# AUTOMATIZACIÓN EN LA CREACIÓN DE MAPAS DE AEMET Y SU PUBLICACIÓN ONLINE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MAYO DE 2018

AUTOR: KONRAD WOJCIECH JANICKI

TUTOR: JOAQUIN GASPAR MORA NAVARRO

MÁSTER EN GEOMÁTICA Y GEOINFORMACIÓN

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA CARTOGRÁFICA Y TOPOGRÁFICA

# 0.- ÍNDICE

## 0.1.- Imágenes

## 0.2- Código

# 1.- INTRODUCCIÓN

Este trabajo se ha realizado como parte de unas prácticas de becario en la empresa Vaersa[1]. Dicha empresa requería que se montase un servicio desde cero, el cual procesase datos enviados desde AEMET[2], en formato grib, a un formato de imagen y vectorial, .png y .shp, para finalmente publicarlo en un visor online.

La necesidad de este proyecto ha surgido por la necesidad de los servicios de emergencia y prevención de incendios de la Comunidad Valenciana para tener unos datos gráficos y alfanuméricos en tiempo real sin depender de otros organismos, como lo hacen hasta ahora.

## 1.1.- Objetivos

Los objetivos que se han impuesto por parte de la empresa contratista para un período de 2 meses fueron los siguientes:

- Descifrar los datos de formato .grib para tener algo legible ya que como se envían están en forma binaria.
- Crear imágenes interpoladas de dichos datos, en el caso de los vientos tanto con las flechas de la dirección del viento como no.
- Crear datos vectoriales, en forma de .shp, con dicha información para ser procesada por otro organismo dentro de la empresa.
- Crear un historial día a día de la información procesada.
- Crear un archivo .log para almacenar cualquier tipo de error que haya podido ocurrir durante la ejecución del programa.
- Crear un servicio para poder visualizar los datos on-line.
- Y la condición más crítica era hacerlo todo con software gratuito, ya que el objetivo principal de la empresa es ir pasando de software de pago, ESRI, a software gratuito.

[1] http://www.vaersa.com/
[2] http://www.aemet.es/es/portada

## 1.2.- Tipo de datos

Los datos enviados por AEMET se envían a un directorio del servidor cada 6 horas con archivos que contienen los modelos de 11.000 puntos de la Comunidad Valenciana para las siguientes 48 horas.

Si se abre el archivo mediante el software Degrib[3] se puede ver que la información del binario se estructura de la siguiente manera:

| | | | |
|---|---|---|---|
| 1.0 | UGRD | 10[m] HTGL (Specified height level above | 01/03/2018 06:00 |
| 2.0 | VGRD | 10[m] HTGL (Specified height level above | 01/03/2018 06:00 |
| 3.0 | TMP | 2[m] HTGL (Specified height level above gı | 01/03/2018 06:00 |
| 4.0 | RH | 2[m] HTGL (Specified height level above g | 01/03/2018 06:00 |

*Imagen 1- Estructura de los datos*

- Intensidad y dirección del Viento en notación de U y V. Las primeras dos "columnas" en el archivo binario lo ocupan los valores de la intensidad del viento en la dirección Este (U) y Norte (V) (Hooper, 2002). Con estos dos atributos se puede saber además el Azimut del viento, en meteorología es de Sur a Norte, y el valor total, km/h. Con el azimut y la intensidad se puede saber hacia dónde y con qué fuerza sopla el viento.

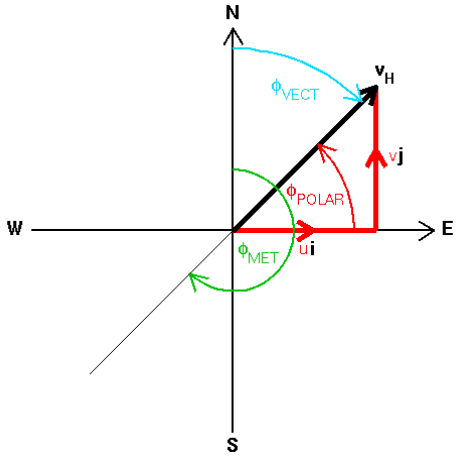



*Imagen 2- Representación de la U y V, (Hooper, 2002)*

- Temperatura en Grados Centígrados (ºC). El tercer campo en la "columna" de los binarios lo ocupa el valor de la temperatura en valor de grados centígrados.
- Humedades en % de humedad. El último valor de la "columna" lo ocupa el valor de las humedades medido en tanto por ciento,

[3] https://www.weather.gov/mdl/degrib_download

Cabe destacar el hecho de que en un principio iban a ser datos sólo referentes a la dirección del viento con una previsión de 72 horas pero el mes antes de acabar las prácticas la empresa ha cambiado el tipo de producto contratado al explicado en la parte superior.

Mientras que los nombres de los archivos se envían con el formato “fc + año + mes + día + época + hora de la previsión”


*Imagen 3- Nombre de los datos*

Por lo que en total se tiene 49 archivos binarios cada seis horas que hay que procesar.

# 2.- DESARROLLO DEL PROYECTO

Al ser un trabajo que no solo requiere muchos pasos para ser completado sino también mezcla diferentes lenguajes de programación se ha decidido hacer un pequeño resumen lo de que se hace para llegar al objetivo final.

- Un script que se ejecute cada seis horas para ejecutar el software
- Generar los datos que se van a visualizar:
  - Generar las imágenes a partir de los datos
  - Generar los datos vectoriales a partir de los datos
  - A la vez que se hacen los pasos anteriores generar un histórico
- Borrar los antiguos datos en Geoserver[4] y actualizarlo con los nuevos
- Creación de un visor básico on-line.

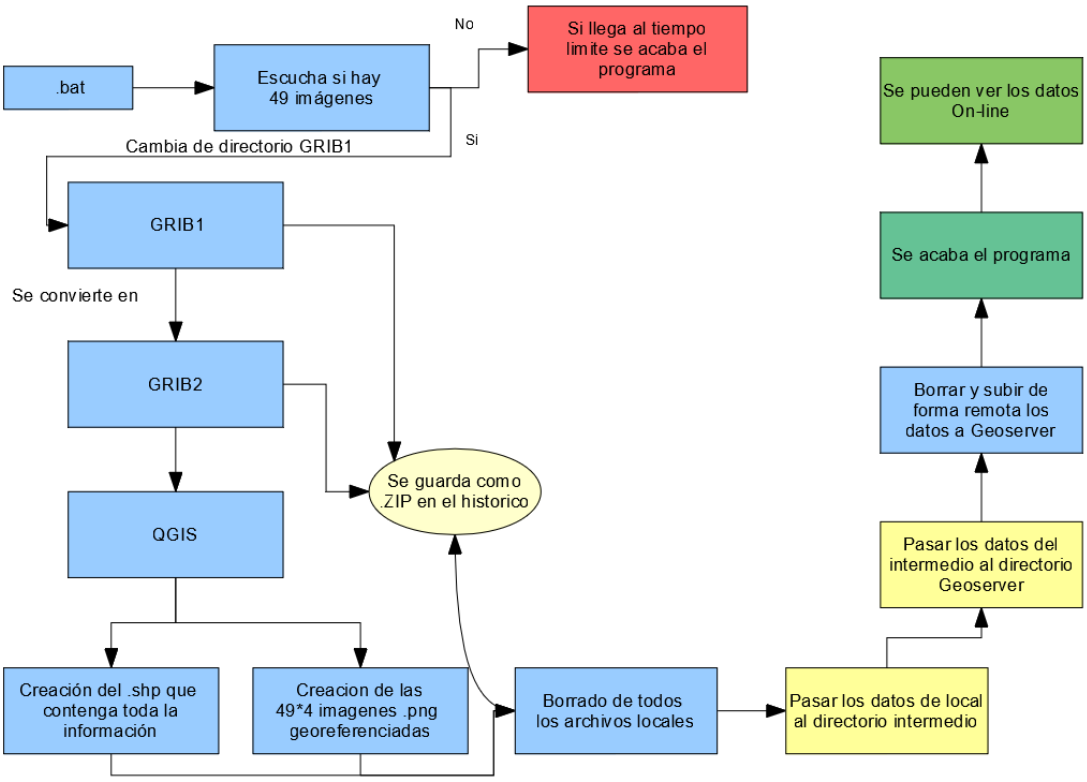



*Imagen 4- Esquema de los pasos que realiza el programa*

[4] http://geoserver.org/

## 2.1.- Software Empleado

Cómo a lo largo de la memoria se va a hacer mención a distintos softwares, para poder orientarse a lo largo de la memoria se van a explicar para que se ha empleado cada uno de ellos a lo largo proyecto.

### 2.1.1.- QGIS 2.18.0

Para poder manejar la creación de los mapas y los datos vectoriales se ha decidido emplear el software gratuito QGIS[5], el cual proporciona las herramientas necesarias para poder realizar el trabajo.
La versión de QGIS utilizada es la 2.18.0, "Las Palmas". La instalación de este software es la estándar proporcionada por la página web oficial.

#### *2.1.1.1.- Plugins*

Los plugins son funciones desarrolladas por los usuarios del programa para ayudar a otros usuarios en la realización de tareas. En este caso se ha instalado el siguiente plugin:

- Plugin Crayfish[6], versión 2.6.1, para poder manejar los datos GRIB2. La instalación se ha realizado mediante la herramienta proporcionada de QGIS de instalación de plugins.

#### *2.1.1.1.- Modificaciones del QGIS*

Para que los procesos ejecutados dentro del proyecto puedan funcionar primero se necesita configurar algunas opciones del QGIS.

[5] https://www.qgis.org/es/site/
[6] https://www.lutraconsulting.co.uk/products/crayfish/

### 2.1.1.1.1.- Macros y deshabilitar la pestaña del guardado

Como el proceso va a ser automatizado son necesarios unos macros, para ello hay que activarlos dentro de la configuración de QGIS ya que de base vienen desactivados.

Dentro de la pestaña de Configuración y Opciones accedemos a las propiedades generales del QGIS.

- El primer paso es el de dejar activados por defecto los macros siempre, esto se hace para que cuando se llame el programa para que se ejecute este a la vez llame sus propios scripts.

- El segundo paso es desactivar la ventana de pregunta de guardado. Hay que desactivarla porque cuando al acabar un proceso se quiera cerrar QGIS éste no se va a cerrar hasta que manualmente se le dé “si”.

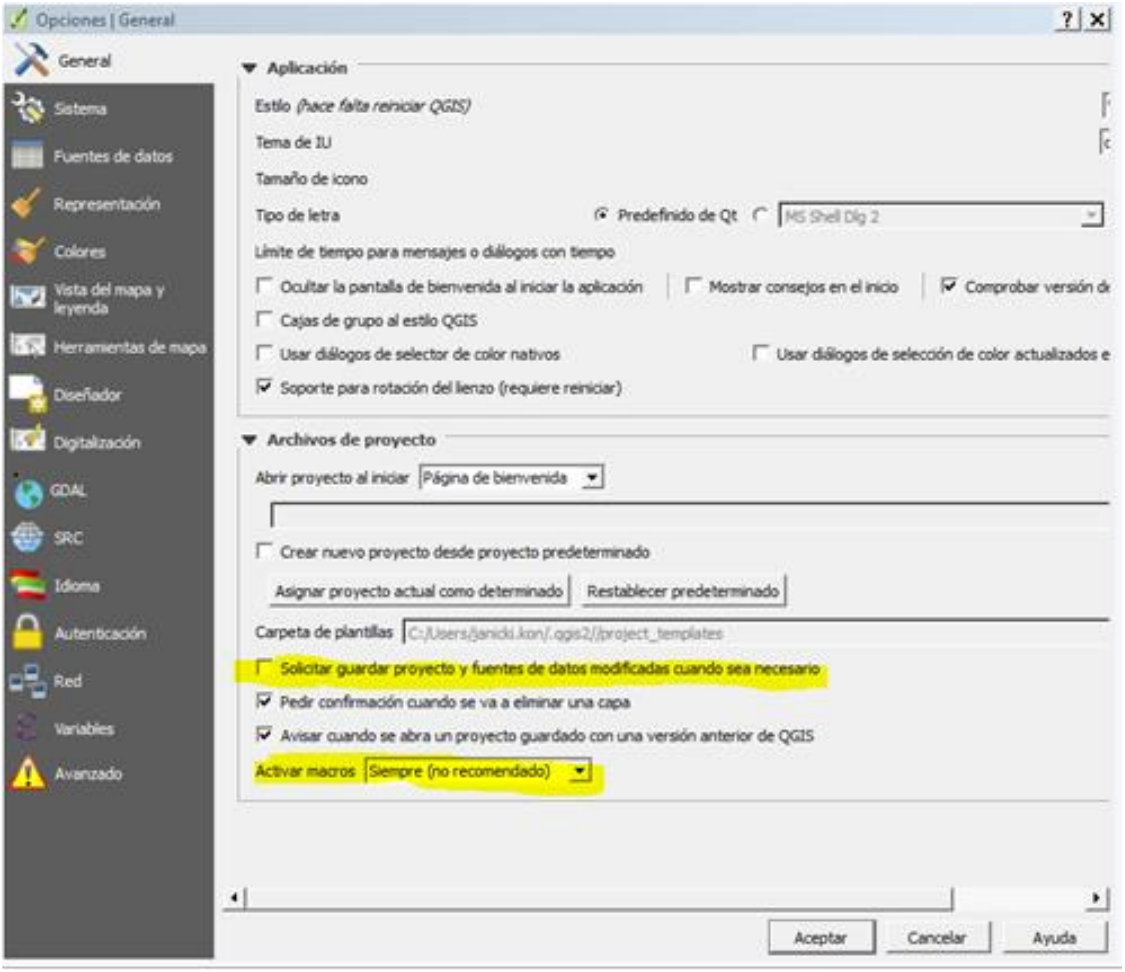



*Imagen 5- Configuración del QGIS*

### 3.1.1.1.2.- Habilitar la consola de Python al inicio de QGIS

La siguiente modificación que hay que realizar para el QGIS es crear un script de Python dentro de un directorio del propio QGIS, en este caso **C:\Users\janicki.kon\.qgis2\python**, con el nombre **startup.py**. Esto es necesario para que el QGIS a la hora de inicializarse encienda la consola de Python para que puedan funcionar los macros llamados.

```
import qgis
qgis.utils.iface.actionShowPythonDialog().trigger()
```

*Código 1- Consola QGIS*

### 2.1.2.- CDO

El CDO[7] es un software desarrollado por el Max-Planck-Institut fur Meterologie[8]. En este proyecto se emplea para poder pasar de los datos GRIB a GRIB2. Este paso es crítico porque ningún software empleado en esta práctica es capaz de leer el primero, por lo que hay que pasarlo de formato.

### 2.1.3.- Degrib

Para descifrar los datos de GRIB2 y pasarlos a un formato legible, .csv, se ha decidido descargar este programa proporcionado por La Administración Nacional Oceánica y Atmosférica (NOAA).

### 2.1.4.- Geoserver

El publicar los datos on-line se ha hecho con el software Geoserver versión 2.11.2.

#### *2.1.4.1.- Configurar el JSONP*

Para poder conectarse a los datos de las capas en formato JSON primero hay que activar esta opción dentro de la configuración de Geoserver. Para ello hay que entrar en el arhcivo **C:\Program Files (x86)\GeoServer 2.11.2\webapps\geoserver\WEB-INF\web.xml** y modificarlo. Concretamente hay que des comentar una parte del código para que se pueda acceder a los JSON.

[7] https://code.mpimet.mpg.de/projects/cdo/
[8] https://code.mpimet.mpg.de/

```
<!--Can be true or false (defaults to: false). -->
<!--When true the JSONP (text/javascript) output format is enabled -->

<context-param>
  <param-name>ENABLE_JSONP</param-name>
  <param-value>true</param-value>
</context-param>
```

*Código 2- Habilitar el JSON*

Seguidamente hay que reiniciar Geoserver, dentro de la carpeta bin en el directorio root se puede hacer eso mediante:

- Ejecutar shutdown.exe como administrador y esperar a que acabe.
- Ejecutar startup.exe como administrador y esperar a que acabe.

### 2.1.5.- PyScripter

Para manejar los scripts de Python se ha decidido emplear la IDE de PyScripter[9], por estar familiarizado con ella.

### 2.1.6.- Notepad++

Se ha empleado el Notepad++[10] para programar la página WEB en los lenguajes HTML, Javascript, CSS.

[9] https://sourceforge.net/projects/pyscripter/
[10] https://notepad-plus-plus.org/download/v7.5.6.html

## 2.2.- Lenguajes de programación

Además de emplear diferentes softwares también se han utilizado diferentes lenguajes de programación para poder llegar a los objetivos requeridos.

### 2.2.1.- Python

El principal lenguaje de programación empleado a lo largo de este proyecto fue el Python[11]. La estructura vertebral del proyecto se sujeta sobre este lenguaje, es el que crea los datos de salida y es quien se encarga además de llamar los otros lenguajes para poder ejecutar softwares, con comandos de CMD, o subir datos a Geoserver, cURL.

La versión instalada para este proyecto es la 2.7 y no hace falta descargar ninguna librería más aparte de las que se proporcionan en la instalación base.

### 2.2.2.- SHELL, consola CMD de Windows

Se ha empleado la consola CMD de Windows para llamar comandos de softwares externos, como el Degrib, ejecutar tareas programadas y ejecutar los proyectos de QGIS.

### 2.2.3.- CURL

El poder entrar en contacto con Geoserver y poder mandarle ordenes de borrado o subida de nuevas capas se ha hecho mediante CURL[12], que no es más que un lenguaje para enviar peticiones HTTP.

[11] https://www.python.org/
[12] https://curl.haxx.se/

### 2.2.4.- HTML, JavaScript, Css

El visualizador on-line se ha hecho mediante los lenguajes de HTML, Javascript y Css. Estos se explicarán más adelante en la memoria.

#### *3.2.4.1.- Openlayers*

La parte de visualización de los mapas se ha hecho mediante las librerías de OpenLayers[13] versión 4.2.

## 2.3.- Estructura de las Carpetas

La estructura del programa se define de una manera en la cual solo hay que copiar el directorio central para que su ejecución funcione sin problemas, eso se consigue haciendo que los scripts trabajen siempre de forma relativa en vez de absoluta.

De esta forma se evita el error de que en algún momento cuando se cambie de servidor los scripts dejen de funcionar.

### 2.3.1.- Directorio de creación de los datos

El directorio base tiene la siguiente estructura:

[13] https://openlayers.org/

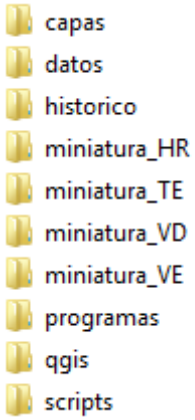



*Imagen 6- Directorio root de creación de datos*

Dentro de cada una de las carpetas se almacenan unos datos distintos:

- Capas: Almacena el .shp de la Comunidad Valenciana para los proyectos de QGIS.
- Datos: A esta carpeta se le envían los datos de los GRIB1, GRIB2, y donde se guardan las imágenes creadas por los QGIS antes de ser enviadas al servidor.

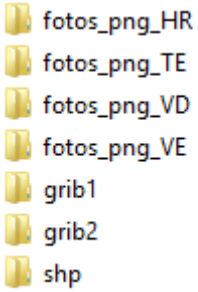



*Imagen 7- Directorio de los datos locales*

- Histórico: Almacena los .zip del histórico de los distintos datos.
- Miniatura_*: Almacena las diferentes miniaturas dependiendo del fenómeno que se quiera representar:
    - HR: Humedad Relativa.
    - TE: Temperatura.
    - VD: Viento con flechas.
    - VE: Viento sin flechas.

- Programas: Almacena los dos software externos que se necesitan para el procesado de los GRIB, el Degrib y CDO.
- QGIS: Guarda todos los proyectos y cosas relacionadas con QGIS, además de las plantillas para georreferenciar las imágenes.

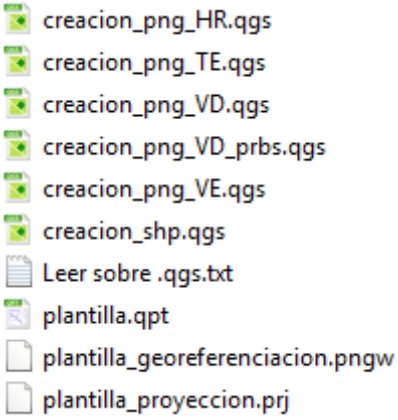



*Imagen 8- Directorio de los proyectos QGIS*

- Scripts: Contiene todos los scripts necesarios para la generación de los datos.

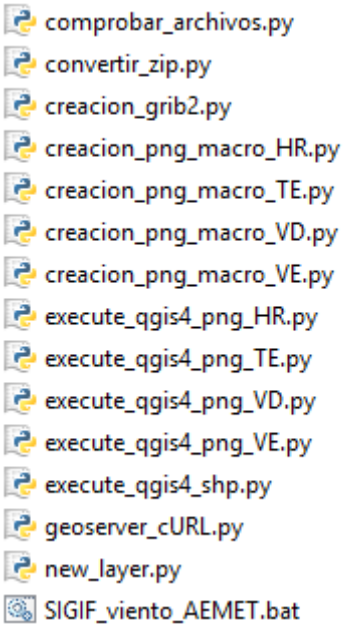



*Imagen 9- Directorio de los Scripts*

### 2.3.2.- Estructura de las carpetas Geoserver

El directorio root de Geoserver, si se ha instalado por defecto en un sistema de x64, se encuentra en **C:\Program Files (x86)\GeoServer 2.11.2**. En este caso el 2.11.2 es por la versión, por lo que dependerá de la versión que se haya instalado. Este directorio se ubica en otro servidor del cual se crean los datos por lo que luego es necesario enviárselos.

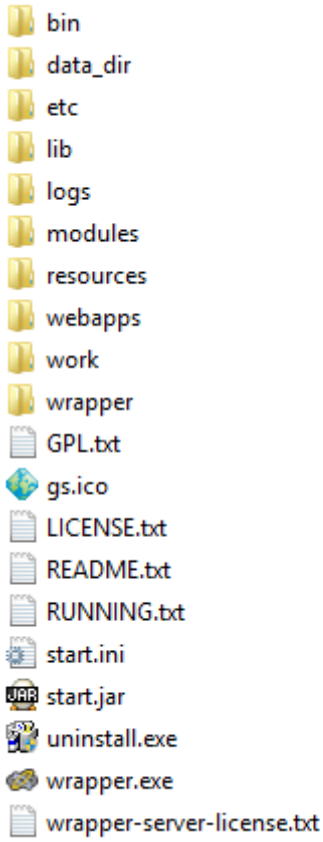



*Imagen 10- Directorio root Geoserver*

No se van a explicar todos los directorios, sino el más importante **data_dir**, que es dónde se guarda la configuración del servidor, la cual en este caso no se ha tocado, y los datos.

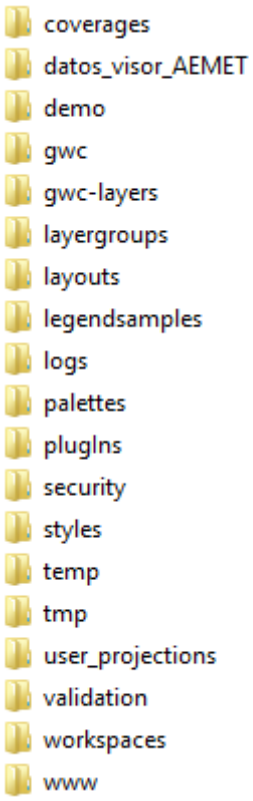



*Imagen 11- Directorio data_dir*

### *2.3.2.1.- Datos del visor*

Los datos se guardan en la carpeta **datos_visor_AEMET**, aquí es donde se envían los datos una vez el proceso de su creación se haya acabado en el paso 4.1.

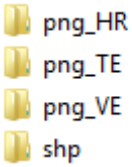



*Imagen 12- Directorio de los datos en Geoserver*

Cada una de las carpetas contiene los datos de uno de los diferentes fenómenos explicados en el apartado 1.2. En el caso de las imágenes son .png georreferenciados y en el del shapefile la capa comprimida en .zip.

