Manual de instalación

En este manual se detalla lo necesario para poner en producción el proyecto. Se partirá desde un sistema operativo Linux como Ubuntu Server.

1. Instalación de dependencias.

Para que la aplicación funcione es necesario instalar el paquete pip y unzip.

```
$ apt-get install python-pip unzip apache2 libapache2-wsgi git
```

Una vez instalado estos dos paquetes instalamos las sisguientes dependencias.

Pip :	Apt-get:
Django==1.11.11	pdftk
django-extensions==2.0.5	poppker-utils
django-shell-plus==1.1.7	
djangorestframework==3.8.2	
pdfminer==20140328	
Pillow==5.0.0	
psycopg2==2.7.4	
PyPDF2==1.26.0	
xhtml2pdf==0.2.1	

2. Copia del proyecto.

Nos descargamos el proyecto de GitHub :

```
$ git clone  https://github.com/edtlopez/ApercibimientosAuto
```

Copiamos el proyecto dentro del directorio /var/www de apache. Y eliminamos los archivos html por defecto.

```
$ rm /var/www/*
$ cp -r ApercibimientosAuto/DjangoProyecto/* /var/www
```

3. Configuración de apache.

Modificamos el fichero 000-default.conf que se encuentra en el directorio /etc/apache2/sites-available. Y añadimos las siguientes lineas.

```
WSGIScriptAlias / /var/www/WebGada/wsgi.py
WSGIDaemonProcess gada python-home=/var/www
WSGIProcessGroup gada



Alias /static /var/www/pdf/static

 <Directory  /var/www/WebGada>
 <Files wsgi.py>
 Require all granted
 </Files>
 </Directory>

 <Directory /var/www/pdf/static>
 Require all granted
 </Directory>
```

Una vez configurado podemos ver la linea WSGIProcessGroup la cual indica el nombre del grupo que tendrá acceso a las carpetas por lo que es necesario crear un dicho grupo y poner los permisos correspondientes a estos.

```
$ groupadd gada
$ chown www-data:gada -R /var/www
```

3.2 Configuración SSL.

Para activar SSL en apache habra que activar el modulo SSL de apache y configurar el fichero **ssl-default.conf** con la configuracion del fichero anterior a la que se le añadira las siquientes lineas.

```
 SSLEngine on
 SSLCertificateFile      /etc/apache2/cert/server.crt
 SSLCertificateKeyFile /etc/apache2/cert/server.key
```

Donde inficaremos la ubicación de nuestro certificado y nuestra clave privada. Tambien es necesario modificar el usuario del proceso WSGI este corre con el usuario “gada” y habra que ponerlo como “www-data”.

```
 WSGIDaemonProcess www-data python-home=/var/www/
```

```
WSGIProcessGroup www-data
```

4. Modificar variables de entorno.

Apache posee un fechero en /etc/apache2 llamado envvars este contiene unas variables las cuales se carga cuando se ejecuta el proceso de apache, dentro de este archivo es necesario añadir estas dos.

```
export LANG='es_ES.UTF-8'
export LC_ALL='es_ES.UTF-8'
```

Esto ara que el interprete de python funcione de manera predeterminada con UTF-8 y no tengamos problemas con los caracteres raros.

5. Fin de configuración de apache.

Una vez echo esto ya temenos configurado nuestro servidor web solo es necesario reiniciar el servicio apache2.

```
$ service apache2 restart
```

6. Configuración Base dedatos (Postgres).

En este caso se ha decidido usar una base de datos en postgres dentro de un contenedor docker. Para esto instalamos el paquete “docker.io”

```
$ apt-get install docker.io
```

Nos descargamos el contenedor posrgres.

```
$ docker pull postgres
```

Una vez descargado la imagen creamos un directorio en /opt donde alojaremos los datos de la Base de Datos. A continuacón creamos el contenedor.

```
$ docker run --name some-postgres -e
POSTGRES_PASSWORD={Passwd_User_Postgres} -v
pgdata:/opt/postgresql/data postgres
 -d postgres
```

Una vez echo esto tendremos nuestro contenedor funcionando y tendremos que crear nuestra BD y un usuario con permisos a la BD.

```
CREATE USER docker WITH PASSWORD ‘{PASSWORD}’ ;
CREATE DATABASE docker;
```

```
 GRANT ALL PRIVILEGES ON DATABASE docker TO docker;
```

Configuramos el fichero setings.py de la aplicación web con las credenciales para que se pueda conectar la aplicación web.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'django',
        'USER': 'django',
        'PASSWORD': 'django',
        'HOST': '172.17.0.2',
        'PORT': '',
    }
}
```

Una vez configurado la conexión a la base de datos nos diriguimos al directorio de la aplicación y ejecutamos .

```
$ python manage.py migrate
```

Esto creara las tablas que necesitamos para que nuestra aplicación funcione. Por ultmo creamos un usuario administrador con el cual poder acceder a la plicación.

```
$ python manage.py createsuperuser
```