initial_data_comiteejecutivo.json
```

Parte III

3 Configuración motor de aplicación *uWS-GI* 17
3.1 Descripción
3.2 Procedimiento

3. Configuración motor de aplicación *uWSGI*

3.1 Descripción

El software *uWSGI* es utilizado como el motor de las aplicaciones, es la puerta de enlace entre servidor web y la aplicación.

3.2 Procedimiento

3.2.1. Instalar los siguientes paquetes:

```
# aptitude install uwsgi uwsgi-plugin-python3
```

3.2.2. Crear un archivo con este contenido en la ubicación siguiente:

```
# vim /etc/uwsgi/apps-available/sapic.xml
```

```
<uwsgi>
        <threads>100</threads>
        <workers>2</workers>
        <master/>
        <chmod-socket>666</chmod-socket>
        <home>/ruta/entorno/virtual/sapic/</home>
        <env>DJANGO\_SETTINGS\_MODULE=sapic.settings</env>
        <uid>www-data</uid>
        <gid>www-data</gid>
        <log-x-forwarded-for/>
        <post-buffering>4096</post-buffering>
        <buffer-size>32768</buffer-size>
        <max-requests>1000</max-requests>
        <chdir>/ruta/proyecto/sapic</chdir>
        <pythonpath>..</pythonpath>
        <module>wsgi</module>
        <plugins>python3</plugins>
<uwsgi>
```

3.2.3. Crear el enlace simbólico del archivo creado a la ruta /etc/uwsgi/apps-enabled/

```
# ln -s /etc/uwsgi/apps-available/sapic.xml /etc/uwsgi/apps-enabled/
```

3.2.4. Reiniciar *uwsgi*

```
# systemctl restart uwsgi
```

IV Parte IV

4 Configuración servidor web *Nginx* . . . 21
4.1 Descripción
4.2 Procedimiento

4. Configuración servidor web *Nginx*

4.1 Descripción

Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico (IMAP/POP3). En este caso, este software es usado como servidor web, el cual se comunica con el motor de aplicaciones *uWSGI*.

4.2 Procedimiento

4.2.1. Instalar el software de *nginx* desde el repositorio de Debian:
aptitude install nginx

4.2.2. Configurar el sitio por defecto de *nginx*. Colocar las siguientes directivas en /etc/nginx/sites-available/default. Se recomienda hacer una copia del archivo, mantener la original y editar la copia:

```
server {
    listen 80; ## listen for ipv4
    server_name _;
    return 444;
}
```

4.2.3. Configurar el sitio para la aplicación SAPIC. Crear el archivo /etc/nginx/sites-available/sapic con el siguiente contenido:

```
server {
        listen IP:80 ;

        server_name nombre-dominio;

        access_log  /var/log/nginx/sapic.access.log combined;
        error_log   /var/log/nginx/sapic.error.log error;

        location ~ ^/static/(.+)$ {
                alias /ruta/al/proyecto/sapic/static/$1;
        }

        location ~ ^/ {
                uwsgi_pass unix:/var/run/uwsgi/app/sapic/socket;
                include uwsgi_params;
                uwsgi_param     UWSGI_SCHEME    $http_x_forwarded_protocol;
                uwsgi_param     SCRIPT_NAME     /;
                uwsgi_intercept_errors off;
                uwsgi_read_timeout 600;
```

```
        }
}
```

4.2.4. Establecer la activación automática creando un enlace simbólico en el directorio sites-enabled:
ln -s /etc/nginx/sites-available/sapic /etc/nginx/sites-enabled/

4.2.5. Verificar el reporte de errores en el arranque y reiniciar el servicio:
nginx -t
systemctl restart nginx

Parte V

5 Configuración fichero setttings.py 25
5.1 Descripción
5.2 Procedimiento

5. Configuración fichero setttings.py

5.1 Descripción

Para concluir el despliegue del sistema resta configurar algunas directivas en el fichero settings.py del proyecto.

5.2 Procedimiento

5.2.1. Modificar el settings.py con el editor de su preferencia, cambiar directiva DEBUG = True a False:
vim /ruta/sapic/sapic/settings.py

Ir a la línea y modificar a:
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

Agregar el administrador o los administradores del servidor. Si ocurre un error en el sistema, éste enviará un correo a los administradores con el error:

```
## Administradores del sistema
ADMINS = [
    ('Nombre y Apellido', 'correo@administrador.com'),
]
```

Agregar las siguientes directivas:
BASE_URL = '/'
STATIC_URL = BASE_URL + 'static/'

Agregar también la configuración de correo vía SMTP:
Configuración de variables para el envío de correo electrónico
Nombre del Servidor de correo SMTP
EMAIL_USE_TLS = False
Puerto del Servidor de correo SMTP
EMAIL_PORT = 25
EMAIL_HOST = 'localhost'
Dirección de correo electronico de quien envía
EMAIL_FROM = 'sapic@correo.com'
SERVER_EMAIL = 'sapic@correo.com'
DEFAULT_FROM_EMAIL = 'sapic@correo.com'

5.2.2. Reiniciar *uwsgi*
systemctl restart uwsgi

5.2.3. Ir al navegador con el dominio en el cual se ha configurado en el servidor.