#### *2.3.2.2.- WorkSpace*

Como se verá más adelante la carpeta “workspaces” contiene los espacios de trabajo de Geoserver, un conjunto de directorios de distintos trabajos. Estos espacios de trabajo contienen la información de dónde se encuentran los datos que debe procesar Goserver y de cómo debe hacerlo.
Toda esta información se encuentra en forma de .xml. A modo de resumen está configurado de la siguiente manera:

- Espacio de trabajo
    - Donde se encuentran los datos
        - Como tratar los datos
            - Sistema de coordenadas
            - Extensión
            - Dónde se ubican los estilos
            - ….

### 2.3.3.- Directorio del visor

La estructura del directorio del visor es muy sencilla, es una carpeta dentro del servidor II7 que contiene:

- Los 4 tipos de leyenda.
    - Vientos.
    - Humedades.
    - Dos de temperatura, según se quiera emplear una u otra.
- Un icono .gif de carga
- El archivo . HTML.
- El archivo JavaScript.


*Imagen 13- Estructura de carpeta del visor*

# 3.- CREACIÓN DE LOS DATOS

En esta parte se va a explicar el proceso de la creación de los datos, en resumen el objetivo de esta parte del código es la generación de la siguiente información:

- Imágenes de la intensidad de Viento con flechas y la Comunidad Valenciana.
- Imágenes de la intensidad de Viento sin flechas y la Comunidad Valenciana.
- Imágenes de la Temperatura y la Comunidad Valenciana.
- Imágenes de la Humedad y la Comunidad Valenciana.
- Shapefile que contenga información geométrica y alfanumérica.

## 3.1.- Ejecución del proceso y escucha de los datos

Para que el proceso se ejecute cada 6 horas se ha programado un proceso de Windows que ejecutará el programa **SIGIF_viento_AEMET.bat** que a su vez ejecutará **comprobar_archivos.py** a las siguientes horas:

- A las 04:00, se ejecutará para los datos de las 00:00
- A las 10:00 , se ejecutará para los datos de las 06:00
- A las 16:00 , se ejecutará para los datos de las 12:00
- A las 22:00 , se ejecutará para los datos de las 18:00

Este retraso de 4 horas entre la hora de ejecución del programa y la primera hora de previsión del modelo es debido a que los datos de AEMET se terminan de recibir unas 4 horas más tarde después de haber creado el modelo.

Una vez el archivo **comprobar_archivos.py** es ejecutado lo primero que hace es comprobar las hora a la cual se ha ejecutado el script para determinar en qué época se encuentra. Por época nos referimos a uno de los 4 posibles inicios de los datos (00, 06, 12, 18) y eliminar los datos de las carpetas locales pos si alguna ejecución anterior haya fallado:

```
now = datetime.datetime.now()
year=now.year
month= now.month
day= now.day
hour= now.hour
minute=now.minute
hour=hour+minute/60.0

if hour>=4 and hour<9:
    t_ep= "00"
if hour>=10 and hour<15:
    t_ep= "06"
if hour>=16 and hour<21:
    t_ep= "12"
if (hour>=22 and hour<24) or (hour>=0 and hour<3):
    t_ep= "18"
print t_ep

for fil in os.listdir(dir_destino_g1):
    os.remove(dir_destino_g1+"\\"+fil)
for fil in os.listdir(path_grib2):
    os.remove(path_grib2+"\\"+fil)
for fil in os.listdir(path_fotos_VD):
    os.remove(path_fotos_VD+"\\"+fil)
for fil in os.listdir(path_fotos_VE):
    os.remove(path_fotos_VE+"\\"+fil)
for fil in os.listdir(path_fotos_TE):
    os.remove(path_fotos_TE+"\\"+fil)
for fil in os.listdir(path_fotos_HR):
    os.remove(path_fotos_HR+"\\"+fil)
for fil in os.listdir(path_shp_ini):
    os.remove(path_shp_ini+"\\"+fil)
```

*Código 3- Determinación de la época de ejecución*

El siguiente paso es comprobar que en el directorio de los datos crudos se encuentran todos los archivos, como se sabe que el número de horas previstas es de 48, habrá 49 archivos. Hasta que no se haya ese número concreto el programa no va a seguir. Pero en el directorio de entrada hay otros datos, incluso de otros días si ha habido cortes eléctricos y el software no se ha podido ejecutar, por lo que se ha de comprobar que se encuentran archivos sólo del año, mes, día y época de ejecución.

```
cantidad=0
while cantidad<49:
    #Cada bucle que entra determina la fecha con las horas y minutos
    now = datetime.datetime.now()
    year=str(now.year)
    month= str(now.month)
    day= str(now.day)
    hour= now.hour
    minute=now.minute
    hour=hour+minute/60.0
    #Para poder igualar o nombrar archivos necesito que los meses y dias siemp
re tengan dos digitos
    if len(day)==1:
        day="0"+day
    if len(month)==1:
        month="0"+month
    onlyfiles=[]
    for fil in os.listdir(path_grib1):
        #Si el grib tiene FC y M en el nombre entonces lo cogerá
        if "fc" and 'm' in fil:
            resto=fil[2:-8]
            #Coger solo archivos de la epoca de ejecuccion, aunque haya muchos
 grib1 esto seleccionara solo los que entren dentro del tiempo de ejecucion
            if resto==(year+month+day+t_ep):
                onlyfiles.append(fil)
    cantidad=len(onlyfiles)
```

*Código 4- Comprobación del número de datos de entrada*

¿Pero qué pasa si nunca llegan los archivos necesarios?

El programa entra en un bucle infinito sin que ejecute nunca nada. No se puede dejar el programa de esta forma por lo que se han creado unos limitadores, estos comprueban la hora cada vez que se entra dentro del bucle y si pasa del límite establecido se parará el programa escribiendo en el archivo del .log un error describiendo que no se ha podido llegar a los 49 archivos, escribiendo cuántos archivos si se han encontrado.

```
    #Si el programa se excede del tiempo limite esto lo cortara y imprimira un
 texto en el archivo de status
    if t_ep=="00" and hour>9:
        f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha parado p
orque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"\n")
        f.close()
        sys.exit()
    if t_ep=="06" and hour>15:
        f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha parado p
orque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"\n")
        f.close()
        sys.exit()
    if t_ep=="12" and hour>21:
        f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha parado p
orque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"\n")
        f.close()
        sys.exit()
```

```
    #Aqui hay que ponerle un doble condicionante porque si solo dejo mayor a 3 si entra a las 23 el programa se cerrara
    #Es por ello que la hora para que se cierre debe estar comprendida entre las 3 y las 22.
    if t_ep=="18" and hour>3 and hour<22 :
        f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha parado porque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"\n")
        f.close()
        sys.exit()

f.write("Log del: "+year+month+day+t_ep+' : Los archivos fueron encontrados'+"\n")
f.flush()
```

*Código 5- Horas límite de ejecución del programa*

## 3.2.- Conversión de GRIB1 a GRIB2

Una vez el programa haya determinado que los 49 archivos existen, se moverán a un directorio local y se le informará al usuario en el archivo .log

```
f.write('                    : Moviendo los datos...'+"\n")
f.flush()
#Muevo los archivos del directorio general a mi directorio personalizado
try:
    for fil in onlyfiles:
        shutil.move(path_grib1+"\\"+fil, dir_destino_g1+"\\"+fil)
    f.write('                    : Los datos de GRIB1 fueron movidos'+"\n")
    f.flush()
except Exception as e:
    f.write('     ERROR          : Un error ha ocurrido moviendo los datos de GRIB1: '+str(e)+"\n")
    f.close()
    sys.exit()
resto=os.listdir(dir_destino_g1)[0][2:-8]
```

### 3.2.1.- Creación de GRIB2

Una vez se tenga los archivos en el directorio local se convierte los datos GRIB1 a GRIB2, este paso es necesario porque los datos GRIB1 no los puede leer el QGIS ni el software de la NOAA.

Se llama a otro script llamado **creación_grib2.py** el cual se encarga de crear los GRIB2 con el software mencionado anteriormente.

```
#creando los grib2
f.write('                    : Convirtiendo a GRIB2...'+"\n")
f.flush()
```

```
try:
    creacion_grib2.g1_2_g2(onlyfiles,dir_destino_g1,path_grib2,path_general)
    f.write('                    : Los GRIB2 se han creado satisfactoriamente'
+"\n")
    f.flush()
except Exception as e:
    f.write('     ERROR          : Ha ocurrdio un error en creacion_grib_2 scr
ipt: '+str(e)+"\n")
    f.close()
    sys.exit()
```

*Código 6- Llamado al convertidor de GRIB a GRIB2*

El script crea los GRIB2 con un nombre más fácil que el GRIB1, “00+hora predicción+.grb”, de esta forma se obtienen unos datos en local que van de 0000.grb a 0048.grb. Después de la ejecución de cada uno de las conversiones se le da un descanso de 2 segundos al programa, para que al ordenador le dé tiempo a crearlos sin crear errores dentro de los datos.

```
def g1_2_g2(onlyfiles,path_grib1,path_grib2,path_general):
##    epoca=onlyfiles[0][10:-2]
    #Recorre una por una todos los grib1
    for ii in range(0,49):
        #Rescato lo que sera el nombre del archivo
        nom_fichero=onlyfiles[ii]
        #Rescato los ultimos dos digitos del nombre del grib1, esto hace refer
encia al numero de la imagen
        n_img=nom_fichero[14:-4]
        #Defino el path hacia el grib1 y el path del fichero de salida, junto
con su nombre y extension .grb
        p_gb1_f=path_grib1+"\\"+nom_fichero
        p_gb2_f=path_grib2+"\\"+"00"+str(n_img)+".grb"
        #Creo lo que es el comando que ira dentro de la llamada a la consola
        cmd_g1_2_g2= path_general+r"\programas\cdo\cdo setgridtype,lonlat "+ p
_gb1_f+ " "+p_gb2_f
        #Llamo a la consola con el mensaje anterior para crear el grib2
        os.system(cmd_g1_2_g2)
        #Le doy dos segundos al programa para crear cada grid
        time.sleep(2)
    time.sleep(10)
```

*Código 7- Conversión de GRIB a GRIB2*

### 3.2.2.- Histórico de GRIB1 y GRIB2

El siguiente paso es el guardado de los GRIB1 y GRIB2 en la carpeta de histórico para que puedan almacenarse por si acaso. Para poder hacerlo se llama a otro script “convertir_zip.py” el cual se encarga de ubicar todos los ficheros de un directorio en el histórico en forma de un .zip

```
1. def convertir_zip(path_datos,out_path,resto,folder):
2.     zipf = zipfile.ZipFile('{0}.zip'.format(os.path.join(out_path, resto+r"_"+
   folder)), 'w', zipfile.ZIP_DEFLATED)
3.     for root, dirs, files in os.walk(os.path.join(path_datos, folder)):
4.         for filename in files:
5.             zipf.write(os.path.abspath(os.path.join(root, filename)), arcname=
   filename)
6.     zipf.close()
```

*Código 8- Creador de los .zip*

Hay que destacar que los GRIB2 no se pueden comprimir en un .zip como todos los demás datos porque se corrompe la cabecera.

Además como la creación del histórico no supone una parte crítica de la ejecución del programa, aunque esta falle el programa seguirá ejecutándose informando al usuario en el .log de un error a la hora de generar los .zip pero seguirá su ejecución.

```
1.  # creando los zip historicos de grib1 y grib2 dentro de la carpeta de historic
    o
2.  f.write('                    : Creando el .zip del GRIB1...'+"\n")
3.  f.flush()
4.  try:
5.      folder="grib1"
6.      convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
7.      f.write('                    : Los datos de GRIB1 se han guardado en el hi
    storico'+"\n")
8.      f.flush()
9.  except Exception as e:
10.     f.write('     ERROR          : Un error ha ocurrido creando el .zip de GRI
    B1: '+str(e)+"\n")
11.     sys.exit()
12.
13. f.write('                    : Creando el historico del GRIB2...'+"\n")
14. f.flush()
15. try:
16.     folder="grib2"
17.     shutil.copytree(path_datos+"\\"+folder,path_hist+"\\"+resto+"_grib2")
18.     f.write('                    : Los datos de GRIB2 se han guardado en el hi
    storico'+"\n")
19.     f.flush()
20. except Exception as e:
```

```
21.     f.write('     ERROR          : Un error ha ocurrido creando el historico d
    e GRIB2: '+str(e)+"\n")
```

*Código 9- Generación del histórico de GRIB y GRIB2*

## 3.5.- Creación de las imágenes

Una vez se tenga los 49 GRIB2 en su directorio se puede empezar a crear las imágenes.

### 3.5.1.- Configuración de los proyectos de QGIS

Antes de poder llamar a cualquier proyecto de QGIS desde una ventana de comandos el primer paso que hay que hacer es decirle que script se va a ejecutar a la hora de cargar el proyecto. Para ello se emplean los Macros, para el primer caso que es el de las imágenes de viento con flechas es:

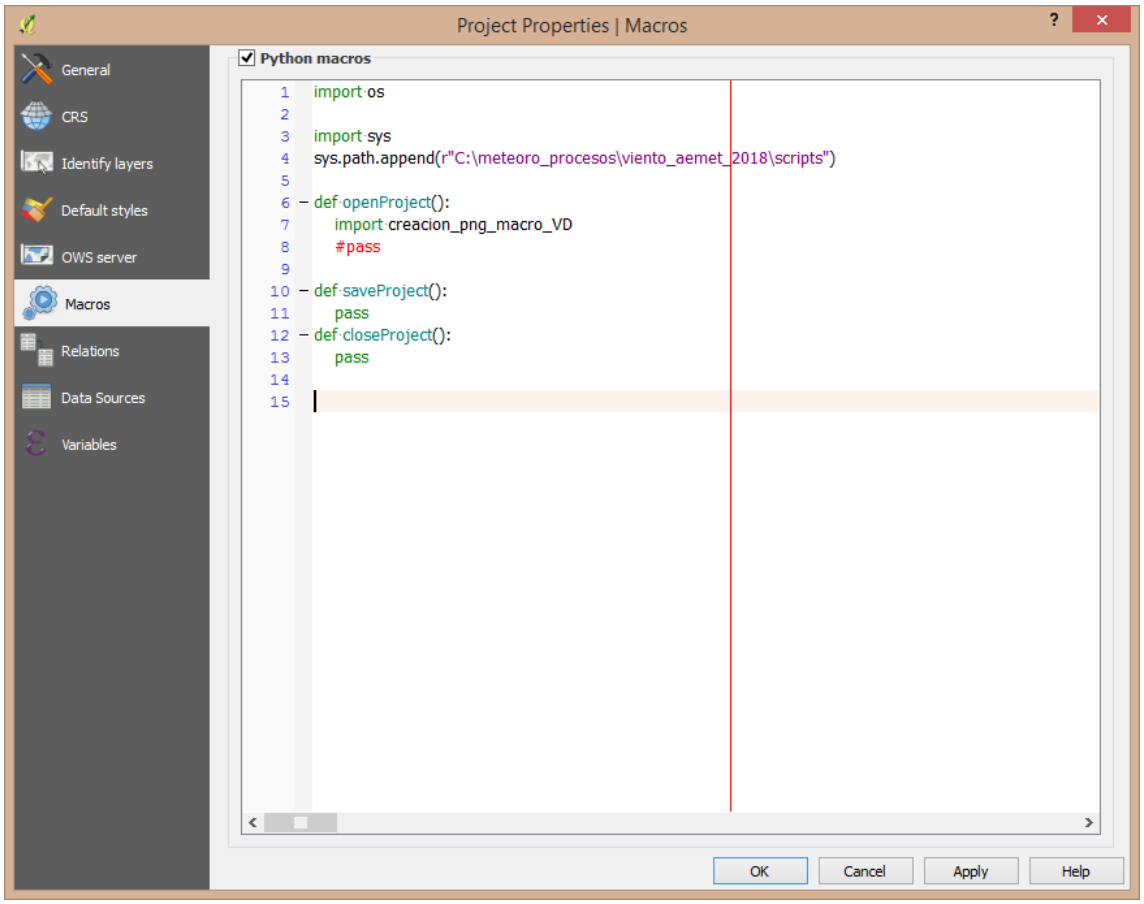



*Imagen 14- Configuración de los Macros en QGIS*

En el cual se puede apreciar que a la hora de abrir el proyecto lo primero que

realizará QGIS es ejecutar el script **creación_png_macro_VD** ubicado en el directorio definido en el apartado 2.3.1.

Una cosa a destacar de los proyectos de QGIS es que a la hora de crear las imágenes es que los proyectos tienen ya los directorios a los datos pre-cargados, es decir, el nombre y dirección de los archivos siempre será la misma, es por eso que a la hora de crear los GRIB2 se les ha asignado que siempre se creen con los mismos nombres, quedando el proyecto cargado sin ejecutar el macro:

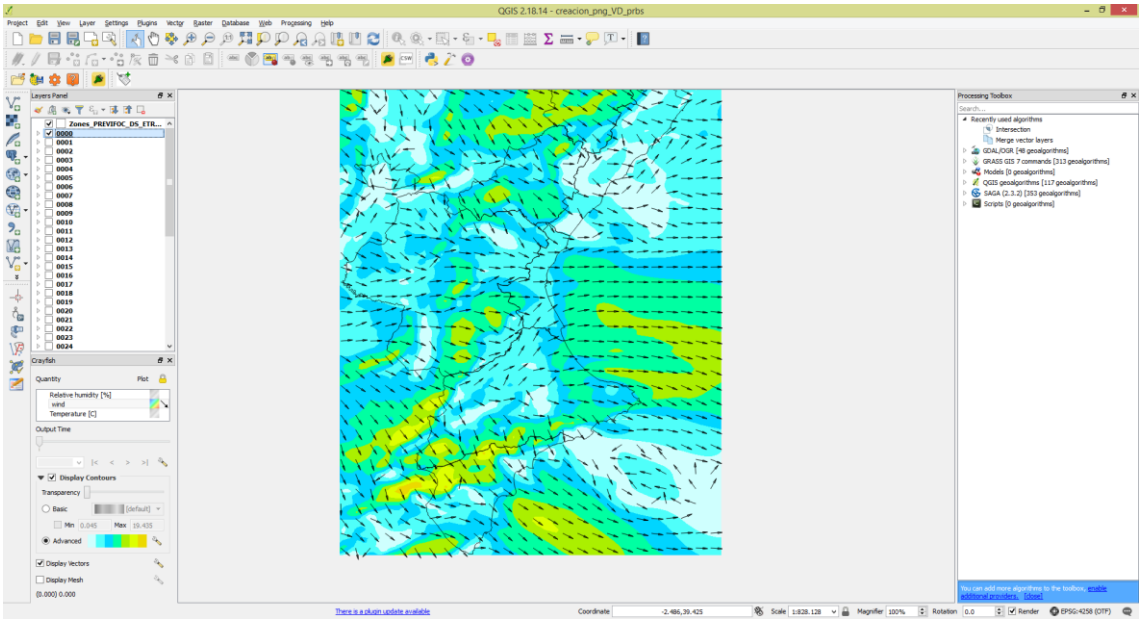


*Imagen 15- Muestra del proyecto de QGIS*

En la cual se puede ver a la izquierda todas las capas pre-cargadas. De esta forma solo hay que activar/desactivarlas para crear las imágenes.

Para poder crear dichas imágenes se ha creado una plantilla que será la que definirá el tamaño, extensión y dpi del archivo final. Dicha plantilla hay que guardarla para luego poder llamarla, para estos casos se la ha definido como **plantilla.qpt**.

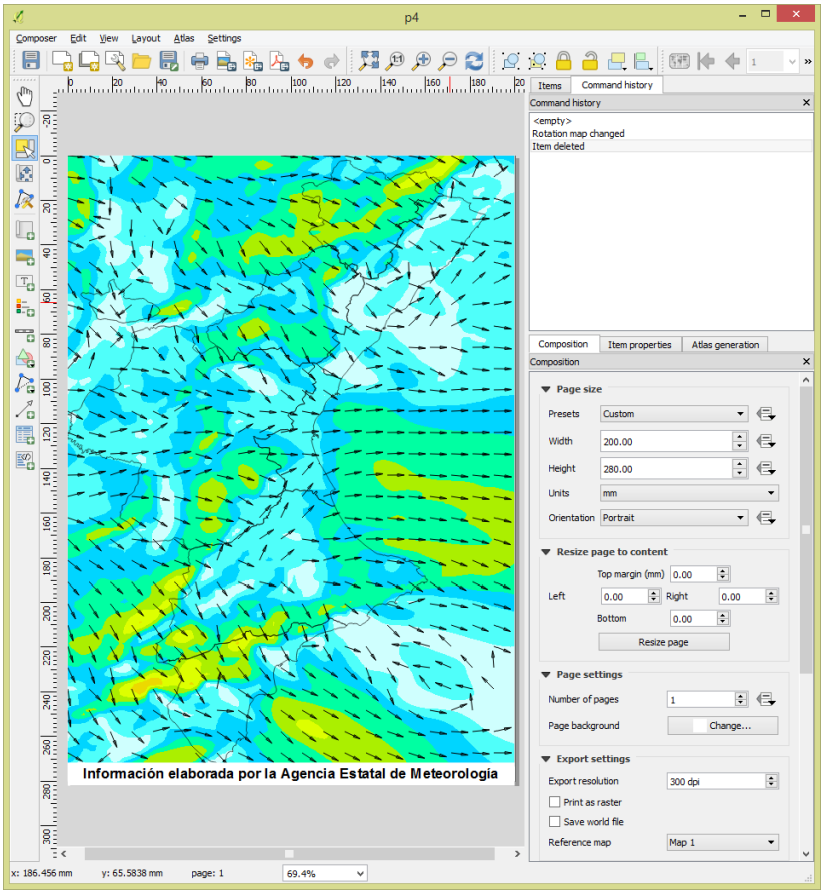



*Imagen 16- Muestra de la plantilla*

## 3.5.2.- Imágenes de Viento con flechas

### *3.5.2.1.- Llamado al proyecto de QGIS*

Las primeras imágenes que se crean son las de los vientos con las Flechas y la Comunidad Valenciana de fondo. Para ello se ejecuta un script **execute_qgis4_png_VD.py** el cual llama un proyecto de QGIS.

```
1.  f.write('                    : Ejecutando el QGIS para la creacion de las imag
    enes de viento con flechas...'+"\n")
2.  f.flush()
3.  try:
4.      execute_qgis4_png_VD.execute_qgis(path_general)
```

*Código 10- Llamado al script de ejecución del proyecto QGIS*

```
def execute_qgis(path_general):
    #el –nologo– hace que se encienda mucho mas rápido el proyecto ya que no carga muchas cosas
    execute_qgis=r"call C:\PROGRA~1\QGIS2~1.18\bin\qgis --nologo --project "+path_general+r"\qgis\creacion_png_VD.qgs"
    os.system(execute_qgis)
    #Suspendo el programa para que le de tiempo a QGIS a crear todas las imagenes
    time.sleep(180)
```

*Código 11- Ejecución del proyecto QGIS*

El proceso tarda unos 120 segundos en crear todas las imágenes por lo que se le ha puesto un tiempo de espera de 180 segundos, por si acaso un 50% de sobra.

#### *3.5.2.2.- Macro de QGIS*

Una vez se haya ejecutado el proyecto de QGIS éste ejecutará un Macro que tenga definido, como se ha explicado en el apartado anterior. Lo que hace dicho macro es activar/desactivar las capas de los vientos y llamar una plantilla, la cual también se ha explicado en el apartado anterior, sobre la cual se imprimirá lo que haya activo en el canvas en el momento de llamar la función de impresión.

Hay un pequeño bug con la función de impresión del QGIS y va a imprimir una imagen hacia atrás, es decir, que la imagen 1 estará en blanco y la imagen 2 representará lo que tendría que haber en la 1ª, ese error se ha conseguido corregir añadiendo una imagen de más al final, 50 en vez de 49, duplicando la última.

```
def creacion_png():
    iface.mapCanvas().refresh()
    #Obtengo las capas
    layers =  iface.legendInterface().layers()
    legend = iface.legendInterface()
    #Pondo una imagen de mas para que el codigo peuda llegar al final, eso es porque el bucle va con una iteracion de retraso

    for i in range(0,51):
        #Cargo una por una las capas a la vez que la anterior a la cargada
```

```
        layer=layers[i]
        layer_ante=layers[i-1]
        if layer.name()!="Zones_PREVIFOC_DS_ETRS89":
            #Reproyecto las capas a ETRS89 porque si no lo hago salen desplaza
dos en el mapa
            print
            crs = layer.crs()
            crs.createFromId(4258)
            layer.setCrs(crs)
            #Empieza a desactivar todos los layers anteriores a excepcion de l
a primera iteraci?n
            if i>=2:
                legend.setLayerVisible(layer_ante, False)
                iface.mapCanvas().refresh()
                layer.triggerRepaint()
            #Activa el layer actual
            legend.setLayerVisible(layer, True)
            #Refresco el canvas
            iface.mapCanvas().refresh()
            layer.triggerRepaint()
            #Le doy un segundo para que el programa reaccione
            sleep(1)
            #Creo el composer, lo que hace que pueda imprimir el canvas
            canvas = iface.mapCanvas()
            myFile =path_general+r'\qgis\plantilla.qpt'
            myTemplateFile = file(myFile, 'rt')
            myTemplateContent = myTemplateFile.read()
            myTemplateFile.close()
            composerAsDocument = QDomDocument()
            composerAsDocument.setContent(myTemplateContent)
            composition = QgsComposition(canvas.mapSettings())
            composition.loadFromTemplate(composerAsDocument, {})
            image = composition.printPageAsRaster(0)
            #Empieza a guardar a partir de la segunda iteracion porque la prim
era imagen esta vacia, es por eso que hay 50 capas cargadas.
            if i>=2:
                #Guardo la imagen
                if len(str(i-2))==1:
                    asw="0"+str(i-2)
                else:
                    asw=str(i-2)
                image.save(path_general+r'\datos\fotos_png_VD\output'+asw+'.pn
g', "png")
    legend.setLayerVisible(layer, False)
    iface.mapCanvas().refresh()
    layer.triggerRepaint()
```

*Código 12- Creación de las imágenes con Macro de QGIS*

Una vez se hayan creado todas las imágenes hay que cerrar el programa, para ello primero se cierra los procesos internos de QGIS y luego QGIS.exe como tal.

```
iface.actionExit().trigger()
os.kill(os.getpid(), 9)
```

*Código 13- Cierre de QGIS*

De esta forma se acaba con 49 imágenes en un directorio local con nombre "output+número de la imagen+.png" con un resultado como la imagen siguiente de ejemplo:

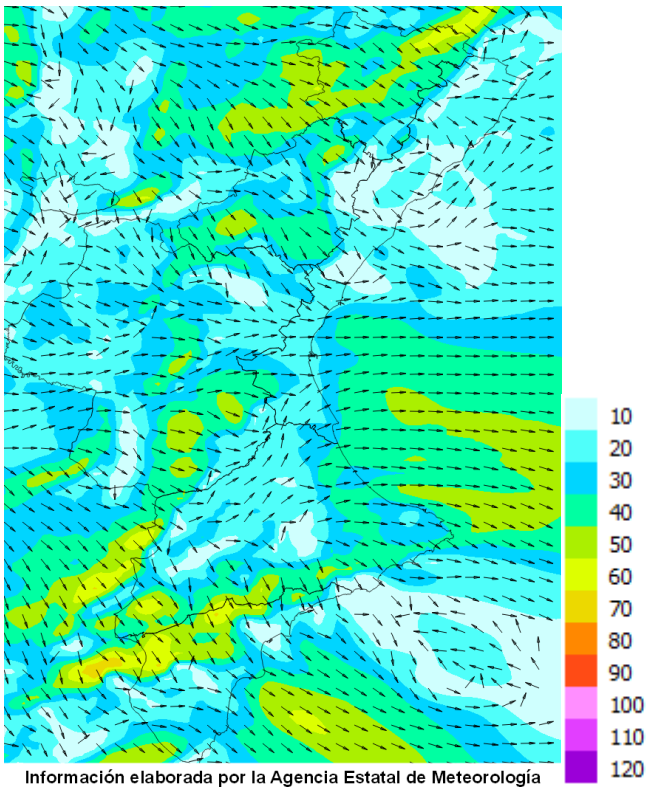



*Imagen 17- Resultado final de los vientos con flechas*

El estilo para el mapa del viento se ha escogido siguiendo el modelo de AEMET (AEMET, 2018).

### *3.5.2.3.- Renombrado de las fotos y su georreferenciación*

Ahora que se tiene todas las imágenes en el directorio local hay que cambiarles el nombre, para que coincida con "Tipo de Imagen_+ Año+ Mes+ Dia+ Epoca.png", como ejemplo el nombre de la imagen anterior es VD_2018_01_03_06.png.

A la vez que se renombran las imágenes se copian y renombran los archivos de georreferenciación de un directorio local, hay que georreferenciarlos porque por ahora son solo imágenes. Se pueden copiar y pegar dichos archivos sin ningún problema porque la zona va a ser siempre la misma.

Las imágenes se georreferencian respecto el sistema de coordenadas ETRS89 UTM Zona 30, aunque los datos vengan en ETRS89. Esto se hace por si en un futuro se quiere emplear las imágenes en otros visores de la empresa, los cuales emplean dicho sistema.

La georreferenciación de las imágenes se compone de dos archivos:

- Archivo .pngw que es el archivo world file y contiene la siguiente información:
    - Tamaño de píxeles en X
    - Rotación eje Y
    - Rotación eje X
    - Tamaño de pixeles en dirección Y
    - Coordenada X del centro del píxel superior izquierdo
    - Coordenada Y del centro del píxel superior izquierdo
- Archivo .prj que contiene el sistema de proyección.

```
    d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:8]), int(t_ep))
    f.write('                    : Renombrando y guarando las fotos...'+"\n")

    f.flush()
        #Se recorren uno por uno los archivos y con la librería de date se va añadiendo uno por uno la hora para nunca equivocarse de hora
    for fil in os.listdir(path_fotos_VD):
        if len(str(d.day))==1:
            day='0'+str(d.day)
        else:
            day=str(d.day)
        if len(str(d.month))==1:
            month='0'+str(d.month)
        else:
            month=str(d.month)

        if len(str(d.hour))==1:
            hour='0'+str(d.hour)
        else:
            hour=str(d.hour)
        os.rename(path_fotos_VD+"\\"+fil,path_fotos_VD+"\\"+"VD_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".png")
        #De paso copio y pego el archivo de georreferenciacion y le cambio el nombre para que coincida con el del png, lo mismo con el prj
        shutil.copyfile(path_qgis_capas+r'\plantilla_georeferenciacion.pngw', path_fotos_VD+"\\"+"VD_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".pgw")
```

```
22.         shutil.copyfile(path_qgis_capas+r'\plantilla_proyeccion.prj', path_fot
    os_VD+"\\"+"VD_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".prj")
23.         d+=datetime.timedelta(hours=1)
```

*Código 14- Renombrado y georreferenciación*

### *3.5.2.4.- Guardado de las fotos al histórico*

El siguiente paso es guardar las fotos en formato .zip al directorio histórico.

```
1.      folder="fotos_png_VD"
2.      convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
3.      f.write('                    : Las fotos fueron renombrados y guardados'+"
    \n")
4.      f.flush()
```

*Código 15- Guardado histórico de las imágenes del Viento con flechas*

### *3.5.2.5.- Mover las fotos al directorio intermedio y eliminar las antiguas*

El siguiente paso es el mover las imágenes al directorio intermedio, directorio entre el servidor de la creación de datos y el servidor donde está Geoserver, a la vez que eliminar todos los archivos anteriores a la Época de ejecución. Primero se mueven todas las fotos para luego borrar todas las anteriores a la fecha definida.

```
1.      #Muevo las imagenes al directorio de la web
2.      f.write('                    : Moviendo las fotos...'+"\n")
3.      f.flush()
4.      for fil in os.listdir(path_fotos_VD):
5.          if "output" not in fil:
6.           shutil.move(path_fotos_VD+"\\"+fil,path_fotos_web_VD+"\\"+fil)
7.      f.write('                    : Las fotos fueron movidos al directorio '+pa
    th_fotos_web_VD+"\n")
8.      f.flush()
9.      f.write('                    : Borrando las fotos antiguas...'+"\n")
10.     f.flush()
11.     d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:8]), in
    t(t_ep))
12.     for fil in os.listdir(path_fotos_web_VD):
13.         #Borrados de todas las imágenes antiguas
14.         if (".png" in fil) or (".pgw" in fil) or (".prj" in fil):
15.             if "output" not in fil:
16.                 d_img=datetime.datetime(int(fil.split("_")[1]), int(fil.split(
    "_")[2]), int(fil.split("_")[3]),int(fil.split("_")[4][:2]))
17.                 if d_img<d:
18.                     os.remove(path_fotos_web_VD+"\\"+fil)
19.             else:
20.                 f.write('                    : No le ha dado tiempo al'+resto+
    ".....................................................................\n
    ")
21.                 f.flush()
22.     f.write('                    : Ha acabado de hacer las fotos de los Viento
    s con flechas'+"\n")
23.     f.flush()
```

*Código 16- Envío y borrado de imágenes al servidor intermedio*

#### *3.5.2.6.- Creación de miniaturas*

El siguiente paso crea las miniaturas de las imágenes, estos datos por ahora no se van a implementar en ningún sitio por lo que no se mueven del directorio local, pero como la empresa ha pedido su creación se ha definido una función que lo haga.

```
1.  f.write('                    : Creando las miniaturas del viento con flechas'+
    "\n")
2.      f.flush()
3.      size = 500, 300
4.      for fil in os.listdir(path_miniatura_VD):
5.          os.remove(path_miniatura_VD+'\\'+fil)
6.      for fil in os.listdir(path_fotos_web_VD):
7.          if '.png' in fil:
8.              im = Image.open(path_fotos_web_VD+'\\'+fil)
9.              im.thumbnail(size, Image.ANTIALIAS)
10.             im.save(path_miniatura_VD+'\\'+fil, "PNG",dpi=(96,96))
11.     f.write('                     : Ha acabado de crear las miniaturas'+"\n")
12.     f.flush()
```

*Código 17- Creación de miniaturas*

### 3.5.3.- Imágenes de Viento sin flechas

El proceso de la creación de las imágenes sin flechas es igual al de creación de las de con flechas. Cambiando los directorios locales, el archivo del proyecto y que dentro del proyecto de QGIS se hayan desactivado las flechas. Se puede ver las diferencias en el Anexo 9.1.5 en la línea 193 y 9.1.9 línea 13.

Esto hace que haya dos scripts nuevos, tres directorios nuevos y un nuevo proyecto:

- El script de llamado al proyecto
- El proyecto
- El script del Macro
- Directorio local de las fotos
- Directorio intermedio de las fotos

- Directorio local de las miniaturas

El resultado queda de la siguiente forma:

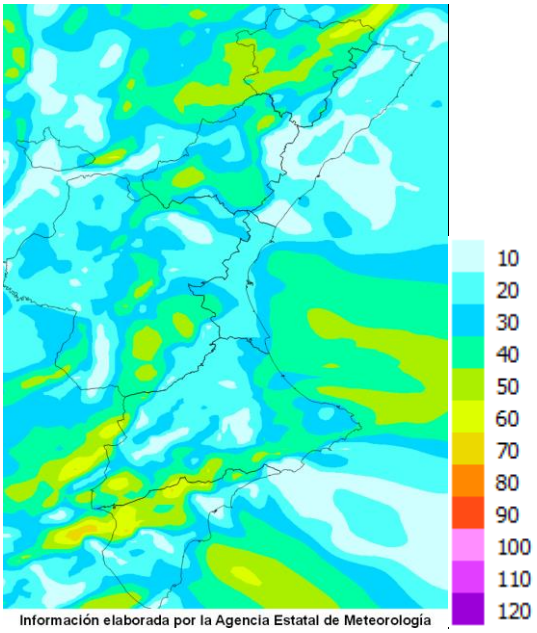



*Imagen 18- Resultado final Viento sin flechas*

La escala de colores es la misma que la empleada por la AEMET (AEMET, 2018).

### 3.5.4.- Imágenes de Humedades

Como el caso anterior, éste caso sigue la misma estructura que los anteriores, pero esta vez el proyecto de QGIS está por separado para poder cubrir unos datos diferentes, también como en el caso anterior hay seis elementos nuevos.

Se puede ver las diferencias en el Anexo 9.1.6 línea 193 y 9.1.10 línea 13.

El resultado queda de la siguiente forma:

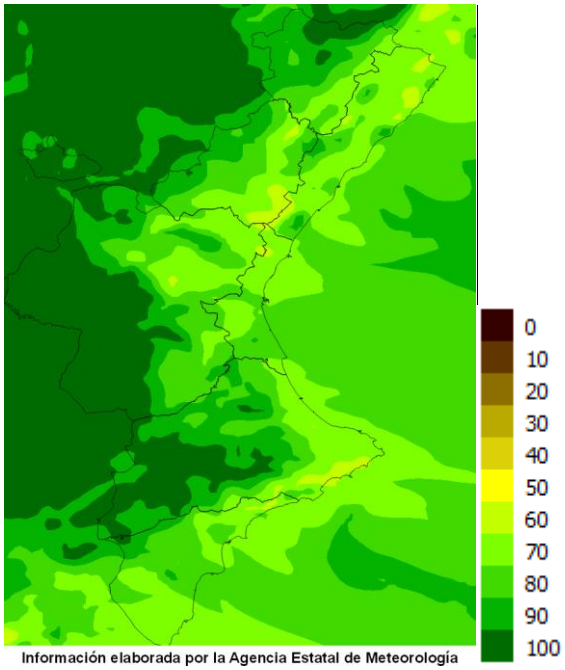



*Imagen 19- Resultado final Humedades*

La escala de colores es la misma que la empleada por Intellicast (Intellicast, 2018).

### 3.5.5.- Imágenes de Temperaturas

Finalmente el caso de las Temperaturas sigue otra vez la misma estructura.

Se puede ver las diferencias en el Anexo 9.1.7 línea 193 y 9.1.11 línea 13.

El resultado queda de la siguiente forma:

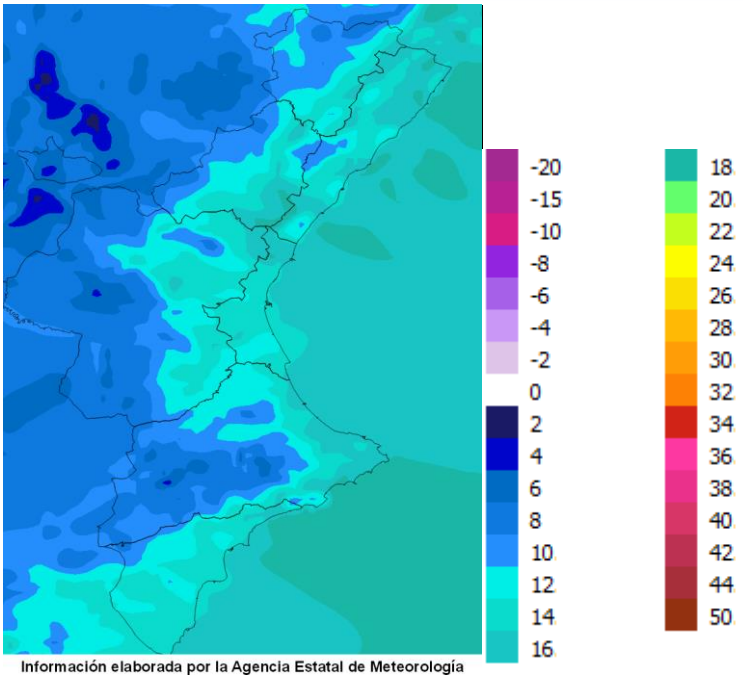



*Imagen 20- Resultado final Temperaturas*

La escala de colores empleada es la misma que la empleada por AEMET (AEMET, 2018).

### 3.5.6.- Horizonte

Además de las fotos georreferenciadas en cada directorio se ubicará un archivo llamado **Horizonte.txt** el cual contiene la primera fecha prevista, es decir, la fecha de la primera imagen en UTC.

```
try:
    f.write('                    : Creando los archivos de horizonte'+"\n")
    f.flush()
    hori_txt = open(path_fotos_web_VD+r'\horizonte.txt','w')
    hori_txt.write(resto[:-2]+r"_"+resto[-2:])
    hori_txt.close()
    shutil.copyfile(path_fotos_web_VD+r'\horizonte.txt', path_fotos_web_VE+r'\horizonte.txt')
    shutil.copyfile(path_fotos_web_VD+r'\horizonte.txt', path_fotos_web_HR+r'\horizonte.txt')
    shutil.copyfile(path_fotos_web_VD+r'\horizonte.txt', path_fotos_web_TE+r'\horizonte.txt')
except Exception as e:
    f.write('     ERROR          : Un error ha ocurrido creando los horizontes : '+str(e)+"\n")
```

*Código 18- Creación del archivo de horizonte*

## 3.6- Creación del Shapefile

El siguiente paso es el crear el archivo Shapefile que contenga la información de los 49 archivos. Este proceso es el más costoso en tiempo ya que tarda casi una hora en terminar, eso es debido a que hay que crear un .shp con un total de más de 3.000.000 atributos dentro de la tabla (11.000 puntos * 49 días * 6 campos).

### 3.6.1- Llamado al proyecto QGIS

El primer paso es igual que el caso de las imágenes, llamar un proyecto de QGIS para que este se ejecute en segundo plano y pueda empezar el proceso del macro, **new_layer.py**.

```
1.  f.write('                    : Ejecutando el QGIS para la creacion de los .SHP
    ...'+"\n")
2.  f.flush()
3.  try:
4.      execute_qgis4_shp.execute_qgis_shp(path_general)
```

*Código 19- Llamado al script de ejecución del pryecto QGIS*

Se le ha puesto una espera de una hora y media para que al programa le dé tiempo cunado el servidor esté saturado.

```
1.  def execute_qgis_shp(path_general):
2.      execute_qgis=r"call C:\PROGRA~1\QGIS2~1.18\bin\qgis --nologo --
    project "+path_general+r"\qgis\creacion_shp.qgs"
3.      os.system(execute_qgis)
4.      #Suspendo el programa para que le de tiempo a QGIS a crear el shapefile
5.      time.sleep(5400)
```

*Código 20- Ejecución del proyecto QGIS*

### 3.6.2- Crear un layer vacío y definir el Horizonte inicial

El primer paso que realiza el Macro de la creación del Shapefile es la creación de un capa vacío, QGIS no dispone de un comando directo para crear un layer en un directorio. Lo que se ha hecho para sobrepasar este inconveniente es crear un layer en la memoria para luego guardarlo y llamarlo, de esta forma se ha conseguido crear un layer vacío.

Además de crear una capa vacía se ha definido al igual que en el caso de la creación de las imágenes un horizonte inicial, aunque en este caso no será un archivo de texto por separado sino un campo dentro de la tabla de atributos.

```
1.      #Creo un nuevo layer sobre el cual se guardara la informacion de las 49 ca
    pas, este tiene una proyection ETRS89 al igual que las demas capas transformad
    as
2.      vl = QgsVectorLayer("Point?crs=epsg:25830", "temporary_points", "memory")

3.      writer = QgsVectorFileWriter.writeAsVectorFormat(vl,path_general+r"\datos\
    shp\NewFile.shp","utf-8",None,"ESRI Shapefile")
4.      layer_total = QgsVectorLayer(path_general+r"\datos\shp\NewFile.shp", "SHP_
    V", "ogr")
5.
6.      name=os.listdir(path_general+r"\datos\grib1")[0].split('+')[0][2:]
7.      horiz=name[:4]+r"_"+name[4:-4]+r"_"+name[6:-2]+r"_"+name[8:]
```

*Código 21- Creación del layer vacío y definición del horizonte*

### 3.6.3- Conversión de GRIB2 a CSV

El siguiente paso que realiza el programa es convertir uno por uno los archivos de GRIB2 en diferentes .csv, conteniendo cada uno de ellos un parámetro diferente de los que se quiera obtener de los datos.

Para ello se ha empleado el llamado software Degrib, el cual permite transformar los datos de GRIB2 a .csv o .shp. Se ha decidido escoger el formato de .csv y juntar los archivos con un script personalizado porque la herramienta de conversión a .shp no funcionaba correctamente y muchas veces cortaba partes del archivo, seguramente debido a la gran cantidad de puntos porque se han hecho pruebas con archivos más pequeños y no había fallos en la herramienta.

Hay que destacar que se obtienen 4 distintos .csv, aunque dos de ellos expresan solo una variable, que es la intensidad del viento:

- Componente U del viento.
- Componente V del viento.
- La Temperatura.
- La Humedad.

```
    cnt=0
    for fil in os.listdir(path_general+r"\datos\grib2"):
        #Creando los csv que contienen el componente U y V, TE y HR
        print fil
        create_u=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
al+r"\datos\grib2"+"\\"+fil+r" -C -msg 1 -Csv -
out "+path_general+r"\datos\shp\csv_u.csv"
        create_v=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
al+r"\datos\grib2"+"\\"+fil+r" -C -msg 2 -Csv -
out "+path_general+r"\datos\shp\csv_v.csv"
        create_t=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
al+r"\datos\grib2"+"\\"+fil+r" -C -msg 3 -Csv -
out "+path_general+r"\datos\shp\csv_tF.csv"
        create_h=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
al+r"\datos\grib2"+"\\"+fil+r" -C -msg 4 -Csv -
out "+path_general+r"\datos\shp\csv_h.csv"
        os.system(create_u)
        os.system(create_v)
        os.system(create_t)
        os.system(create_h)
```

*Código 22- Creación de los .csv a partir de los GRIB2*

Seguidamente se carga dichos archivos:

```
        dat_u=numpy.genfromtxt(path_general+r"\datos\shp\csv_u.csv",delimiter=
',')
        dat_u = numpy.delete(dat_u,0, axis=0)
        dat_v=numpy.genfromtxt(path_general+r"\datos\shp\csv_v.csv",delimiter=
',')
        dat_v = numpy.delete(dat_v,0, axis=0)
        dat_tF=numpy.genfromtxt(path_general+r"\datos\shp\csv_tF.csv",delimite
r=',')
        dat_tF = numpy.delete(dat_tF,0, axis=0)
        dat_h=numpy.genfromtxt(path_general+r"\datos\shp\csv_h.csv",delimiter=
',')
        dat_h = numpy.delete(dat_h,0, axis=0)
```

*Código 23- Carga de los .csv dentro de QGIS*

### 3.6.4- Rellenado de los campos

Ahora que hay una forma de poder leer los datos de los archivos, se van a guardar éstos dentro del shapefile vacío. En la primera iteración además de los propios valores de los datos se guardarán en las primeras seis columnas:

- El ID del punto
- El Horizonte
- Coordenada X en ETRS89 hus0 30
- Coordenada Y en ETRS89 huso 30
- Latitud en ETRS89
- Longitud en ETRS89

```
1.          #En la primera iteracion se llena lel nuevo .shp con los datos de los
    ID y las coordenadas de cada uno de los puntos
2.          if cnt==0:
3.              layer_total.dataProvider().addAttributes( [ QgsField("ID", QVarian
    t.Int)] )
4.              layer_total.dataProvider().addAttributes( [ QgsField("Horizonte",
    QVariant.String,"string",13)] )
5.              layer_total.dataProvider().addAttributes( [ QgsField("X_ETRS89", Q
    Variant.Double, "double", 12, 3)] )
6.              layer_total.dataProvider().addAttributes( [ QgsField("Y_ETRS89", Q
    Variant.Double, "double", 12, 3)] )
7.              layer_total.dataProvider().addAttributes( [ QgsField("ETRS89_lat",
     QVariant.Double, "double", 6, 3)] )
8.              layer_total.dataProvider().addAttributes( [ QgsField("ETRS89_lon",
     QVariant.Double, "double", 6, 3)] )
9.              for rowu in dat_u:
10.                     rx=str(int(rowu[0]))
11.                     if len(rx)==1:
12.                        rx="0"+rx
13.                     ry=str(int(rowu[1]))
14.                     if len(rx)==1:
15.                        ry="0"+ry
16.                     ID_pnt=rx+'00'+ry
17.                     x=QgsPoint(float(rowu[3]),float(rowu[2]))
18.                     pt = xform.transform(x)
19.                     feat = QgsFeature()
20.                     feat.setGeometry(QgsGeometry.fromPoint(QgsPoint(pt)))
21.                     feat.setAttributes([int(ID_pnt), horiz, pt[0], pt[1],float
    (rowu[2]),float(rowu[3])])
22.                     layer_total.dataProvider().addFeatures([feat])
```

*Código 24- Definición de las primeras columnas*

Antes de poder añadir los datos a las columnas de la tabla primero hay que crearlas, con un tipo de dato y una longitud definida. Si no se realiza este

paso QGIS presume que la longitud del campo es la máxima y reserva un espacio en memoria que con la cantidad de información con la que se trata es demasiado grande. Como ejemplo sin definir el tamaño de los campos él .dbf final pesa unos 400mb mientras que cuando se define éste el tamaño baja a unos 22mb.

El nombre de los campos de las columnas es fijo, “Campo + tipo de dato + tiempo previsión UTC”

```
1.          layer_total.dataProvider().addAttributes( [ QgsField("C_U_"+str(cnt),
    QVariant.Double, "double", 5, 3)] )
2.          layer_total.dataProvider().addAttributes( [ QgsField("C_V_"+str(cnt),
    QVariant.Double, "double", 5, 3)] )
3.          layer_total.dataProvider().addAttributes( [ QgsField("C_T_"+str(cnt),
    QVariant.Double, "double", 5, 3)] )
4.          layer_total.dataProvider().addAttributes( [ QgsField("C_A_"+str(cnt),
    QVariant.Double, "double", 8, 1)] )
5.          layer_total.dataProvider().addAttributes( [ QgsField("C_TMP_"+str(cnt)
    , QVariant.Double, "double", 8, 3)] )
6.          layer_total.dataProvider().addAttributes( [ QgsField("C_H_"+str(cnt),
    QVariant.Double, "double", 5, 3)] )
```

*Código 25- Definición de los campos*

Seguidamente se procede a la carga de los datos dentro de la tabla, por ahora de la componente U y V del viento, temperatura y humedad.

```
1.          #Cargo los valores de los rows de las tablas u v y final
2.          fin_feats=layer_total.getFeatures()
3.          #Se presupone que los  datos siempre siguen el mismo orden, para ahorr
    ar tiempo de calculo
4.          cnt_attb=0
5.          for fin_feat in fin_feats:
6.              fin_attrs = fin_feat.attributes()
7.              puVal=float(dat_u[cnt_attb][4])
8.              pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_U_
    "+str(cnt)] : puVal}})
9.              pvVal=float(dat_v[cnt_attb][4])
10.             pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_V_
    "+str(cnt)] : pvVal}})
11.             ptFVal=float(dat_tF[cnt_attb][4])
12.             pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_TM
    P_"+str(cnt)] : ptFVal}})
13.             phVal=float(dat_h[cnt_attb][4])
14.             pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_H_
    "+str(cnt)] : phVal}})
15.             cnt_attb+=1
```

*Código 26- Carga de los atributos dentro del shapefile*

Pero además de los datos predefinidos dentro de los GRIB se ha de decidido

calcular dos campos más:

- La intensidad Total del Viento (componente cuadrática de U y V).
- El Azimut de la dirección del viento, a diferencia del Azimut topográfico el azimut del viento viene del sur.

```
#           Ahora que tenemos el valor de U y V se puede calcular el Azimut, d
esde el Sur, y el valor total
            Tval=math.sqrt(puVal**2+pvVal**2)
            dx=puVal
            dy=pvVal
            #Si alguno de estos valroes de 0 la funcion del azimut da error po
r lo que se le da un valor minusculo
            if dx==0:
                dx=0.00000000001
            if dy==0:
                dy=0.00000000001
            if dx>=0 and dy>=0:
                c=0
            if dx>=0 and dy<=0:
                c=180
            if dx<=0 and dy<=0:
                c=180
            if dx<=0 and dy>=0:
                c=360
            azi=c+math.atan((dx)/(dy))*180/math.pi
            azi=azi-180.0
            if azi<0:
                azi=azi+360.0
            if azi>360:
                azi=azi-360.0
            pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_T_
"+str(cnt)] : Tval}})
            pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_A_
"+str(cnt)] : azi}})
```

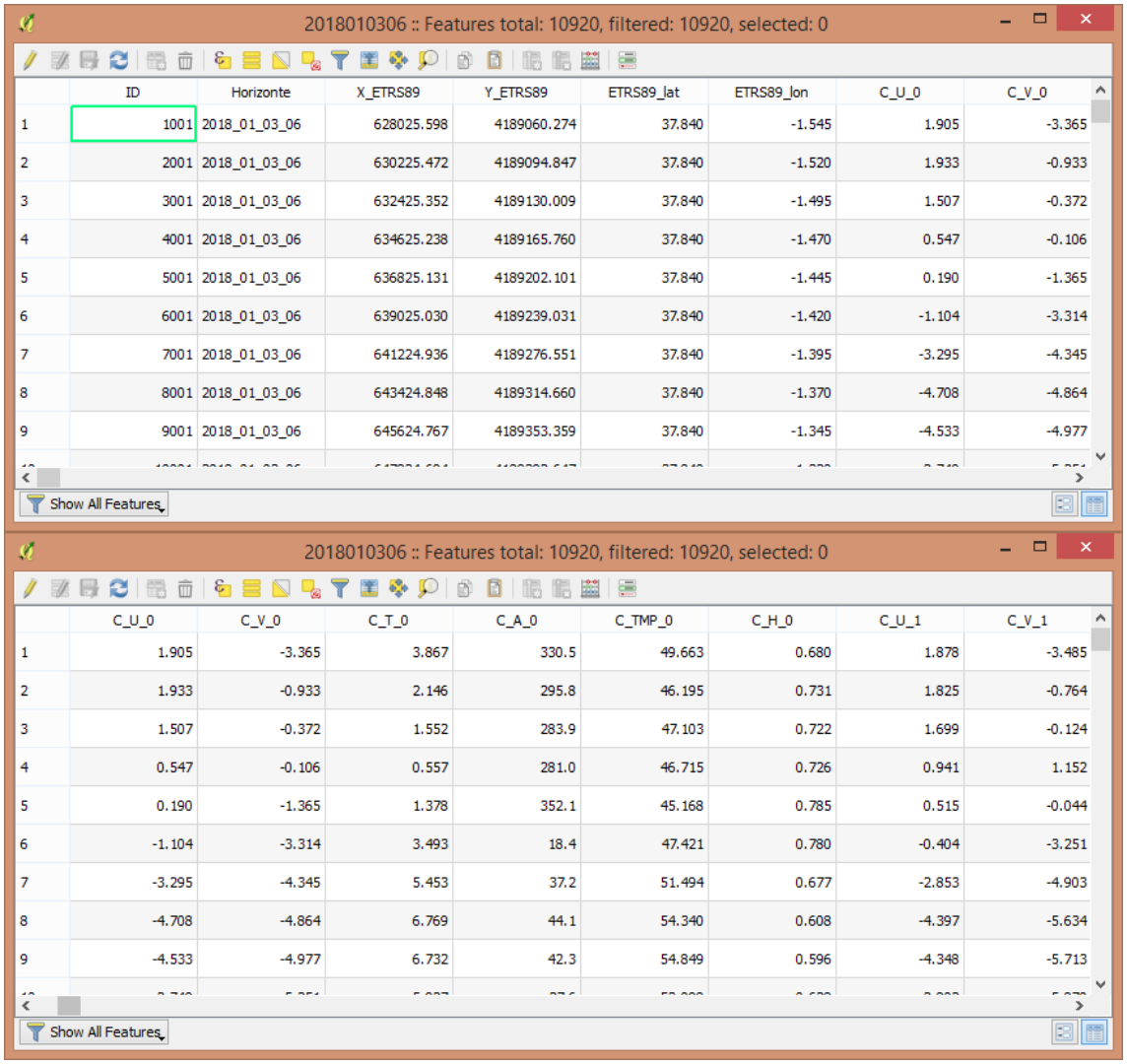
*Código 27- Carga de los nuevos atributos dentro del shapefile*

El último paso dentro del macro al igual que en el caso de las imágenes es salir fuera del proceso de QGIS y luego cerrarlo.

```
1. iface.actionExit().trigger()
2. os.kill(os.getpid(), 9)
```

*Código 28- Cierre de QGIS*

El resultado final de la tabla queda como la siguiente imagen:




| | ID | Horizonte | X_ETRS89 | Y_ETRS89 | ETRS89_lat | ETRS89_lon | C_U_0 | C_V_0 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1001 | 2018_01_03_06 | 628025.598 | 4189060.274 | 37.840 | -1.545 | 1.905 | -3.365 |
| 2 | 2001 | 2018_01_03_06 | 630225.472 | 4189094.847 | 37.840 | -1.520 | 1.933 | -0.933 |
| 3 | 3001 | 2018_01_03_06 | 632425.352 | 4189130.009 | 37.840 | -1.495 | 1.507 | -0.372 |
| 4 | 4001 | 2018_01_03_06 | 634625.238 | 4189165.760 | 37.840 | -1.470 | 0.547 | -0.106 |
| 5 | 5001 | 2018_01_03_06 | 636825.131 | 4189202.101 | 37.840 | -1.445 | 0.190 | -1.365 |
| 6 | 6001 | 2018_01_03_06 | 639025.030 | 4189239.031 | 37.840 | -1.420 | -1.104 | -3.314 |
| 7 | 7001 | 2018_01_03_06 | 641224.936 | 4189276.551 | 37.840 | -1.395 | -3.295 | -4.345 |
| 8 | 8001 | 2018_01_03_06 | 643424.848 | 4189314.660 | 37.840 | -1.370 | -4.708 | -4.864 |
| 9 | 9001 | 2018_01_03_06 | 645624.767 | 4189353.359 | 37.840 | -1.345 | -4.533 | -4.977 |

| | C_U_0 | C_V_0 | C_T_0 | C_A_0 | C_TMP_0 | C_H_0 | C_U_1 | C_V_1 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1.905 | -3.365 | 3.867 | 330.5 | 49.663 | 0.680 | 1.878 | -3.485 |
| 2 | 1.933 | -0.933 | 2.146 | 295.8 | 46.195 | 0.731 | 1.825 | -0.764 |
| 3 | 1.507 | -0.372 | 1.552 | 283.9 | 47.103 | 0.722 | 1.699 | -0.124 |
| 4 | 0.547 | -0.106 | 0.557 | 281.0 | 46.715 | 0.726 | 0.941 | 1.152 |
| 5 | 0.190 | -1.365 | 1.378 | 352.1 | 45.168 | 0.785 | 0.515 | -0.044 |
| 6 | -1.104 | -3.314 | 3.493 | 18.4 | 47.421 | 0.780 | -0.404 | -3.251 |
| 7 | -3.295 | -4.345 | 5.453 | 37.2 | 51.494 | 0.677 | -2.853 | -4.903 |
| 8 | -4.708 | -4.864 | 6.769 | 44.1 | 54.340 | 0.608 | -4.397 | -5.634 |
| 9 | -4.533 | -4.977 | 6.732 | 42.3 | 54.849 | 0.596 | -4.348 | -5.713 |

*Imagen 21- Muestra de los resultados finales del Shapefile*

### 3.6.5- Renombrado, guardado y movimiento al directorio intermedio

Los últimos pasos a la hora de crear el archivo del shapefile son:

- Renombrar el archivo para que coincida con la estructura “Año + Mes + Día + Época”.
- Guardar todos los archivos del shapefile en el histórico.
- Mover el Shapefile en formato .zip al directorio intermedio.

```
f.write('                    : Renombrando, moviendo y guardando el .SHP ...'+"\n")
f.flush()
#Renombrar los archivos .shp y eliminar los que no son los de la propia capa
for fil in os.listdir(path_shp_ini):
    if r"csv_" not in fil:
        os.rename(path_shp_ini+"\\"+fil, path_shp_ini+"\\"+resto+fil[-4:])
    else:
        os.remove(path_shp_ini+"\\"+fil)
#Compromir en .ZIP las capas
folder="shp"
convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
#Borrar los archivos de web
for fil in os.listdir(path_shp_dest):
    os.remove(path_shp_dest+"\\"+fil)
#Mover los .shp al directorio de la web
for fil in os.listdir(path_shp_ini):
    shutil.copy2(path_shp_ini+"\\"+fil, path_shp_dest+"\\"+resto+fil[-4:])
#Creacion del .zip para el geoserver dentro de la carpeta shp_web
folder="shp"
zip_name='shp_cmp'
for fil in os.listdir(path_shp_ini):
    if r"csv_" not in fil:
        os.rename(path_shp_ini+"\\"+fil, path_shp_ini+"\\shp_geoserver"+fil[-4:])
zipf = zipfile.ZipFile('{0}.zip'.format(os.path.join(path_shp_dest, zip_name)), 'w', zipfile.ZIP_DEFLATED)
for root, dirs, files in os.walk(os.path.join(path_datos, folder)):
    for filename in files:
        zipf.write(os.path.abspath(os.path.join(root, filename)), arcname=filename)
zipf.close()
f.write('                    : Se ha creado el .SHP de forma correcta'+"\n")
f.flush()
```

*Código 29- Renombrar y mover al histórico y al directorio intermedio el Shapefile*

## 3.7- Borrado de los datos locales

El siguiente paso que realiza el programa es el borrado de cualquier fichero dentro de las carpetas locales, para que la siguiente ejecución no tenga ningún error a la hora de cargar los datos.

```
1.  f.write('                    : Borrando los datos locales...'+"\n")
2.  f.flush()
3.  #Borrado de los datos de sus carpetas
4.  for fil in os.listdir(dir_destino_g1):
5.      os.remove(dir_destino_g1+"\\"+fil)
6.  for fil in os.listdir(path_grib2):
7.      os.remove(path_grib2+"\\"+fil)
8.  for fil in os.listdir(path_fotos_VD):
9.      os.remove(path_fotos_VD+"\\"+fil)
10. for fil in os.listdir(path_fotos_VE):
11.     os.remove(path_fotos_VE+"\\"+fil)
12. for fil in os.listdir(path_fotos_TE):
13.     os.remove(path_fotos_TE+"\\"+fil)
14. for fil in os.listdir(path_fotos_HR):
15.     os.remove(path_fotos_HR+"\\"+fil)
16. for fil in os.listdir(path_shp_ini):
17.     os.remove(path_shp_ini+"\\"+fil)
18. f.write('                    : Se han borrado los datos locales'+"\n")
19. f.flush()
```

*Código 30- Borrar los datos de las carpetas locales*

# 4.- MANEJO DE LOS DATOS EN GEOSERVER

Una vez se tenga los datos en el directorio intermedio hay que enviarlos al directorio de Geoserver.

Cabe destacar que a partir de este punto los directorios dejan de ser relativos y son absolutos, se ha seguido esta presuposición porque los directorios de Geoserver a la hora de la instalación por defecto son siempre iguales.

## 4.1- Envío de los datos a Geoserver

El siguiente paso es el de subir los datos del directorio intermedio a un directorio dentro de Geoserver, el directorio intermedio y el del Geoserver no son el mismo, el intermedio es un directorio intermedio entre el servidor donde se procesan los datos y el servidor donde se ubica Geoserver. Se vio este directorio en el apartado "*3.2.2.1*" de esta memoria.

Se envían las imágenes georreferenciadas, 3 archivos por imagen, y el shapefile comprimido en formato .zip.

```
1.  f.write('                    : Subiendo los datos a la carpeta de Geoserver'+"
    \n")
2.  f.flush()
3.  try:
4.      for fil in os.listdir(path_geoserver+r'\png_VE'):
5.          os.remove(path_geoserver+r'\png_VE'+"\\"+fil)
6.      for fil in os.listdir(path_geoserver+r'\png_HR'):
7.          os.remove(path_geoserver+r'\png_HR'+"\\"+fil)
8.      for fil in os.listdir(path_geoserver+r'\png_TE'):
9.          os.remove(path_geoserver+r'\png_TE'+"\\"+fil)
10.     for fil in os.listdir(path_geoserver+r'\shp'):
11.         os.remove(path_geoserver+r'\shp'+"\\"+fil)
12.
13.     for fil in os.listdir(path_fotos_web_VE):
14.         if 'horizonte' not in fil:
15.             shutil.copy2(path_fotos_web_VE+"\\"+fil, path_geoserver+r'\png_VE'
    +"\\"+fil)
16.     for fil in os.listdir(path_fotos_web_HR):
17.         if 'horizonte' not in fil:
18.             shutil.copy2(path_fotos_web_HR+"\\"+fil, path_geoserver+r'\png_HR'
    +"\\"+fil)
19.     for fil in os.listdir(path_fotos_web_TE):
20.         if 'horizonte' not in fil:
21.             shutil.copy2(path_fotos_web_TE+"\\"+fil, path_geoserver+r'\png_TE'
    +"\\"+fil)
22.     for fil in os.listdir(path_shp_dest):
```

```
23.         if '.zip' in fil:
24.          shutil.copy2(path_shp_dest+"\\"+fil, path_geoserver+r'\shp'+"\\"+fil)

25. except Exception as e:
26.     f.write('     ERROR          : Un error ha ocurrido subiendo los datos a l
    a carpeta de geoserver: '+str(e)+"\n")
```

*Código 31- Paso de los datos del directorio intermedio a Geoserver*

## 4.2.- Subida de los datos a Geoserver

Una vez se han subido los datos al directorio de Geoserver se llama la última función, **Geoserver_Curl.py** la cual se encarga de actualizar los datos del servidor de Geoserver. Recordemos que el server en el cual se crean los datos y donde luego se envían es distinto. Para ello se va a emplear el Curl, con los comandos de "delete" para eliminar capas, "put" para sobrescribir el shapefile y "post" para subir las imágenes.

El punto positivo de emplear esta metodología es que no hace falta reiniciar Geoserver, por lo que se puede acceder a otras capas, y el proceso en sí es muy rápido, tarda unos 5 segundos en completarse.

### 4.2.1.- Borrado de las imágenes en el Geoserver

El primer paso que realiza el programa es comprobar dentro de la carpeta **workspace** que imágenes hay publicadas actualmente, para borrarlas todas con la petición HTTP de "delete". Esto es posible a que Geoserver dispone de unas herramientas de conexión externas muy útiles denominadas REST.

Pero primero para poder hacer este tipo de acción hay que tener un usuario y contraseña dentro de Geoserver que tenga permisos de modificación de datos. En este caso se ha empleado el usuario con permisos de Administrador. Por razones de seguridad de la empresa para este proyecto se han borrado y sustituido con asteriscos la información de la conexión.

```
1.      headers = {'Content-type': 'text/xml'}
2.      #Nombre del Workspace
3.      wk='AEMET'
4.      #Directorio del Workspace, se conecta al servidor de Vaersa
```

```
    dr_wkspc=r"\\arcgis.vaersa.org"+"\\"+wk
    #Compreuba dentro del directorio que capas hay disponibles
    dirs = [d for d in os.listdir(dr_wkspc) if os.path.isdir(os.path.join(dr_wkspc, d))]
    #Borra todas las capas con imágenes
    for img_st in dirs:
        #Para que salte las carpetas del shape y de estilos
        if img_st!='AEMET_shp' and img_st!='styles':
            #ahora hay que borrar el storage junto con la capa, lo ahce el recurse
            r1 = requests.delete("http://arcgis.vaersa.org:8080/geoserver/rest/workspaces/"+wk+"/coveragestores/"+img_st+"?recurse=true",
                #Todas las conexiones requieren tener de antemano el usuario y contraseña de Geoserver, además de los permisos para poder modificar los datos.
                #En este caso se han cubierto por seguridad.
                auth=('******', '**********'),
                headers=headers)
```

*Código 32- Conexión al Geoserver y borrado de las imágenes actuales*

### 4.2.2.- Actualizado del Shapefile

Una vez se han borrado todas las imágenes se puede proceder a la actualización del Shapefile, en este caso no hace borrarlo porque lo único que hace es sustituirse una base de datos por otra, no como en el caso de las imágenes en el que además los nombres de las imágenes cambian de una ejecución a otra.

```
    #Actualiza el .shp del servidor, no hace falta borrarla previamente porque el nombre del .shp no cambia
    headers = {'Content-type':  "application/zip", 'Accept': 'application/xml'}
    bakemono=file(r'\\arcgis.vaersa.org\datos_visor_previsiones\shp\shp_cmp.zip','rb').read()
    r1 = requests.put("http://arcgis.vaersa.org:8080/geoserver/rest/workspaces/AEMET/datastores/AEMET_shp/file.shp",
        auth=('******', '**********'),
        data= bakemono,
        headers=headers)
```

*Código 33- Actualización del Shapefile*

### 4.2.3.- Subida de las Imágenes

El último paso es subir las imágenes a Geoserver, para ello se han hecho seis peticiones post, dos para cada tipo de imagen que se vaya a subir.

- Creación del CoverageStore (dónde se almacenan los datos)
- Creación de la propia capa (cómo tratar los datos)

Se va a poner de ejemplo solo el primer caso ya que los otros dos son análogos, si se quiere comprobar las diferencias entre los tres casos se puede ver ésta en el Anexo 9.1.14.

Lo que primero hace el programa es conectarse a dónde se encuentran los datos dentro del **root\data_dir** y recorrer las imágenes una por una para obtener su nombre. Una vez se tiene el nombre se crea dentro de una plantilla .xml los datos mínimos que necesita Geoserver para saber de dónde obtener los datos:

- El nombre del propio CoverageStore.
- El workspace de trabajo.
- Si la capa va a estar activa (visible).
- Tipo de dato, en este caso “WorldImage”.
- Dónde se encuentran la imagen.

Una vez creado del CoverageStore se procede a crear otra plantilla, en este caso del layer, para que Geoserver sepa cómo tratar la información que se encuentra en la carpeta de los directorios, para ello necesita saber:

- La extensión, que se calcula automáticamente en la petición POST.
- El nombre del Storage, para saber de dónde obtener los datos.
- El título que va a tener la capa.
- El sistema de Referencia.

```
1.      #Sube las imágenes de los vientos con flechas
2.      headers = {'Content-type': 'text/xml'}
3.      os.chdir(r"\\arcgis.vaersa.org\datos_visor_previsiones\png_VE")
```

```
    for filep in glob.glob("*.png"):
        if "output" not in filep:
            nombre=filep.split(r'.')[0]
            #Primero crea el CoverageStore, el de dónde se van a obtener los d
atos
            r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver/rest/w
orkspaces/"+wk+"/coveragestores",
                auth=('******', '**********'),
                data= '<coverageStore> \
                      <name>'+nombre+r'_storage'+'</name> \
                      <workspace>'+wk+'</workspace> \
                      <enabled>true</enabled> \
                      <type>WorldImage</type> \
                      <url>datos_visor_previsiones/png_VE/'+filep+'</url> \
                      </coverageStore>'  ,
                headers=headers)
            #Seguidamente crea el layer, en el caso de las imágenes sólo hace
falta el nombre de la capa, título, sistema de referencia y extensión, la cual
 se calcula automáticamente gracias al comando del POST.
            #La parte de recalculate se encarga de calcular la extension.
            r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver/rest/w
orkspaces/"+wk+"/coveragestores/"+nombre+r'_storage'+"/coverages?recalculate=n
ativebbox,latlonbbox",
                auth=('******', '**********'),
                data= '<coverage> \
                       <name>'+nombre+r'_coverage'+'</name>\
                       <title>'+nombre+r'_coverage'+'</title>\
                       <srs>EPSG:25830</srs>\
                       </coverage>' ,
                headers=headers)
```

*Código 34- Subida de las imágenes de Vientos a Geoserver*

### 4.2.4.- Dentro de Geoserver

Una vez subidos los datos aparecen de la siguiente manera dentro de la página WEB de Geoserver:

| | | | | |
|---|---|---|---|---|
| ☐ | ▩ | HR_2017_12_18_00_coverage | AEMET:HR_2017_12_18_00_coverage | HR_2017_12_18_00_storage |
| ☐ | ▩ | HR_2017_12_18_01_coverage | AEMET:HR_2017_12_18_01_coverage | HR_2017_12_18_01_storage |
| ☐ | ▩ | HR_2017_12_18_02_coverage | AEMET:HR_2017_12_18_02_coverage | HR_2017_12_18_02_storage |
| ☐ | ▩ | HR_2017_12_18_03_coverage | AEMET:HR_2017_12_18_03_coverage | HR_2017_12_18_03_storage |
| ☐ | ▩ | HR_2017_12_18_04_coverage | AEMET:HR_2017_12_18_04_coverage | HR_2017_12_18_04_storage |
| ☐ | ▩ | HR_2017_12_18_05_coverage | AEMET:HR_2017_12_18_05_coverage | HR_2017_12_18_05_storage |
| ☐ | ▩ | HR_2017_12_18_06_coverage | AEMET:HR_2017_12_18_06_coverage | HR_2017_12_18_06_storage |
| ☐ | ▩ | HR_2017_12_18_07_coverage | AEMET:HR_2017_12_18_07_coverage | HR_2017_12_18_07_storage |
| ☐ | ▩ | HR_2017_12_18_08_coverage | AEMET:HR_2017_12_18_08_coverage | HR_2017_12_18_08_storage |
| ☐ | ▩ | HR_2017_12_18_09_coverage | AEMET:HR_2017_12_18_09_coverage | HR_2017_12_18_09_storage |

*Imagen 22- Muestra de las capas de imágenes en Geoserver*

Donde se puede apreciar que en este caso se han calculado las imágenes en la época “00” para el día 18 de Diciembre de 2017.

Mientras que el Shapefile siempre tendrá el mismo nombre, sólo se actualizan los datos de éste:

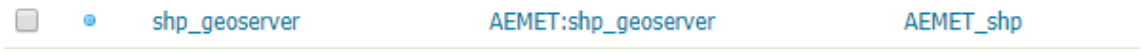



*Imagen 23- Muestra del shapefile en Geoserver*

# 5.- MENSAJE FINAL

Una vez se haya acabado el proceso entero de la creación de los datos, su subida al servidor y su subida a Geoserver el archivo de status.log tendrá la siguiente forma:

```
Log del: 2018010306 : Los archivos fueron encontrados
                    : Moviendo los datos...
                    : Los datos de GRIB1 fueron movidos
                    : Convirtiendo a GRIB2...
                    : Los GRIB2 se han creado satisfactoriamente
                    : Creando el .zip del GRIB1...
                    : Los datos de GRIB1 se han guardado en el historico
                    : Creando el historico del GRIB2...
                    : Los datos de GRIB2 se han guardado en el historico
                    : Ejecutando el QGIS para la creacion de las imagenes de viento con flechas...
                    : Se han creado las imagenes
                    : Renombrando y guarando las fotos...
                    : Las fotos fueron renombrados y guardados
                    : Moviendo las fotos...
                    : Las fotos fueron movidos al directorio C:\meteoro_procesos\viento_aemet_2018\fotos_web_VD
                    : Borrando las fotos antiguas...
                    : Ha acabado de hacer las fotos de los Vientos con flechas

                    : Creando las miniaturas del viento con flechas
                    : Ha acabado de crear las miniaturas
                    : Ejecutando el QGIS para la creacion de las imagenes georeferenciadas...
                    : Las imagenes georeferenciadas fueron creadas
                    : Renombrando y guarando las fotos georeferenciadas...
                    : Se han renombrado y guardado las imagenes georeferenciadas
                    : Moviendo las fotos georeferenciadas...
                    : Se han movido las fotos al directorio C:\meteoro_procesos\viento_aemet_2018\fotos_web_VE
                    : Borrando las fotos antiguas...
                    : Las imagenes de los vientos se han creado correctamente

                    : Creando las miniaturas del viento
                    : Ha acabado de crear las miniaturas
                    : Ejecutando el QGIS para la creacion de las imagenes georeferenciadas de humedades...
                    : Las imagenes georeferenciadas de humedades fueron creadas
                    : Renombrando y guarando las fotos georeferenciadas de humedades...
                    : Se han renombrado y guardado las imagenes georeferenciadas de humedades
                    : Moviendo las fotos georeferenciadas...
                    : Se han movido las fotos al directorio C:\meteoro_procesos\viento_aemet_2018\fotos_web_HR
                    : Borrando las fotos antiguas de humedades...
                    : Las imagenes de humedades fueron creadas de forma correcta
                    : Creando las miniaturas de humedades
                    : Ha acabado de crear las miniaturas
```

```
40.                     : Ejecutando el QGIS para la creacion de las imagenes geor
    eferenciadas de temperaturas...
41.                     : Las imagenes georeferenciadas de temperaturas fueron cre
    adas
42.                     : Renombrando y guarando las fotos georeferenciadas de tem
    peraturas...
43.                     : Se han renombrado y guardado las imagenes georeferenciad
    as de temperaturas
44.                     : Moviendo las fotos georeferenciadas...
45.                     : Se han movido las fotos al directorio C:\meteoro_proceso
    s\viento_aemet_2018\fotos_web_TE
46.                     : Borrando las fotos antiguas de temperaturas...
47.                     : Las fotos de las temperaturas se han creado de forma cor
    recta
48.                     : Creando las miniaturas de temperatura
49.                     : Ha acabado de crear las miniaturas
50.                     : Creando los archivos de horizonte
51.                     : Ejecutando el QGIS para la creacion de los .SHP...
52.                     : Se ha creado el .SHP con todas las horas
53.                     : Renombrando, moviendo y guardando el .SHP ...
54.                     : Se ha creado el .SHP de forma correcta
55.                     : Borrando los datos locales...
56.                     : Se han borrado los datos locales
57.                     : Subiendo los datos a la carpeta de Geoserver
58.                     : Ha empezado a subir archivos a geoserver
59.                     : Ha acabado de subir archivos a geoserver
60.                     : FINALIZADO
```

*Imagen 24- Mensaje final del .log*

# 6.- VISOR ON-LINE

Para finalizar el proyecto se he requerido hacer un simple visor online en el cual se pueda llegar a visualizar los datos, se han puesto los siguientes requisitos:

- Que funcione en diferentes navegadores.
- Tenga una barra deslizadora que permita cambiar entre fechas.
- El cambio entre los distintos fenómenos.
- Que contenga la leyenda del fenómeno que se represente.

## 6.1.- HTML y CSS

El HTML y el CSS se encargan de dar la forma a la página WEB:

- HTML: Se encarga de crear los elementos básicos de la página WEB, es el esqueleto.
- CSS: Se encarga de darle estilo a la página creada, tipo de letra, colores, posición de los distintos elementos

### 6.1.1.- Cabecera

Lo primero que hace el HTML es llamar a las distintas librerías de JavaScript:

- Openlayers, para crear los mapas.
- Proj4, para crear un sistema de referencia personalizado.
- Boostrap: Para facilitar la ceración de botonos, barras deslizadoras….
- Visor: JavasScript creado para poder ver el visor en el HTML, este fue creado por mi.

```
1.    <head>
2.      <meta charset="utf-8" />
3.      <title>Image ArcGIS MapServer</title>
4.      <link rel="stylesheet" href="https://openlayers.org/en/v4.2.0/css/ol.css"
    type="text/css">
5.      <!-
    - The line below is only needed for old environments like Internet Explorer an
    d Android 4.x -->
6.      <script src="https://cdn.polyfill.io/v2/polyfill.min.js?features=requestAn
    imationFrame,Element.prototype.classList,URL"></script>
7.      <script src="https://openlayers.org/en/v4.2.0/build/ol.js"></script>
```

```
    <script src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/1.3.3/FileSaver.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/proj4js/2.3.12/proj4-src.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/proj4js/2.3.12/proj4.js"></script>
    <script src="https://code.jquery.com/jquery-2.2.3.min.js"></script>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

    <!-- Optional theme -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

    <!-- Latest compiled and minified JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

    <script type = "text/javascript" src = "visor.js"></script>
  </head>
```

*Código 35- Cabecera del HTML*

### 7.1.2- El cuerpo

Aquí se definen las distintas partes de la página WEB, el dónde se va a ubicar el visor con qué tamaño, los botones, leyenda, desplegables…

```
  <body>
    <div id="cont_map" >
        <div id="cargando">
            <img src="loading2.gif" style="width: 300px;height: auto;position: absolute;z-index:2;top:100px;">
            <div style="height: 500px;width: 300px;background-color:#A9A9A9;opacity:0.5;position:absolute;z-index:1;"></div>
        </div>
        <div id="map" class="map"></div>
        <div id="capas" style="border: 1px solid black;width: 300px;padding:10px;float: left;">
            <p>Activar/Desactivar Capas</p>
            <label><input type="checkbox" checked name="vehicle" onclick="capa_vis();"> Municipios</label><br>
            <label><input type="checkbox" checked name="vehicle" onclick="capa_vis();"> Variable</label><br>
            <label><input type="checkbox" checked name="vehicle" onclick="capa_vis();"> Dirección del viento</label><br>
        </div>
        <div id="contendor_barra">
            <div style="padding-bottom:5%">
```

```
16.                 <input type="range" min="0" max="48" value="0" class="slider"
    id="valor_Rango" title="Texto del hover">
17.             </div>
18.             <p>Horizonte de la Previsión: <span id="fecha_img"></span></p>
19.             <p>Previsión elaborada el: <span id="fecha_creacion"></span></p>
20.         </div>
21.         <div id="contendor_Leyenda" style="border: 1px solid black;width: 300p
    x;height: 312px;padding:10px;float: left;">
22.             <p>Leyenda (Km/h)</p>
23.             <img src="leyenda_VE.PNG" >
24.         </div>
25.         <select id="seleccion_visor" style="position:absolute;left:100px;top:5
    05px;">
26.           <option >Viento</option>
27.           <option >Temperatura</option>
28.           <option >Humedad</option>
29.         </select>
30.     </div>
```

*Código 36- El cuerpo del HTML*

### 7.1.3- Estilo (CSS)

Finalmente la última parte del .HTML son los estilos, se ha decidido ubicarlos dentro del propio archivo .html debido a su reducido tamaño. En esta parte se definen los estilos que se llaman desde el apartado anterior para darle como su nombre indica el estilo.

```
1.      <style>
2.      label{
3.       font-weight : normal;
4.      }
5.      #cont_map {
6.          margin:0;
7.          padding:0;
8.          height: 500px;
9.          width: 600px;
10.     }
11.     #map{
12.         height: 500px;
13.         width: 300px;
14.         border: 1px solid black;
15.         z-index:0;
16.         float: left;
17.     }
18.     #contendor_barra {
19.         float:left;
20.         border: 1px solid black;
21.         padding:10px;
22.         width: 300px;
23.     }
24.
25.     .slider {
26.         -webkit-appearance: none;
27.         width: 100%;
```

```
28.         /*height: 5px;*/
29.         border-radius: 5px;
30.         background: #d3d3d3;
31.         outline: none;
32.         opacity: 0.7;
33.         -webkit-transition: .2s;
34.         transition: opacity .2s;
35.     }
36.
37.
38.     .slider:hover {
39.         opacity: 1;
40.     }
41.
42.     .slider::-webkit-slider-thumb {
43.         -webkit-appearance: none;
44.         appearance: none;
45.         width: 10px;
46.         height: 15px;
47.         border-radius: 30%;
48.         background: #4CAF50;
49.         cursor: pointer;
50.     }
51.
52.     .slider::-moz-range-thumb {
53.         width: 25px;
54.         height: 25px;
55.         border-radius: 50%;
56.         background: #4CAF50;
57.         cursor: pointer;
58.     }
59.     </style>
```

*Código 37- Definición del CSS*

## 6.2.- JavaScript

En esta parte es dónde recae la mayoría del trabajo referente al visor on-line, básicamente se encarga de:

- Dibujar los mapas
- Cambiar los mapas cuando se mueve la barra deslizadora
- Detectar cuándo se han cargado todas las capas
- Conectarse a la base de datos de Geoserver
- Cambiar la leyenda y los valores horarios.

### 6.2.1.- Creación del mapa base

La primera parte del programa carga los elementos dentro del HTML que van a sufrir una modificación, no relacionados con el mapa.

```
1.      var slider = document.getElementById("valor_Rango");
2.      var output = document.getElementById("fecha_img");
3.      var output_crea = document.getElementById("fecha_creacion");
```

*Código 38- Llamado a los elemtos del HTML que se van a modificar dinámicamente*

Seguidamente se define una proyección personalizada porque las librerías de Openlayers no contienen la proyección “EPSG:25830”.

```
1.      var myProjectionName = 'EPSG:25830';
2.      proj4.defs(myProjectionName, "+proj=utm +zone=30 +ellps=GRS80 +units=m +no
    _defs");
```

*Código 39- Creación de la proyección EPSG:25830*

El siguiente paso es crear las capas que se van a ver dentro del visor, una de ellas se puede llamar sin problema porque va a ser fija (la de los municipios), pero las otras se crean vacías estableciendo solo el nombre que van a tener.

```
1.      urlsig="http://arcgis.vaersa.org/ArcGIS/rest/services/contorno_municipios/
    MapServer"
2.      layer_muni=new ol.layer.Image({
3.          title: 'Municipios',
4.          name: 'Municipios',
5.          source: new ol.source.ImageArcGISRest({
6.              url: urlsig,
7.              crossOrigin: 'anonymous',
8.              rendererOptions: { zIndexing: true },
9.              params:{LAYERS:"show:0", }
10.         })
11.     })
12.     layer = new ol.layer.Image({
13.           title: 'Imagen',
14.           name: 'Imagen de Intensidades',
15.         })
16.     flechas = new ol.layer.Image({
17.           title: 'Imagen',
18.           name: 'Flechas',
19.         })
20.
21.     var layers = [
22.             layer,
23.             layer_muni,
24.             flechas
25.
26.         ];
```

```
    map = new ol.Map({
        controls: [mousePositionControl],

        layers: layers,
        logo: false,
        target: 'map',

        view: new ol.View({
            projection: 'EPSG:25830',
            center: ol.proj.transform([-
0.5, 39.5],'EPSG:4326','EPSG:25830'),
            zoom: 8,
            zoomFactor: 1.8
        }),
    });
    //Se hace Zoom a la Comunidad Valenciana
    map.getView().fit([626000, 4190000, 815000, 4519000], map.getSize());
```

*Código 40- Creación de las capas base*

### 6.2.2.- Conexión a la base de datos de Geoserver

El siguiente paso que realiza el programa es conectarse a la base de datos de Geoserver, para ser exactos a la capa Shapefile que contiene todos los datos.

Para ser precisos se conecta solo al primer punto a la columna “Horizonte” para rescatar el valor de éste. Este paso es muy importante porque a partir de este dato se obtiene el año, mes, día y época sobre la cual se han calculado los datos. A partir de éste dato se obtendrán las imágenes, porque éstas no tienen siempre el mismo nombre, sino que depende de la época de su cálculo.

El dato del horizonte viene en un diccionario JSON, por lo que hay que primero rescatarlo de dentro.

```
    //Rescato el primer punto de los 11k y solo el atributo del horizonte para
 saber cuando se calcularon los datos
    var url = "http://localhost:8080/geoserver/wfs?service=wfs&version=1.0.0&r
equest=GetFeature&typeName=AEMET:shp_geoserver&outputformat=text/javascript&fo
rmat_options=callback:getJson&cql_filter=ID=1001&propertyName=Horizonte";
    $.ajax({jsonp: false,jsonpCallback: 'getJson',type: 'GET',url: url,async:
false,dataType: 'jsonp',success: function(response) {
        tod_features=response.features;
        console.log(tod_features)
        horizonte=tod_features[0].properties.Horizonte
```

*Código 41- Obtener el dato de horizonte inicial en JSONP*

El siguiente paso es pasar la hora del horizonte inicial, que como se ha explicado ya, está en UTC y hay que pasarla a la a hora local para poder interpretar mejor los datos:

```
1.          var temp_part=horizonte.split('_')
2.          var hor_js= new Date( Date.UTC(temp_part[0],temp_part[1]-
    1,temp_part[2],temp_part[3]) );
3.          hor_js.setHours(hor_js.getHours())
4.          var any_c=hor_js.getFullYear().toString()
5.          var mes_c=(hor_js.getMonth()+1).toString()
6.          if (mes_c.length==1){
7.              mes_c="0"+mes_c
8.          }
9.          var dia_c=hor_js.getDate().toString()
10.         if (dia_c.length==1){
11.             dia_c="0"+dia_c
12.         }
13.         var h_c=hor_js.getHours().toString()
14.         if (h_c.length==1){
15.             h_c="0"+h_c
16.         }
17.         var fecha_img_viento=any_c+"/"+mes_c+"/"+dia_c+" "+h_c+"h"
18.         output_crea.innerHTML = fecha_img_viento;
```

*Código 42- Pasar de hora UTC a hora local*

### 6.2.3.- Creación de los mapas de los Fenómenos

Una vez ya se sabe la primera hora de previsión, el horizonte inicial, se pueden empezar a hacer peticiones al servidor para obtener las imágenes.

Para saber la fecha de la imagen que se quiera obtener se rescata el valor de la barra deslizadora, el rango de la cual va de 0-48. Dependiendo del valor de dicha barra se le sumará horas al horizonte inicial, en UTC, para saber la imagen que se está pidiendo. Para saber el tipo de imagen que se está pidiendo el programa mira el valor de selección del usuario del tipo de fenómeno que se quiera representar en un desplegable (por defecto cunado se carga la página web el primero en aparecer es el fenómeno de los Vientos).

Una vez se sabe toda esa información se monta un nuevo objeto, que contiene la información de dónde debe pedir OpenLayers la información y se actualiza dicha información sobre uno de las capas vacías que se ha creado

al principio del programa.

```
        var temp_part=horizonte.split('_')

        //console.log(temp_part)

        var hor_js= new Date(temp_part[0],temp_part[1]-
1,temp_part[2],temp_part[3] );
        hor_js.setHours(parseInt(hor_js.getHours()) + parseInt(slider.value))

        var any_c=hor_js.getFullYear().toString()
        var mes_c=(hor_js.getMonth()+1).toString()
        if (mes_c.length==1){
            mes_c="0"+mes_c
        }
        var dia_c=hor_js.getDate().toString()
        if (dia_c.length==1){
            dia_c="0"+dia_c
        }
        var h_c=hor_js.getHours().toString()
        if (h_c.length==1){
            h_c="0"+h_c
        }

        if (document.getElementById("seleccion_visor").value=='Viento'){
            prefijo='VD'
        }else if (document.getElementById("seleccion_visor").value=='Temperatu
ra'){
            prefijo='TE'
        }else{
            prefijo='HR'
        }


        //console.log(any_c,mes_c,dia_c,h_c)
        var fecha_img_viento=any_c+"_"+mes_c+"_"+dia_c+"_"+h_c
        //console.log(fecha_img)
        source_layer= new ol.source.ImageWMS({
            url: 'http://localhost:8080/geoserver/AEMET/wms',
            params: {'LAYERS': 'AEMET:'+prefijo+'_'+fecha_img_viento+'_coverag
e'},
            ratio: 1,
            serverType: 'geoserver'
          })
        //MUY IMPORANTE, le dice al OL que el source de la capa (layer) ha cam
biado
        layer.setSource(source_layer)
```

*Código 43- Obtención de la imagen seleccionada*

### 6.2.4.- Creación de las flechas para los Vientos

Esta parte de código comprueba que la imagen que se está pidiendo es el fenómeno de los Vientos, si ese es el caso dibuja las flechas en el mapa.

Hay que destacar un punto muy importante en esta parte. Si se hace una petición normal a la capa aparecerá una cosa como la siguiente imagen:

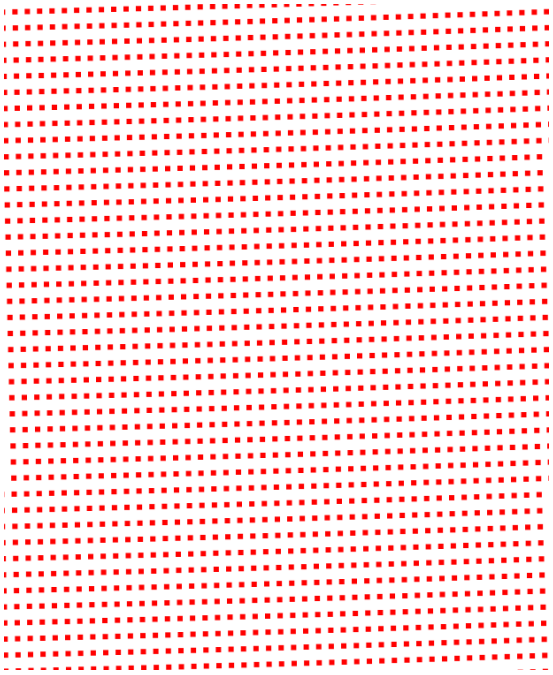


*Imagen 25- Muestra del Shapefile con estilo base*

Como se puede apreciar es una malla de puntos, pero sin ninguna flecha, que al final es lo que se quiere obtener.

Este problema se puede abordar de tres formas:

- Cargar la información de la capa a OpenLayers y dibujar las flechas con las direcciones a partir del JSON obtenido. Esta opción fue la originalmente elegida para el caso de la malla de 200 puntos y funcionaba sin ningún problema, pero al aumentar dicha malla a 11.000 puntos con más de 3.000.000 registros se sobrecargaba Openlayers y funcionaba muy lento, en ordenadores con menos RAM incluso se llegaba a colapsar la página web.
- La segunda opción sería crear tantos Shapefiles como imágenes de viento y subirlos a Geoserver. El siguiente paso sería crear 49 estilos, uno para cada una de las horas. Este proceso ocuparía demasiada memoria en el servidor

- La última opción es tener una capa Shapefile con un estilo predefinido e ir cambiándolo cada vez que se haga una petición. Esta es la opción que se ha escogido como la buena, porque no satura al usuario y tampoco al servidor.

A partir del valor que tenga la barra deslizadora se rescata su valor, entre 0 y 49, porque es el valor que tendrá el campo de Azimut en la base de datos si se recuerda su nomenclatura del paso 3.6.4.

```
1.          if (document.getElementById("seleccion_visor").value=='Viento'){
2.              var sldBody =
3.              '<?xml version="1.0" encoding="ISO-8859-1"?>\
4.              <StyledLayerDescriptor version="1.0.0"\
5.              xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescript
    or.xsd"\
6.              xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.n
    et/ogc"\
7.              xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w
    3.org/2001/XMLSchema-instance">\
8.                <NamedLayer>\
9.                  <Name>AEMET:shp_geoserver</Name>\
10.                 <UserStyle>\
11.                   <Title>Point Stacker</Title>\
12.                   <Abstract>A sample style that aggregates points</Abstract>\

13.                   <FeatureTypeStyle>\
14.                     <Rule>\
15.                       <TextSymbolizer>\
16.                         <Label><![CDATA[ ]]></Label> <!-- fake label -->\
17.                         <Graphic>\
18.                           <Mark>\
19.                             <WellKnownName>ttf://Segoe UI Symbol#0x2193</Well
    KnownName>\
20.                             <Fill>\
21.                               <CssParameter name="fill">#000000</CssParameter
    >\
22.                             </Fill>\
23.                           </Mark>\
24.                         <Size>20</Size>\
25.                         <Rotation>\
26.                           <PropertyName>C_A_'+slider.value.toString()+'</Prop
    ertyName>\
27.                         </Rotation>\
28.                       </Graphic>\
29.                       <LabelPlacement>\
30.                         <PointPlacement>\
31.                           <AnchorPoint>\
32.                             <AnchorPointX>1.0</AnchorPointX>\
33.                             <AnchorPointY>1.0</AnchorPointY>\
34.                           </AnchorPoint>\
35.                           <Displacement>\
36.                             <DisplacementX>0</DisplacementX>\
37.                             <DisplacementY>-5</DisplacementY>\
38.                           </Displacement>\
39.                         </PointPlacement>\
40.                       </LabelPlacement>\
```

```
                    </TextSymbolizer>\
                  </Rule>\
                </FeatureTypeStyle>\
              </UserStyle>\
            </NamedLayer>\
          </StyledLayerDescriptor>' ;
          source_flechas= new ol.source.ImageWMS({
              url: 'http://localhost:8080/geoserver/AEMET/wms',
              params: {'LAYERS': 'AEMET:shp_geoserver', 'SLD_BODY' : sldBody },
              ratio: 1,
              serverType: 'geoserver'
            })
          //MUY IMPORANTE, le dice al OL que el source de la capa (layer) ha cambiado
          flechas.setSource(source_flechas)
```

*Código 44- Creación del estilo de la capa de las flechas del Viento*

### 6.2.5.- Detección de la carga de los datos

El siguiente paso es crear una imagen que se sobreponga con el visor hasta que todos los datos estén listos. Se crea un contador que cuenta el número de peticiones que se han hecho de las capas y otro resta valores al primero cuando se detecta que una capa se ha cargado. Cuando el contador llegue a 0, que significa que todas las capas se han cargado, se deshabilitará la imagen que cubre el visor.

```
        for (i=0; i<layers.length; i++) {
         laker=layers[i]

         laker.getSource().on('imageloadstart', function(event) {
             layerCounter++;
         });
         laker.getSource().on('imageloadend', function(event) {
             layerCounter--;
             if (layerCounter == 0) {
                 document.getElementById('cargando').style.display = 'none'
             }
         });
        }
```

*Código 45- Detección de la carga de las capas*

### 6.2.6.- Cambio entre fenómenos

El último paso del programa es actualizar la página web para cuando se cambia el tipo de fenómeno elegido. Básicamente lo que se hace es:

- Actualizar la imagen de la leyenda.
- Actualizar en nombre de la leyenda.
- Activar/desactivar el checkbox de las flechas.
- Poner/Quitar el check sobre el checkbox de las flechas.
- Poner opacidad sobre el nombre de la capa con flechas.
- Activar/Desactivar la capa de las flechas.

```
1.      if (document.getElementById("seleccion_visor").value=='Viento'){
2.          document.getElementById('contendor_Leyenda').getElementsByTagName('img
    ')[0].src='leyenda_VE.PNG'
3.          document.getElementById('contendor_Leyenda').getElementsByTagName('p')
    [0].innerHTML='Leyenda (Km/h)'
4.          document.getElementById('capas').getElementsByTagName('input')[2].disa
    bled = false;
5.          document.getElementById('capas').getElementsByTagName('input')[2].chec
    ked = true;
6.          document.getElementById('capas').getElementsByTagName('label')[2].styl
    e.opacity=1;
7.          flechas.setVisible(true)
8.      }else if (document.getElementById("seleccion_visor").value=='Temperatura')
    {
9.          document.getElementById('contendor_Leyenda').getElementsByTagName('img
    ')[0].src='leyenda_TEv2.PNG'
10.         document.getElementById('contendor_Leyenda').getElementsByTagName('p')
    [0].innerHTML='Leyenda (ºC)'
11.         document.getElementById('capas').getElementsByTagName('input')[2].disa
    bled = true;
12.         document.getElementById('capas').getElementsByTagName('input')[2].chec
    ked = false;
13.         document.getElementById('capas').getElementsByTagName('label')[2].styl
    e.opacity=0.6;
14.         flechas.setVisible(false)
15.     }else{
16.         document.getElementById('contendor_Leyenda').getElementsByTagName('img
    ')[0].src='leyenda_HR.PNG'
17.         document.getElementById('contendor_Leyenda').getElementsByTagName('p')
    [0].innerHTML='Leyenda (%)'
18.         document.getElementById('capas').getElementsByTagName('input')[2].disa
    bled = true;
19.         document.getElementById('capas').getElementsByTagName('input')[2].chec
    ked = false;
20.         document.getElementById('capas').getElementsByTagName('label')[2].styl
    e.opacity=0.6;
21.         flechas.setVisible(false)
22.     }
```

*Código 46- Actualización de la página a la hora de cambiar entre fenómenos*

Finalmente se programa el Activado/Desactivado de las capas base:

```
1.      if (document.getElementById('capas').getElementsByTagName('input')[2].chec
    ked) {
2.          flechas.setVisible(true)
3.      }else{
4.          flechas.setVisible(false)
5.      }
6.      if (document.getElementById('capas').getElementsByTagName('input')[0].chec
    ked) {
7.          layer_muni.setVisible(true)
8.      }else{
9.          layer_muni.setVisible(false)
10.     }
11.     if (document.getElementById('capas').getElementsByTagName('input')[1].chec
    ked) {
12.         layer.setVisible(true)
13.     }else{
14.         layer.setVisible(false)
15.     }
```

*Código 47- Opciones de Activar/Desactivar las capas*

## 6.3.- Resultado Final

Los resultados finales para cada uno de los casos son los siguientes:

- Para el caso de los Vientos

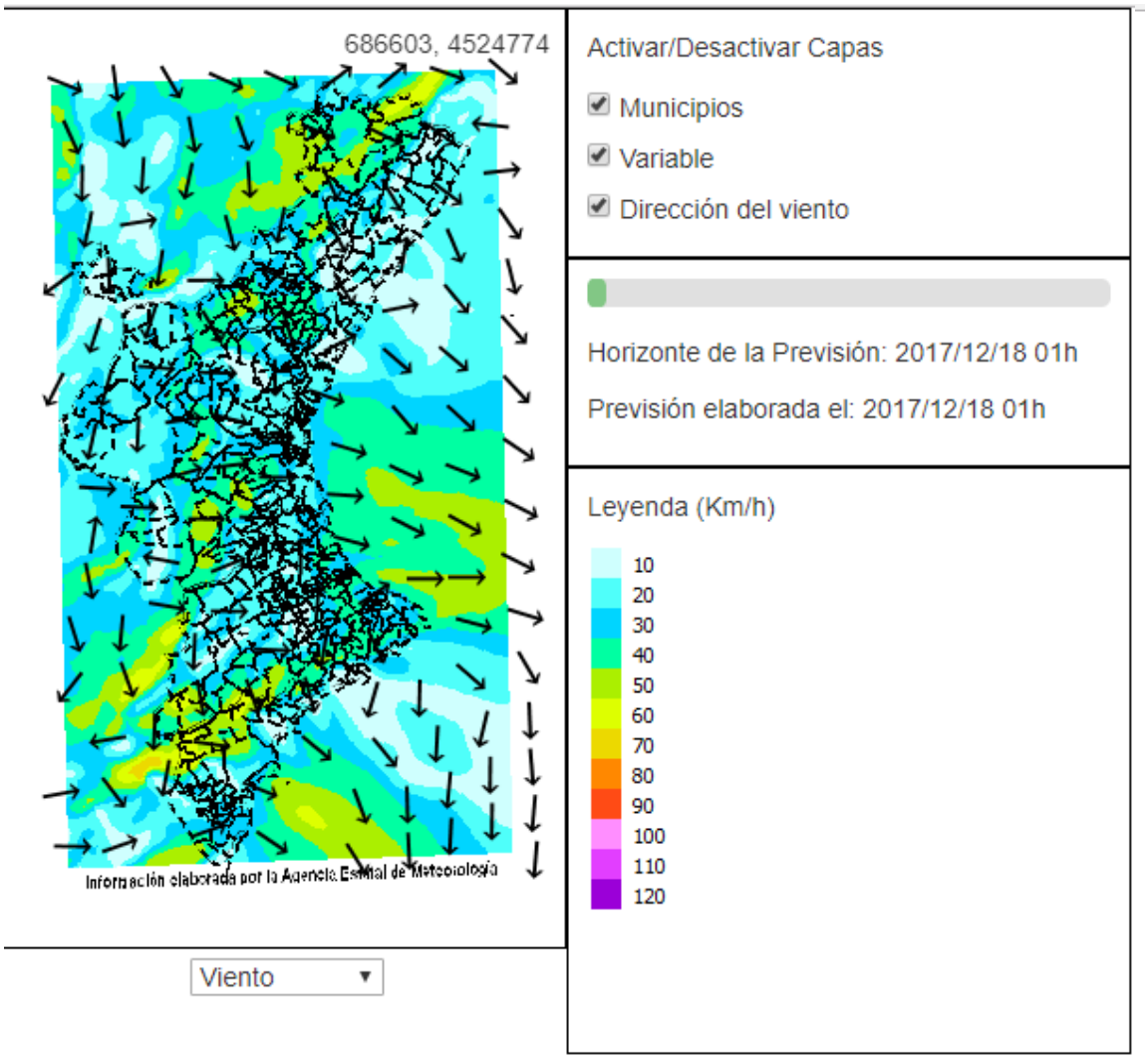



*Imagen 26- Resultado WEB final para Vientos*

- Para el caso de las Temperaturas

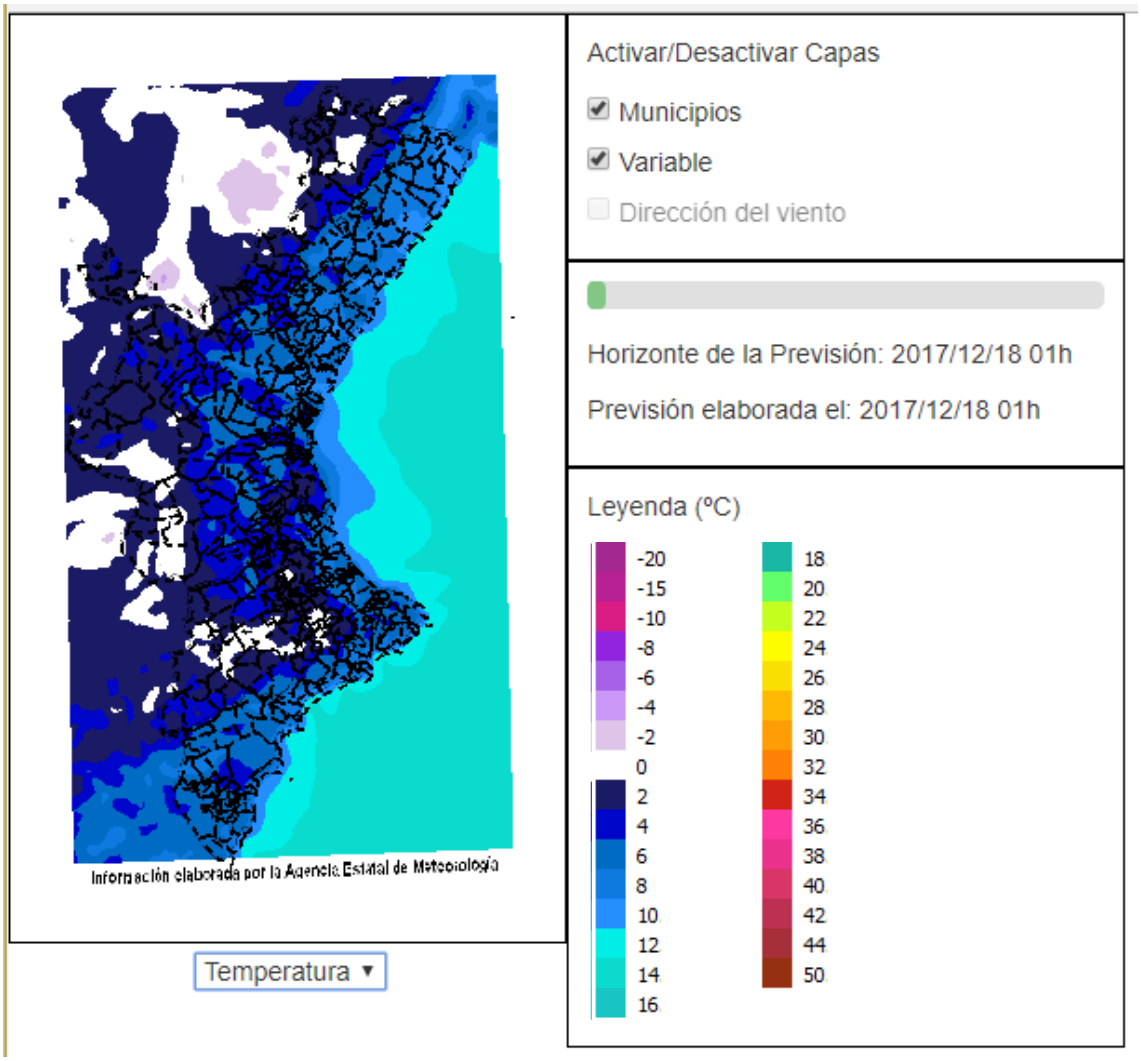



*Imagen 27- Resultado WEB final para Temperaturas*

- Para el caso de las Humedades:

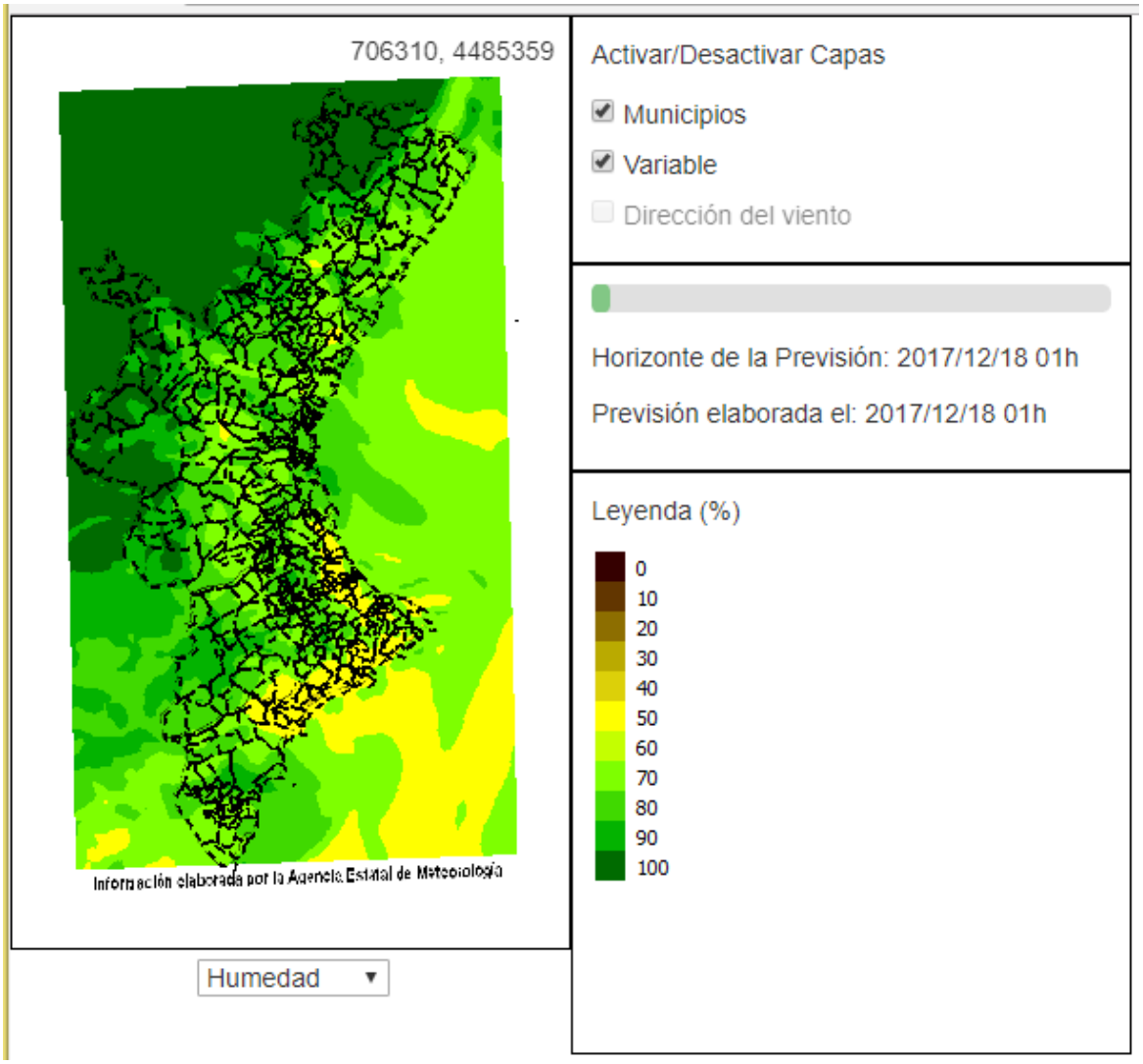



*Imagen 28- Resultado WEB final para Humedades*

Este es el resultado para la empresa Vaersa, que requerían de un estilo lo más simplista. Esto fue hecho porque la página web en la cual se va a ubicar el visor seguía en desarrollo, por lo que los estilos de letra, colores, tamaños, etc no estaban definidos del todo por lo que se decidió a que lo harían los informáticos al acabar dicha página.

# 7.- MODIFICACIÓN DE LOS DATOS DE ENTRADA, A ÚLTIMA HORA

Debo mencionar que una gran parte del código ha sido modificado porque, como he comentado al principio de la memoria, poco antes de acabar mis prácticas la empresa ha decidido cambiar los datos contratados, por una parte era más fácil porque la predicción pasaba de las siguientes 72 horas a 48 horas, pero por otra la malla de puntos pasaba de 200 a 11.000, por lo que el número de datos que se tenían que almacenar aumentó considerablemente y el número de fenómenos que se había de representar también aumentó.

El primer problema era el cambio de los datos, cómo tenía poco tiempo he decidido reciclar el mismo código que había hecho para el caso de sólo Vientos, porque el código estaba en modo de pruebas desde hace mas de dos meses en el servidor por lo que problemas que hayan ido surgiendo con este se han ido podiendo arreglar, mayoritariamente las primeras dos semanas.

El otro problema era la gran cantidad de datos que había que procesar para crear el nuevo Shapefile. El programa actual no se parece en nada al que había en el caso de la primera malla de 200 puntos, porque cuando intentaba procesar los datos de los 11.000 puntos el proceso tardaba casi 8 horas en ejecutarse, algo totalmente inviable ya que los datos se envían cada 6 horas. He tenido que eliminar muchos chequeos previos, que tenían para ver si los datos seguían una estructura, y asumir que siempre van a seguir la misma estructura sin comprobar nada, además de tener que limitar el espacio que ocupan los datos en el .dbf, como ya he mencionado en el apartado 3.6.4.

Por lo que, teniendo en cuenta lo anterior, he llegado a aprender a trabajar y modificar un proyecto en fechas críticas, además de intentar optimizar al máximo por si ocurren errores inesperados como el de la ejecución de más de 8 horas.

# 8.- CONCLUSIONES

Las conclusiones que a las que he llegado como resultado del desarrollo de este trabajo son las siguientes:

- He aprendido a ser autosuficiente, tratando de resolver por mí mismo mis propios problemas, ya que yo era el experto en el trabajo que me habían encomendado.
- He aprendido a colaborar con otros profesionales, planteando el trabajo, de forma que pudiese luego ser publicado por los desarrolladores de la página web en la cual se va a subir el visor.
- He aprendido a utilizar muchas herramientas nuevas, como ejecutar proyectos de QGIS, generar imágenes, operaciones con archivos, transferencia de archivos, compresión de archivos, cambiar el contenido de las capas publicadas por geoserver, etc.
- En general el uso de estas nuevas herramientas no es sencillo, pero con una buena base cartográfica y de programación, y, gracias a la documentación que se puede encontrar en internet, es posible obtener soluciones con un esfuerzo moderado.

He tenido que modificar el proyecto para adaptarlo a los nuevos datos de entrada en muy poco tiempo. Esto me ha hecho aprender a optimizar el tiempo y a trabajar bajo presión.

# 9. REFERENCIAS

## 9.1.- Referencias Conceptos

AEMET, 2018. *AEMET Temperatura.* [En línea]
Available at:
http://www.aemet.es/es/eltiempo/prediccion/modelosnumericos/harmonie_arome?opc2=pybal&opc3=tm
[Último acceso: 10 12 2017].

AEMET, 2018. *AEMET Viento.* [En línea]
Available at:
http://www.aemet.es/es/eltiempo/prediccion/modelosnumericos/harmonie_arome?opc2=pybal&opc3=vi
[Último acceso: 12 12 2017].

Hooper, D., 2002. *NERC MST.* [En línea]
Available at: http://mst.nerc.ac.uk/wind_vect_convs.html
[Último acceso: 22 3 2018].

Intellicast, 2018. *Intellicast.* [En línea]
Available at: http://www.intellicast.com/National/Humidity/Current.aspx
[Último acceso: 10 12 2017].

## 9.2.- Referencias Código

Cómo ejemplos de código o preguntas relacionadas con algún problema que se haya tenido con ello no cuenta como una referencia válida según los estándares no se ha incluido éstas a la hora de escribir la memoria. Pero como se ha querido dar crédito a los autores se incluye este apartado al final del trabajo.

### 9.2.1.- Python


- Lista de todos los archivos en un directorio

  StackOverflow. [En línea]
  Available at:
  https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory
  [Último acceso: 12 12 2017].

- Terminando la ejecución de un script en Python

  StackOverflow. [En línea]
  Available at:
  https://stackoverflow.com/questions/73663/terminating-a-python-script
  [Último acceso: 12 12 2017].

- Ejecutar un comando de SHELL con Python

  StackExchange. [En línea]
  Available at:
  https://unix.stackexchange.com/questions/238180/execute-shell-commands-in-python

  [Último acceso: 12 12 2017].

- Crear un .zip

  StackOverflow. [En línea]
  Available at:
  https://stackoverflow.com/questions/14568647/create-zip-in-python
  [Último acceso: 12 12 2017].

- Cargar un .csv

  StackOverflow. [En línea]
  Available at:
  https://stackoverflow.com/questions/3518778/how-to-read-csv-into-record-array-in-numpy
  [Último acceso: 12 12 2017].

### 9.2.2.- QGIS

- Mostrar/Ocultar las capas en QGIS

  StackExchange. [En línea]
  Available at:
  https://gis.stackexchange.com/questions/42986/how-to-hide-show-qgisvectorlayer-from-python-code
  [Último acceso: 12 12 2017].

- Exportar una imagen con plantilla

  StackExchange. [En línea]
  Available at:
  https://gis.stackexchange.com/questions/144792/save-print-map-qgis-composer-view-as-png-pdf-using-python-without-changing-anyt
  [Último acceso: 12 12 2017].

StackExchange. [En línea]
Available at:
https://gis.stackexchange.com/questions/77848/programmatically-load-composer-from-template-and-generate-atlas-using-pyqgis
[Último acceso: 12 12 2017].

- Crear una capa vacía

StackExchange. [En línea]
Available at:
https://gis.stackexchange.com/questions/30261/how-to-create-a-new-empty-vector-layer-programmatically
[Último acceso: 12 12 2017].

StackExchange. [En línea]
Available at:
https://gis.stackexchange.com/questions/127749/exporting-layer-to-shapefile-using-pyqgis
[Último acceso: 12 12 2017].

- Crear y guardar atributos en un Shapefile

StackExchange. [En línea]
Available at:
https://gis.stackexchange.com/questions/174971/how-to-define-the-number-of-decimals-when-adding-a-new-field-as-double-to-an-att
[Último acceso: 12 12 2017].

- Crear y guardar atributos en un Shapefile

StackExchange. [En línea]
Available at:
https://gis.stackexchange.com/questions/83641/setting-feature-attribute-by-name-via-qgis-python-api
[Último acceso: 12 12 2017].

### 9.2.3.- Geoserver REST, CURL

- Borrado de una capa

  StackExchange. [En línea]
  Available at:
  https://gis.stackexchange.com/questions/118906/how-to-delete-a-geoserver-coveragestore-using-rest-api

  [Último acceso: 12 12 2017].

- Subir un Shapefile

  StackOverflow. [En línea]
  Available at:
  https://stackoverflow.com/questions/1242797/rest-uploading-a-shapefile

  [Último acceso: 12 12 2017].

- Subir una imagen

  StackExchange. [En línea]
  Available at:
  https://gis.stackexchange.com/questions/237135/creating-image-pyramid-datastore-using-geoserver-rest-api

  [Último acceso: 12 12 2017].

### 9.2.3.- JavaScript


- Definición de una nueva proyección

  Proj4js. [En línea]

  Available at:
  http://proj4js.org/

  [Último acceso: 12 12 2017].

- Conexión WFS JSONP a la capa

  StackExchange. [En línea]

  Available at:
  https://gis.stackexchange.com/questions/34892/how-to-get-json-from-geoserver-using-ajax-request/34907

  [Último acceso: 12 12 2017].

- Modificación dinámica del estilo de la capa de Flechas

  Anitagraser. [En línea]

  Available at:
  https://anitagraser.com/2010/06/10/dynamic-styling-and-filtering-of-a-geoserver-wms-using-openlayers-layer-wms-post/

  [Último acceso: 12 12 2017].

- Detección de la carga de las capas

  StackOverflow. [En línea]

  Available at:
  https://stackoverflow.com/questions/34311461/openlayers-how-to-detect-the-map-view-is-completely-loaded

  [Último acceso: 12 12 2017].

# 10.- ANEXOS

En esta parte se expondrán los códigos enteros, con la carga de librerías, directorios etc…

## 10.1.- Códigos en Python

### 10.1.1.- Comprobar_archivos.py

```
import os
import os.path
import shutil
import time
import datetime
import sys
from PIL import Image
from os.path import isfile, join
import zipfile
import creacion_grib2
import convertir_zip
import execute_qgis4_png_VD
import execute_qgis4_png_VE
import execute_qgis4_png_HR
import execute_qgis4_png_TE
import execute_qgis4_shp
import geoserver_cURL

now = datetime.datetime.now()
year=now.year
month= now.month
day= now.day
hour= now.hour
minute=now.minute
hour=hour+minute/60.0


#Aqui defino en cual de las 4 epocas empieza a ejecutarse el Script porque los
    datos llegan 4 veces al dia
if hour>=4 and hour<9:
    t_ep= "00"
if hour>=10 and hour<15:
    t_ep= "06"
if hour>=16 and hour<21:
    t_ep= "12"
if (hour>=22 and hour<24) or (hour>=0 and hour<3):
    t_ep= "18"
print t_ep

#Defino todos los paths con los que vaya a trabajar
#Lo obtengo como el path a los scripts eliminando este ultimo
path_general=str((os.path.dirname(os.path.realpath(__file__)))[:-8])

path_grib1=r"C:\Datos\temasims\todo\rayos_aemet"
path_grib2=path_general+r"\datos\grib2"
path_datos=path_general+"\datos"
path_hist=path_general+r"\historico"
```

```
47. dir_destino_g1=path_general+r"\datos\grib1"
48. 
49. path_qgis_capas=path_general+r"\qgis"
50. path_shp_ini=path_general+r"\datos\shp"
51. path_shp_dest=path_general+r"\shp_web"
52. 
53. path_fotos_VD=path_general+r"\datos\fotos_png_VD"
54. path_fotos_VE=path_general+r'\datos\fotos_png_VE'
55. path_fotos_HR=path_general+r"\datos\fotos_png_HR"
56. path_fotos_TE=path_general+r"\datos\fotos_png_TE"
57. 
58. #fotos_web-- ve,hr,te
59. 
60. path_fotos_web_VD=r"C:\inetpub\prevencion2\Documents\previsiones\fotos_web_VD"

61. path_fotos_web_VE=path_general+r"\fotos_web_VE"
62. path_fotos_web_HR=r"C:\inetpub\prevencion2\Documents\previsiones\fotos_web_HR"

63. path_fotos_web_TE=r"C:\inetpub\prevencion2\Documents\previsiones\fotos_web_TE"

64. 
65. 
66. path_miniatura_VD=path_general+r"\miniatura_VD"
67. path_miniatura_VE=path_general+r"\miniatura_VE"
68. path_miniatura_HR=path_general+r"\miniatura_HR"
69. path_miniatura_TE=path_general+r"\miniatura_TE"
70. 
71. path_geoserver=r'\\arcgis.vaersa.org\datos_visor_previsiones'
72. 
73. f = open(path_hist+r'\status.txt','a')
74. 
75. #               ------------------------------------------------------------
    ------------------------------------------------
76. #                   ---------------------------
    BORRADO DE TODO LOS DATOS LOCALES ANTES DE EJECUTAR------------------------
77. 
78. #Lo primero que hace ele programa es limpiar las carpetas temporales por si ac
    aso, aunque estas se hayan limpiado anteriormente
79. for fil in os.listdir(dir_destino_g1):
80.     os.remove(dir_destino_g1+"\\"+fil)
81. for fil in os.listdir(path_grib2):
82.     os.remove(path_grib2+"\\"+fil)
83. for fil in os.listdir(path_fotos_VD):
84.     os.remove(path_fotos_VD+"\\"+fil)
85. for fil in os.listdir(path_fotos_VE):
86.     os.remove(path_fotos_VE+"\\"+fil)
87. for fil in os.listdir(path_fotos_TE):
88.     os.remove(path_fotos_TE+"\\"+fil)
89. for fil in os.listdir(path_fotos_HR):
90.     os.remove(path_fotos_HR+"\\"+fil)
91. for fil in os.listdir(path_shp_ini):
92.     os.remove(path_shp_ini+"\\"+fil)
93. #               ------------------------------------------------------------
    ------------------------------------------------
94. #                   ------------------------------------ESCUCHA DE LOS DATOS-
    ------------------------------------------
95. 
96. cantidad=0
97. while cantidad<49:
98.     #Cada bucle que entra determina la fecha con las horas y minutos
99.     now = datetime.datetime.now()
100.          year=str(now.year)
101.          month= str(now.month)
```

```
102.            day= str(now.day)
103.            hour= now.hour
104.            minute=now.minute
105.            hour=hour+minute/60.0
106.
107.            #Para poder igualar o nombrar archivos necesito que los meses y dia
    s siempre tengan dos digitos
108.            if len(day)==1:
109.                day="0"+day
110.            if len(month)==1:
111.                month="0"+month
112.            onlyfiles=[]
113.            for fil in os.listdir(path_grib1):
114.                #Si el grib tiene FC y M en el nombre entonces lo cogera
115.                if "fc" and 'm' in fil:
116.                    resto=fil[2:-8]
117.                    #Coger solo archivos de la epoca de ejecuccion, aunque haya
     muchos grib1 esto seleccionara solo los que entren dentro del tiempo de ejecu
    cion
118.                    if resto==(year+month+day+t_ep):
119.                        onlyfiles.append(fil)
120.            cantidad=len(onlyfiles)
121.
122.            #Si el programa se excede del tiempo limite esto lo cortara y impri
    mira un texto en el archivo de status
123.            if t_ep=="00" and hour>9:
124.                f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha p
    arado porque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"
    \n")
125.                f.close()
126.                sys.exit()
127.            if t_ep=="06" and hour>15:
128.                f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha p
    arado porque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"
    \n")
129.                f.close()
130.                sys.exit()
131.            if t_ep=="12" and hour>21:
132.                f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha p
    arado porque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"
    \n")
133.                f.close()
134.                sys.exit()
135.            #Aqui hay que ponerle un doble condicionante porque si solo dejo ma
    yor a 3 si entra a las 23 el programa se cerrara
136.            #Es por ello que la hora para que se cierre debe estar comprendida
    entre las 3 y las 22.
137.            if t_ep=="18" and hour>3 and hour<22 :
138.                f.write("Log del: "+year+month+day+t_ep+' : El servicio se ha p
    arado porque no ha encontrado datos. Numero actual de datos: '+str(cantidad)+"
    \n")
139.                f.close()
140.                sys.exit()
141.
142.        f.write("Log del: "+year+month+day+t_ep+' : Los archivos fueron encontr
    ados'+"\n")
143.        f.flush()
144.
145.        #               ------------------------------------------------------
    ------------------------------------------------------
146.        #                   ------------------------------------MOVER GRIB 1--
    ------------------------------------------------
147.
```

```
148.        f.write('                    : Moviendo los datos...'+"\n")
149.        f.flush()
150.        #Muevo los archivos del directorio general a mi directorio personalizad
    o
151.        try:
152.            for fil in onlyfiles:
153.                shutil.move(path_grib1+"\\"+fil, dir_destino_g1+"\\"+fil)
154.            f.write('                    : Los datos de GRIB1 fueron movidos'+"
    \n")
155.            f.flush()
156.        except Exception as e:
157.            f.write('     ERROR          : Un error ha ocurrido moviendo los da
    tos de GRIB1: '+str(e)+"\n")
158.            f.close()
159.            sys.exit()
160.        #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    !!!!!!!!! Lee el primer archivo pasado
161.        resto=os.listdir(dir_destino_g1)[0][2:-8]
162.        #               ------------------------------------------------------
    ----------------------------------------------------
163.        #                   ------------------------------------CREAR GRIB 2--
    ------------------------------------------------
164.
165.        #creando los grib2
166.        f.write('                    : Convirtiendo a GRIB2...'+"\n")
167.        f.flush()
168.        try:
169.            creacion_grib2.g1_2_g2(onlyfiles,dir_destino_g1,path_grib2,path_gen
    eral)
170.            f.write('                    : Los GRIB2 se han creado satisfactori
    amente'+"\n")
171.            f.flush()
172.        except Exception as e:
173.            f.write('     ERROR          : Ha ocurrdio un error en creacion_gri
    b_2 script: '+str(e)+"\n")
174.            f.close()
175.            sys.exit()
176.
177.        #               ------------------------------------------------------
    ----------------------------------------------------
178.        #                   ------------------------------------
    HISTORICO GRIB1 Y GRIB2----------------------------------------
179.
180.        # creando los zip historicos de grib1 y grib2 dentro de la carpeta de h
    istorico
181.        f.write('                    : Creando el .zip del GRIB1...'+"\n")
182.        f.flush()
183.        try:
184.            folder="grib1"
185.            convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
186.            f.write('                    : Los datos de GRIB1 se han guardado e
    n el historico'+"\n")
187.            f.flush()
188.        except Exception as e:
189.            f.write('     ERROR          : Un error ha ocurrido creando el .zip
     de GRIB1: '+str(e)+"\n")
190.
191.        f.write('                    : Creando el historico del GRIB2...'+"\n")

192.        f.flush()
193.        try:
194.            folder="grib2"
```

```
195.            shutil.copytree(path_datos+"\\"+folder,path_hist+"\\"+resto+"_grib2
    ")
196.            f.write('                    : Los datos de GRIB2 se han guardado e
    n el historico'+"\n")
197.            f.flush()
198.        except Exception as e:
199.            f.write('     ERROR          : Un error ha ocurrido creando el hist
    orico de GRIB2: '+str(e)+"\n")
200.
201.
202.        #Elimina todas las fotos del directorio temporal de las fotos para que
    solo se queden las temporales
203.        for fil in os.listdir(path_fotos_VD):
204.                os.remove(path_fotos_VD+"\\"+fil)
205.
206.        #               ------------------------------------------------------
    ------------------------------------------------------
207.        #                   ------------------------------------
    QGIS PNG CON FLECHAS GEOREFERENCIADO--------------------------
208.
209.        #Ejecutando el QGIS, este proceso tarda 210 segundos en ejecutarse por
    o que si se queire hacer pruebas con el codigo se aconseja comentarlo.
210.        f.write('                    : Ejecutando el QGIS para la creacion de l
    as imagenes de viento con flechas...'+"\n")
211.        f.flush()
212.        try:
213.            execute_qgis4_png_VD.execute_qgis(path_general)
214.            f.write('                    : Se han creado las imagenes '+"\n")
215.            f.flush()
216.            d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
    8]), int(t_ep))
217.            f.write('                    : Renombrando y guarando las fotos...'
    +"\n")
218.            f.flush()
219.            for fil in os.listdir(path_fotos_VD):
220.                #Comprueba si el mes,dia y hora son de un digito
221.                if len(str(d.day))==1:
222.                    day='0'+str(d.day)
223.                else:
224.                    day=str(d.day)
225.                if len(str(d.month))==1:
226.                    month='0'+str(d.month)
227.                else:
228.                    month=str(d.month)
229.
230.                if len(str(d.hour))==1:
231.                    hour='0'+str(d.hour)
232.                else:
233.                    hour=str(d.hour)
234.                os.rename(path_fotos_VD+"\\"+fil,path_fotos_VD+"\\"+"VD_"+str(d
    .year)+r"_"+month+r"_"+day+r"_"+hour+r".png")
235.                #De paso copio y pego el archivo de georreferenciacion y le cam
    bio el nombre para que coincida con el del png, lo mismo con el prj
236.                shutil.copyfile(path_qgis_capas+r'\plantilla_georeferenciacion.
    pngw', path_fotos_VD+"\\"+"VD_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".p
    gw")
237.                shutil.copyfile(path_qgis_capas+r'\plantilla_proyeccion.prj', p
    ath_fotos_VD+"\\"+"VD_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".prj")
238.                d+=datetime.timedelta(hours=1)
239.            #Guardando las fotos en el historico
240.            folder="fotos_png_VD"
241.            convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
```

```
242.           f.write('                    : Las fotos fueron renombrados y guard
    ados'+"\n")
243.           f.flush()
244.           #Muevo las imagenes al directorio de la web
245.           f.write('                    : Moviendo las fotos...'+"\n")
246.           f.flush()
247.           for fil in os.listdir(path_fotos_VD):
248.               if "output" not in fil:
249.                shutil.move(path_fotos_VD+"\\"+fil,path_fotos_web_VD+"\\"+fil)

250.           f.write('                    : Las fotos fueron movidos al director
    io '+path_fotos_web_VD+"\n")
251.           f.flush()
252.           f.write('                    : Borrando las fotos antiguas...'+"\n"
    )
253.           f.flush()
254.           d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
    8]), int(t_ep))
255.           for fil in os.listdir(path_fotos_web_VD):
256.               if (".png" in fil) or (".pgw" in fil) or (".prj" in fil):
257.                   if "output" not in fil:
258.                       d_img=datetime.datetime(int(fil.split("_")[1]), int(fil
    .split("_")[2]), int(fil.split("_")[3]),int(fil.split("_")[4][:2]))
259.                       if d_img<d:
260.                           os.remove(path_fotos_web_VD+"\\"+fil)
261.                   else:
262.                       f.write('                    : No le ha dado tiempo al'
    +resto+".....
    .....\n")
263.                       f.flush()
264.           f.write('                    : Ha acabado de hacer las fotos de los
     Vientos con flechas'+"\n")
265.           f.flush()
266.           f.write('                    : Creando las miniaturas del viento co
    n flechas'+"\n")
267.           f.flush()
268.           size = 500, 300
269.           for fil in os.listdir(path_miniatura_VD):
270.               os.remove(path_miniatura_VD+'\\'+fil)
271.           for fil in os.listdir(path_fotos_web_VD):
272.               if '.png' in fil:
273.                   im = Image.open(path_fotos_web_VD+'\\'+fil)
274.                   im.thumbnail(size, Image.ANTIALIAS)
275.                   im.save(path_miniatura_VD+'\\'+fil, "PNG",dpi=(96,96))
276.           f.write('                    : Ha acabado de crear las miniaturas'+
    "\n")
277.           f.flush()
278.       except Exception as e:
279.           f.write('     ERROR          : Un error ha ocurrido creando las ima
    genes de Vientos con flechas: '+str(e)+"\n")
280.
281.       #               ----------------------------------------------------
    ----------------------------------------------------
282.       #                   ------------------------------------
    QGIS PNG DE VIENTOS GEOREFERENCIADO-----------------------------
283.
284.
285.
286.       f.write('                    : Ejecutando el QGIS para la creacion de l
    as imagenes georeferenciadas...'+"\n")
287.       f.flush()
288.       try:
289.           execute_qgis4_png_VE.execute_qgis(path_general)
```

```
290.            f.write('                       : Las imagenes georeferenciadas fueron
    creadas'+"\n")
291.            f.flush()
292.            f.write('                       : Renombrando y guarando las fotos geo
    referenciadas...'+"\n")
293.            f.flush()
294.            d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
    8]), int(t_ep))
295.            for fil in os.listdir(path_fotos_VE):
296.                if len(str(d.day))==1:
297.                    day='0'+str(d.day)
298.                else:
299.                    day=str(d.day)
300.
301.                if len(str(d.month))==1:
302.                    month='0'+str(d.month)
303.                else:
304.                    month=str(d.month)
305.
306.                if len(str(d.hour))==1:
307.                    hour='0'+str(d.hour)
308.                else:
309.                    hour=str(d.hour)
310.                os.rename(path_fotos_VE+"\\"+fil,path_fotos_VE+"\\"+"VE_"+str(d
    .year)+r"_"+month+r"_"+day+r"_"+hour+r".png")
311.                #De paso copio y pego el archivo de georreferenciacion y le cam
    bio el nombre para que coincida con el del png, lo mismo con el prj
312.                shutil.copyfile(path_qgis_capas+r'\plantilla_georeferenciacion.
    pngw', path_fotos_VE+"\\"+"VE_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".p
    gw")
313.                shutil.copyfile(path_qgis_capas+r'\plantilla_proyeccion.prj', p
    ath_fotos_VE+"\\"+"VE_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".prj")
314.                d+=datetime.timedelta(hours=1)
315.            #Guardando las fotos en el historico
316.            folder="fotos_png_VE"
317.            convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
318.            f.write('                       : Se han renombrado y guardado las ima
    genes georeferenciadas'+"\n")
319.            f.flush()
320.            #Muevo las imagenes al directorio de la web
321.            f.write('                       : Moviendo las fotos georeferenciadas.
    ..'+"\n")
322.            f.flush()
323.            for fil in os.listdir(path_fotos_VE):
324.                shutil.move(path_fotos_VE+"\\"+fil, path_fotos_web_VE+"\\"+fil)

325.            f.write('                       : Se han movido las fotos al directori
    o '+path_fotos_web_VE+"\n")
326.            f.flush()
327.            f.write('                       : Borrando las fotos antiguas...'+"\n"
    )
328.            f.flush()
329.            d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
    8]), int(t_ep))
330.            for fil in os.listdir(path_fotos_web_VE):
331.                if (".png" in fil) or (".pgw" in fil) or (".prj" in fil):
332.                    if "output" not in fil:
333.                        d_img=datetime.datetime(int(fil.split("_")[1]), int(fil
    .split("_")[2]), int(fil.split("_")[3]),int(fil.split("_")[4][:2]))
334.                        if d_img<d:
335.                            os.remove(path_fotos_web_VE+"\\"+fil)
336.                    else:
```

```
                        f.write('                    : No le ha dado tiempo al'
    +resto+".....\n")
                        f.flush()
            f.write('                    : Las imagenes de los vientos se han c
    reado correctamente'+"\n")
            f.flush()
            f.write('                    : Creando las miniaturas del viento'+"
    \n")
            f.flush()
            size = 500, 300
            for fil in os.listdir(path_miniatura_VE):
                os.remove(path_miniatura_VE+'\\'+fil)
            for fil in os.listdir(path_fotos_web_VE):
                if '.png' in fil:
                    im = Image.open(path_fotos_web_VE+'\\'+fil)
                    im.thumbnail(size, Image.ANTIALIAS)
                    im.save(path_miniatura_VE+'\\'+fil, "PNG",dpi=(96,96))
            f.write('                    : Ha acabado de crear las miniaturas'+
    "\n")
            f.flush()
        except Exception as e:
            f.write('     ERROR          : Un error ha ocurrido creando las ima
    genes de los vientos: '+str(e)+"\n")


        #               ------------------------------------------------------
    ------------------------------------------------------
        #                   -------------------------------------
    QGIS PNG DE HUMEDADES GEOREFERENCIADO-----------------------------------------
    -


        f.write('                    : Ejecutando el QGIS para la creacion de l
    as imagenes georeferenciadas de humedades...'+"\n")
        f.flush()
        try:
            execute_qgis4_png_HR.execute_qgis(path_general)
            f.write('                    : Las imagenes georeferenciadas de hum
    edades fueron creadas'+"\n")
            f.flush()
            f.write('                    : Renombrando y guarando las fotos geo
    referenciadas de humedades...'+"\n")
            f.flush()
            d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
    8]), int(t_ep))
            for fil in os.listdir(path_fotos_HR):
                if len(str(d.day))==1:
                    day='0'+str(d.day)
                else:
                    day=str(d.day)

                if len(str(d.month))==1:
                    month='0'+str(d.month)
                else:
                    month=str(d.month)

                if len(str(d.hour))==1:
                    hour='0'+str(d.hour)
                else:
                    hour=str(d.hour)
```

```
386.                os.rename(path_fotos_HR+"\\"+fil,path_fotos_HR+"\\"+"HR_"+str(d
    .year)+r"_"+month+r"_"+day+r"_"+hour+r".png")
387.                #De paso copio y pego el archivo de georreferenciacion y le cam
    bio el nombre para que coincida con el del png, lo mismo con el prj
388.                shutil.copyfile(path_qgis_capas+r'\plantilla_georeferenciacion.
    pngw', path_fotos_HR+"\\"+"HR_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".p
    gw")
389.                shutil.copyfile(path_qgis_capas+r'\plantilla_proyeccion.prj', p
    ath_fotos_HR+"\\"+"HR_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".prj")
390.                d+=datetime.timedelta(hours=1)
391.            #Guardando las fotos en el historico
392.            folder="fotos_png_HR"
393.            convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
394.            f.write('                    : Se han renombrado y guardado las ima
    genes georeferenciadas de humedades'+"\n")
395.            f.flush()
396.            #Muevo las imagenes al directorio de la web
397.            f.write('                    : Moviendo las fotos georeferenciadas.
    ..'+"\n")
398.            f.flush()
399.            for fil in os.listdir(path_fotos_HR):
400.                shutil.move(path_fotos_HR+"\\"+fil, path_fotos_web_HR+"\\"+fil)

401.            f.write('                    : Se han movido las fotos al directori
    o '+path_fotos_web_HR+"\n")
402.            f.flush()
403.            f.write('                    : Borrando las fotos antiguas de humed
    ades...'+"\n")
404.            f.flush()
405.            d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
    8]), int(t_ep))
406.            for fil in os.listdir(path_fotos_web_HR):
407.                if (".png" in fil) or (".pgw" in fil) or (".prj" in fil):
408.                    if "output" not in fil:
409.                        d_img=datetime.datetime(int(fil.split("_")[1]), int(fil
    .split("_")[2]), int(fil.split("_")[3]),int(fil.split("_")[4][:2]))
410.                        if d_img<d:
411.                            os.remove(path_fotos_web_HR+"\\"+fil)
412.                    else:
413.                        f.write('                    : No le ha dado tiempo al'
    +resto+".....
    .....\n")
414.                        f.flush()
415.            f.write('                    : Las imagenes de humedades fueron cre
    adas de forma correcta'+"\n")
416.            f.flush()
417.            f.write('                    : Creando las miniaturas de humedades'
    +"\n")
418.            f.flush()
419.            size = 500, 300
420.            for fil in os.listdir(path_miniatura_HR):
421.                os.remove(path_miniatura_HR+'\\'+fil)
422.            for fil in os.listdir(path_fotos_web_HR):
423.                if '.png' in fil:
424.                    im = Image.open(path_fotos_web_HR+'\\'+fil)
425.                    im.thumbnail(size, Image.ANTIALIAS)
426.                    im.save(path_miniatura_HR+'\\'+fil, "PNG",dpi=(96,96))
427.            f.write('                    : Ha acabado de crear las miniaturas'+
    "\n")
428.            f.flush()
429.        except Exception as e:
430.            f.write('     ERROR          : Un error ha ocurrido creando las ima
    genes de las humedades: '+str(e)+"\n")
```

```
431.
432.       #               ------------------------------------------------------
   ------------------------------------------------------
433.       #                    -------------------------------------
   QGIS PNG DE TEMPERATURAS GEOREFERENCIADO-----------------------
434.
435.
436.       f.write('                    : Ejecutando el QGIS para la creacion de l
   as imagenes georeferenciadas de temperaturas...'+"\n")
437.       f.flush()
438.       try:
439.           execute_qgis4_png_TE.execute_qgis(path_general)
440.           f.write('                    : Las imagenes georeferenciadas de tem
   peraturas fueron creadas'+"\n")
441.           f.flush()
442.           f.write('                    : Renombrando y guarando las fotos geo
   referenciadas de temperaturas...'+"\n")
443.           f.flush()
444.           d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
   8]), int(t_ep))
445.           for fil in os.listdir(path_fotos_TE):
446.               if len(str(d.day))==1:
447.                   day='0'+str(d.day)
448.               else:
449.                   day=str(d.day)
450.
451.               if len(str(d.month))==1:
452.                   month='0'+str(d.month)
453.               else:
454.                   month=str(d.month)
455.
456.               if len(str(d.hour))==1:
457.                   hour='0'+str(d.hour)
458.               else:
459.                   hour=str(d.hour)
460.               os.rename(path_fotos_TE+"\\"+fil,path_fotos_TE+"\\"+"TE_"+str(d
   .year)+r"_"+month+r"_"+day+r"_"+hour+r".png")
461.               #De paso copio y pego el archivo de georreferenciacion y le cam
   bio el nombre para que coincida con el del png, lo mismo con el prj
462.               shutil.copyfile(path_qgis_capas+r'\plantilla_georeferenciacion.
   pngw', path_fotos_TE+"\\"+"TE_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".p
   gw")
463.               shutil.copyfile(path_qgis_capas+r'\plantilla_proyeccion.prj', p
   ath_fotos_TE+"\\"+"TE_"+str(d.year)+r"_"+month+r"_"+day+r"_"+hour+r".prj")
464.               d+=datetime.timedelta(hours=1)
465.           #Guardando las fotos en el historico
466.           folder="fotos_png_TE"
467.           convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
468.           f.write('                    : Se han renombrado y guardado las ima
   genes georeferenciadas de temperaturas'+"\n")
469.           f.flush()
470.           #Muevo las imagenes al directorio de la web
471.           f.write('                    : Moviendo las fotos georeferenciadas.
   ..'+"\n")
472.           f.flush()
473.           for fil in os.listdir(path_fotos_TE):
474.               shutil.move(path_fotos_TE+"\\"+fil, path_fotos_web_TE+"\\"+fil)

475.           f.write('                    : Se han movido las fotos al directori
   o '+path_fotos_web_TE+"\n")
476.           f.flush()
477.           f.write('                    : Borrando las fotos antiguas de tempe
   raturas...'+"\n")
```

```
478.           f.flush()
479.           d = datetime.datetime(int(resto[:4]), int(resto[4:6]), int(resto[6:
    8]), int(t_ep))
480.           for fil in os.listdir(path_fotos_web_TE):
481.               if (".png" in fil) or (".pgw" in fil) or (".prj" in fil):
482.                   if "output" not in fil:
483.                       d_img=datetime.datetime(int(fil.split("_")[1]), int(fil
    .split("_")[2]), int(fil.split("_")[3]),int(fil.split("_")[4][:2]))
484.                       if d_img<d:
485.                           os.remove(path_fotos_web_TE+"\\"+fil)
486.                   else:
487.                       f.write('                    : No le ha dado tiempo al'
    +resto+".....
    .....\n")
488.                       f.flush()
489.           f.write('                    : Las fotos de las temperaturas se han
     creado de forma correcta'+"\n")
490.           f.flush()
491.           f.write('                    : Creando las miniaturas de temperatur
    a'+"\n")
492.           f.flush()
493.           size = 500, 300
494.           for fil in os.listdir(path_miniatura_TE):
495.               os.remove(path_miniatura_TE+'\\'+fil)
496.           for fil in os.listdir(path_fotos_web_TE):
497.               if '.png' in fil:
498.                   im = Image.open(path_fotos_web_TE+'\\'+fil)
499.                   im.thumbnail(size, Image.ANTIALIAS)
500.                   im.save(path_miniatura_TE+'\\'+fil, "PNG",dpi=(96,96))
501.           f.write('                    : Ha acabado de crear las miniaturas'+
    "\n")
502.           f.flush()
503.       except Exception as e:
504.           f.write('     ERROR          : Un error ha ocurrido creando las ima
    genes de temperatura: '+str(e)+"\n")
505.
506.       #               ------------------------------------------------------
    ------------------------------------------------------
507.       #                   ------------------------------------
    HORIZONTE FOTOS-------------------------------------------------
508.
509.       #Crear un log para saber la fecha de la epoca de la 1a epoca en la cual
     estan hechas las predicciones
510.       #COPIAR EL HORIZONTE.TXT DONDE SE EXPORTEN LOS DATOS
511.       print resto
512.       try:
513.           f.write('                    : Creando los archivos de horizonte'+"
    \n")
514.           f.flush()
515.           hori_txt = open(path_fotos_web_VD+r'\horizonte.txt','w')
516.           hori_txt.write(resto[:-2]+r"_"+resto[-2:])
517.           hori_txt.close()
518.           shutil.copyfile(path_fotos_web_VD+r'\horizonte.txt', path_fotos_web
    _VE+r'\horizonte.txt')
519.           shutil.copyfile(path_fotos_web_VD+r'\horizonte.txt', path_fotos_web
    _HR+r'\horizonte.txt')
520.           shutil.copyfile(path_fotos_web_VD+r'\horizonte.txt', path_fotos_web
    _TE+r'\horizonte.txt')
521.       except Exception as e:
522.           f.write('     ERROR          : Un error ha ocurrido creando los hor
    izontes: '+str(e)+"\n")
523.
524.
```

```
525.        #               ------------------------------------------------------
    ------------------------------------------------------
526.        #                   ------------------------------------QGIS SHP -----
    ------------------------------------------
527.
528.        #Ejecutando el QGIS, este proceso tarda 210 segundos en ejecutarse por
    o que si se queire hacer pruebas con el codigo se aconseja comentarlo.
529.        f.write('                    : Ejecutando el QGIS para la creacion de l
    os .SHP...'+"\n")
530.        f.flush()
531.        try:
532.            execute_qgis4_shp.execute_qgis_shp(path_general)
533.            f.write('                    : Se ha creado el .SHP con todas las h
    oras'+"\n")
534.            f.flush()
535.            f.write('                    : Renombrando, moviendo y guardando el
     .SHP ...'+"\n")
536.            f.flush()
537.            #Renombrar los archivos .shp y eliminar los que no son los de la pr
    opia capa
538.            for fil in os.listdir(path_shp_ini):
539.                if r"csv_" not in fil:
540.                    os.rename(path_shp_ini+"\\"+fil, path_shp_ini+"\\"+resto+fi
    l[-4:])
541.                else:
542.                    os.remove(path_shp_ini+"\\"+fil)
543.            #Compromir en .ZIP las capas
544.            folder="shp"
545.            convertir_zip.convertir_zip(path_datos,path_hist,resto,folder)
546.            #Borrar los archivos de web
547.            for fil in os.listdir(path_shp_dest):
548.                os.remove(path_shp_dest+"\\"+fil)
549.            #Mover los .shp al directorio de la web
550.            for fil in os.listdir(path_shp_ini):
551.                shutil.copy2(path_shp_ini+"\\"+fil, path_shp_dest+"\\"+resto+fi
    l[-4:])
552.            #Creacion del .zip para el geoserver dentro de la carpeta shp_web
553.            folder="shp"
554.            zip_name='shp_cmp'
555.            for fil in os.listdir(path_shp_ini):
556.                if r"csv_" not in fil:
557.                    os.rename(path_shp_ini+"\\"+fil, path_shp_ini+"\\shp_geoser
    ver"+fil[-4:])
558.            zipf = zipfile.ZipFile('{0}.zip'.format(os.path.join(path_shp_dest,
     zip_name)), 'w', zipfile.ZIP_DEFLATED)
559.            for root, dirs, files in os.walk(os.path.join(path_datos, folder)):

560.                for filename in files:
561.                    zipf.write(os.path.abspath(os.path.join(root, filename)), a
    rcname=filename)
562.            zipf.close()
563.            f.write('                    : Se ha creado el .SHP de forma correc
    ta'+"\n")
564.            f.flush()
565.        except Exception as e:
566.            f.write('     ERROR          : Un error ha ocurrido creando el .SHP
    : '+str(e)+"\n")
567.
568.        #               ------------------------------------------------------
    ------------------------------------------------------
569.        #                   ------------------------------------
    BORRADO DATOS LOCALES ------------------------------------------------------
570.
```

```
571.        f.write('                    : Borrando los datos locales...'+"\n")
572.        f.flush()
573.        #Borrado de los datos de sus carpetas
574.        for fil in os.listdir(dir_destino_g1):
575.            os.remove(dir_destino_g1+"\\"+fil)
576.        for fil in os.listdir(path_grib2):
577.            os.remove(path_grib2+"\\"+fil)
578.        for fil in os.listdir(path_fotos_VD):
579.            os.remove(path_fotos_VD+"\\"+fil)
580.        for fil in os.listdir(path_fotos_VE):
581.            os.remove(path_fotos_VE+"\\"+fil)
582.        for fil in os.listdir(path_fotos_TE):
583.            os.remove(path_fotos_TE+"\\"+fil)
584.        for fil in os.listdir(path_fotos_HR):
585.            os.remove(path_fotos_HR+"\\"+fil)
586.        for fil in os.listdir(path_shp_ini):
587.            os.remove(path_shp_ini+"\\"+fil)
588.        f.write('                    : Se han borrado los datos locales'+"\n")

589.        f.flush()
590.        f.write('                    : Subiendo los datos a la carpeta de Geose
    rver'+"\n")
591.        f.flush()
592.        try:
593.            for fil in os.listdir(path_geoserver+r'\png_VE'):
594.                os.remove(path_geoserver+r'\png_VE'+"\\"+fil)
595.            for fil in os.listdir(path_geoserver+r'\png_HR'):
596.                os.remove(path_geoserver+r'\png_HR'+"\\"+fil)
597.            for fil in os.listdir(path_geoserver+r'\png_TE'):
598.                os.remove(path_geoserver+r'\png_TE'+"\\"+fil)
599.            for fil in os.listdir(path_geoserver+r'\shp'):
600.                os.remove(path_geoserver+r'\shp'+"\\"+fil)
601.
602.            for fil in os.listdir(path_fotos_web_VE):
603.                if 'horizonte' not in fil:
604.                    shutil.copy2(path_fotos_web_VE+"\\"+fil, path_geoserver+r'\
    png_VE'+"\\"+fil)
605.            for fil in os.listdir(path_fotos_web_HR):
606.                if 'horizonte' not in fil:
607.                    shutil.copy2(path_fotos_web_HR+"\\"+fil, path_geoserver+r'\
    png_HR'+"\\"+fil)
608.            for fil in os.listdir(path_fotos_web_TE):
609.                if 'horizonte' not in fil:
610.                    shutil.copy2(path_fotos_web_TE+"\\"+fil, path_geoserver+r'\
    png_TE'+"\\"+fil)
611.            for fil in os.listdir(path_shp_dest):
612.                if '.zip' in fil:
613.                 shutil.copy2(path_shp_dest+"\\"+fil, path_geoserver+r'\shp'+"\
    \"+fil)
614.        except Exception as e:
615.            f.write('     ERROR          : Un error ha ocurrido subiendo los da
    tos a la carpeta de geoserver: '+str(e)+"\n")
616.
617.        f.write('                    : Ha empezado a subir archivos a gesoerver
    '+"\n")
618.        try:
619.            geoserver_cURL.geoserver_layers()
620.        except Exception as e:
621.            f.write('     ERROR          : Un error ha ocurrido subiendo los da
    tos a geoserver: '+str(e)+"\n")
622.        f.write('                    : Ha acabado a subir archivos a gesoerver'
    +"\n")
623.
```

```
624.       #copiando el archivo de horizonte de las fotos georeferenciadas
625.       f.write('                    : FINALIZADO'+"\n")
626.       f.flush()
627.       f.close()
```

## 10.1.2.- Convertir_zip.py

```
1.  import os
2.  import os.path
3.  from os.path import isfile, join
4.  import zipfile
5.
6.  def convertir_zip(path_datos,out_path,resto,folder):
7.      zipf = zipfile.ZipFile('{0}.zip'.format(os.path.join(out_path, resto+r"_"+
    folder)), 'w', zipfile.ZIP_DEFLATED)
8.      for root, dirs, files in os.walk(os.path.join(path_datos, folder)):
9.          for filename in files:
10.             zipf.write(os.path.abspath(os.path.join(root, filename)), arcname=
    filename)
11.     zipf.close()
```

## 10.1.3.- Creacion_grib2.py

```
1.  import os
2.  import time
3.  #Necesita el path general, pathgrib1, pathgrib2 y el array con los nombres de
    los archivos
4.
5.  def g1_2_g2(onlyfiles,path_grib1,path_grib2,path_general):
6.  ##    epoca=onlyfiles[0][10:-2]
7.      #Recorre una por una todos los grib1
8.      for ii in range(0,49):
9.          #Rescato lo que sera el nombre del archivo
10.         nom_fichero=onlyfiles[ii]
11.         #Rescato los ultimos dos digitos del nombre del grib1, esto hace refer
    encia al numero de la imagen
12.         n_img=nom_fichero[14:-4]
13.         #Defino el path hacia el grib1 y el path del fichero de salida, junto
    con su nombre y extension .grb
14.         p_gb1_f=path_grib1+"\\"+nom_fichero
15.         p_gb2_f=path_grib2+"\\"+"00"+str(n_img)+".grb"
16.         #Creo lo que es el comando que ira deentro de la llamada a la consola

17.         cmd_g1_2_g2= path_general+r"\programas\cdo\cdo setgridtype,lonlat "+ p
    _gb1_f+ " "+p_gb2_f
18.         #Llamo a la consola con el mensaje anterior para crear el grib2
19.         os.system(cmd_g1_2_g2)
20.         #Le doy dos segundos al programa para crear cada grid
21.         time.sleep(2)
22.     time.sleep(10)
```

## 10.1.4.- Creacion_png_macro_VD.py

```
1.  from PyQt4.QtCore import *
2.  from PyQt4.QtGui import *
3.  from PyQt4.QtXml import *
4.  from qgis.core import *
5.  from qgis.gui import *
6.  from qgis.utils import iface
7.  from time import sleep
8.  import os
9.
10. path_general=str((os.path.dirname(os.path.realpath(__file__)))[:-8])
11.
12. def creacion_png():
13.     iface.mapCanvas().refresh()
14.     #Obtengo las capas
15.     layers =  iface.legendInterface().layers()
16.     legend = iface.legendInterface()
17.     #Pondo una imagen de mas para que el codigo peuda llegar al final, eso es
    porque el bucle va con una iteracion de retraso
18.
19.     for i in range(0,51):
20.         #Cargo una por una las capas a la vez que la anterior a la cargada
21.         layer=layers[i]
22.         layer_ante=layers[i-1]
23.         if layer.name()!="Zones_PREVIFOC_DS_ETRS89":
24.             print
25.             #Reproyecto las capas a ETRS89 porque si no lo hago salen desplaza
    dos en el mapa
26.             crs = layer.crs()
27.             crs.createFromId(4258)
28.             layer.setCrs(crs)
29.             #Empieza a desactivar todos los layers anteriores a excepcion de l
    a primera iteraci?n
30.             if i>=2:
31.                 legend.setLayerVisible(layer_ante, False)
32.                 iface.mapCanvas().refresh()
33.                 layer.triggerRepaint()
34.             #Activa el layer actual
35.             legend.setLayerVisible(layer, True)
36.             #Refresco el canvas
37.             iface.mapCanvas().refresh()
38.             layer.triggerRepaint()
39.             #Le doy un segundo para que el programa reaccione
40.             sleep(1)
41.             #Creo el composer, lo que hace que pueda imprimir el canvas
42.             canvas = iface.mapCanvas()
43.             myFile =path_general+r'\qgis\plantilla.qpt'
44.             myTemplateFile = file(myFile, 'rt')
45.             myTemplateContent = myTemplateFile.read()
46.             myTemplateFile.close()
47.             composerAsDocument = QDomDocument()
48.             composerAsDocument.setContent(myTemplateContent)
49.             composition = QgsComposition(canvas.mapSettings())
50.             composition.loadFromTemplate(composerAsDocument, {})
51.             image = composition.printPageAsRaster(0)
52.             #Empieza a guardar a partir de la segunda iteracion porque la prim
    era imagen esta vacia, es por eso que hay 74 capas cargadas.
53.             if i>=2:
```

```
                #Guardo la imagen
                if len(str(i-2))==1:
                    asw="0"+str(i-2)
                else:
                    asw=str(i-2)
                image.save(path_general+r'\datos\fotos_png_VD\output'+asw+'.png', "png")
    legend.setLayerVisible(layer, False)
    iface.mapCanvas().refresh()
    layer.triggerRepaint()
creacion_png()
#print("--- %s seconds ---" % (time.time() - start_time))
#Cierre del proyecto y de QGIS
iface.actionExit().trigger()
os.kill(os.getpid(), 9)
```

## 10.1.5.- Creacion_png_macro_VE.py

```
from PyQt4.QtCore import *
from PyQt4.QtGui import *
from PyQt4.QtXml import *
from qgis.core import *
from qgis.gui import *
from qgis.utils import iface
from time import sleep
import os

path_general=str((os.path.dirname(os.path.realpath(__file__)))[:-8])

def creacion_png():
    iface.mapCanvas().refresh()
    #Obtengo las capas
    layers =  iface.legendInterface().layers()
    legend = iface.legendInterface()
    #Pondo una imagen de mas para que el codigo peuda llegar al final, eso es porque el bucle va con una iteracion de retraso

    for i in range(0,51):
        #Cargo una por una las capas a la vez que la anterior a la cargada
        layer=layers[i]
        layer_ante=layers[i-1]
        if layer.name()!="Zones_PREVIFOC_DS_ETRS89":
            print
            #Reproyecto las capas a ETRS89 porque si no lo hago salen desplazados en el mapa
            crs = layer.crs()
            crs.createFromId(4258)
            layer.setCrs(crs)
            #Empieza a desactivar todos los layers anteriores a excepcion de la primera iteraci?n
            if i>=2:
                legend.setLayerVisible(layer_ante, False)
                iface.mapCanvas().refresh()
                       layer.triggerRepaint()
                   #Activa el layer actual
                   legend.setLayerVisible(layer, True)
                   #Refresco el canvas
                   iface.mapCanvas().refresh()
                   layer.triggerRepaint()
                   #Le doy un segundo para que el programa reaccione
                   sleep(1)
```

```
                    #Creo el composer, lo que hace que pueda imprimir el canvas

                    canvas = iface.mapCanvas()
                    myFile =path_general+r'\qgis\plantilla.qpt'
                    myTemplateFile = file(myFile, 'rt')
                    myTemplateContent = myTemplateFile.read()
                    myTemplateFile.close()
                    composerAsDocument = QDomDocument()
                    composerAsDocument.setContent(myTemplateContent)
                    composition = QgsComposition(canvas.mapSettings())
                    composition.loadFromTemplate(composerAsDocument, {})
                    image = composition.printPageAsRaster(0)
                    #Empieza a guardar a partir de la segunda iteracion porque la primera imagen esta vacia, es por eso que hay 74 capas cargadas.
                    if i>=2:
                        #Guardo la imagen
                        if len(str(i-2))==1:
                            asw="0"+str(i-2)
                        else:
                            asw=str(i-2)
                        image.save(path_general+r'\datos\fotos_png_VE\output'+asw+'.png', "png")
            legend.setLayerVisible(layer, False)
            iface.mapCanvas().refresh()
            layer.triggerRepaint()
        creacion_png()
        #print("--- %s seconds ---" % (time.time() - start_time))
        #Cierre del proyecto y de QGIS
        iface.actionExit().trigger()
        os.kill(os.getpid(), 9)
```

## 10.1.6.- Creacion_png_macro_HR.py

```
        from PyQt4.QtCore import *
        from PyQt4.QtGui import *
        from PyQt4.QtXml import *
        from qgis.core import *
        from qgis.gui import *
        from qgis.utils import iface
        from time import sleep
        import os

        path_general=str((os.path.dirname(os.path.realpath(__file__)))[:-8])

        def creacion_png():
            iface.mapCanvas().refresh()
            #Obtengo las capas
            layers =  iface.legendInterface().layers()
            legend = iface.legendInterface()
            #Pondo una imagen de mas para que el codigo peuda llegar al final, eso es porque el bucle va con una iteracion de retraso

            for i in range(0,51):
                #Cargo una por una las capas a la vez que la anterior a la cargada
                layer=layers[i]
                layer_ante=layers[i-1]
                if layer.name()!="Zones_PREVIFOC_DS_ETRS89":
                    print
                    #Reproyecto las capas a ETRS89 porque si no lo hago salen desplazados en el mapa
```

```
                    crs = layer.crs()
                    crs.createFromId(4258)
                    layer.setCrs(crs)
                    #Empieza a desactivar todos los layers anteriores a excepci
    on de la primera iteraci?n
                    if i>=2:
                        legend.setLayerVisible(layer_ante, False)
                        iface.mapCanvas().refresh()
                        layer.triggerRepaint()
                    #Activa el layer actual
                    legend.setLayerVisible(layer, True)
                    #Refresco el canvas
                    iface.mapCanvas().refresh()
                    layer.triggerRepaint()
                    #Le doy un segundo para que el programa reaccione
                    sleep(1)
                    #Creo el composer, lo que hace que pueda imprimir el canvas

                    canvas = iface.mapCanvas()
                    myFile =path_general+r'\qgis\plantilla.qpt'
                    myTemplateFile = file(myFile, 'rt')
                    myTemplateContent = myTemplateFile.read()
                    myTemplateFile.close()
                    composerAsDocument = QDomDocument()
                    composerAsDocument.setContent(myTemplateContent)
                    composition = QgsComposition(canvas.mapSettings())
                    composition.loadFromTemplate(composerAsDocument, {})
                    image = composition.printPageAsRaster(0)
                    #Empieza a guardar a partir de la segunda iteracion porque
    la primera imagen esta vacia, es por eso que hay 74 capas cargadas.
                    if i>=2:
                        #Guardo la imagen
                        if len(str(i-2))==1:
                            asw="0"+str(i-2)
                        else:
                            asw=str(i-2)
                        image.save(path_general+r'\datos\fotos_png_HR\output'+a
    sw+'.png', "png")
            legend.setLayerVisible(layer, False)
            iface.mapCanvas().refresh()
            layer.triggerRepaint()
        creacion_png()
        #print("--- %s seconds ---" % (time.time() - start_time))
        #Cierre del proyecto y de QGIS
        iface.actionExit().trigger()
        os.kill(os.getpid(), 9)
```

## 10.1.7.- Creacion_png_macro_TE.py

```
        from PyQt4.QtCore import *
        from PyQt4.QtGui import *
        from PyQt4.QtXml import *
        from qgis.core import *
        from qgis.gui import *
        from qgis.utils import iface
        from time import sleep
        import os

        path_general=str((os.path.dirname(os.path.realpath(__file__)))[:-8])

        def creacion_png():
```

```
214.            iface.mapCanvas().refresh()
215.            #Obtengo las capas
216.            layers =  iface.legendInterface().layers()
217.            legend = iface.legendInterface()
218.            #Pondo una imagen de mas para que el codigo peuda llegar al final,
    eso es porque el bucle va con una iteracion de retraso
219.
220.            for i in range(0,51):
221.                #Cargo una por una las capas a la vez que la anterior a la carg
    ada
222.                layer=layers[i]
223.                layer_ante=layers[i-1]
224.                if layer.name()!="Zones_PREVIFOC_DS_ETRS89":
225.                    print
226.                    #Reproyecto las capas a ETRS89 porque si no lo hago salen d
    esplazados en el mapa
227.                    crs = layer.crs()
228.                    crs.createFromId(4258)
229.                    layer.setCrs(crs)
230.                    #Empieza a desactivar todos los layers anteriores a excepci
    on de la primera iteraci?n
231.                    if i>=2:
232.                        legend.setLayerVisible(layer_ante, False)
233.                        iface.mapCanvas().refresh()
234.                        layer.triggerRepaint()
235.                    #Activa el layer actual
236.                    legend.setLayerVisible(layer, True)
237.                    #Refresco el canvas
238.                    iface.mapCanvas().refresh()
239.                    layer.triggerRepaint()
240.                    #Le doy un segundo para que el programa reaccione
241.                    sleep(1)
242.                    #Creo el composer, lo que hace que pueda imprimir el canvas

243.                    canvas = iface.mapCanvas()
244.                    myFile =path_general+r'\qgis\plantilla.qpt'
245.                    myTemplateFile = file(myFile, 'rt')
246.                    myTemplateContent = myTemplateFile.read()
247.                    myTemplateFile.close()
248.                    composerAsDocument = QDomDocument()
249.                    composerAsDocument.setContent(myTemplateContent)
250.                    composition = QgsComposition(canvas.mapSettings())
251.                    composition.loadFromTemplate(composerAsDocument, {})
252.                    image = composition.printPageAsRaster(0)
253.                    #Empieza a guardar a partir de la segunda iteracion porque
    la primera imagen esta vacia, es por eso que hay 74 capas cargadas.
254.                    if i>=2:
255.                        #Guardo la imagen
256.                        if len(str(i-2))==1:
257.                            asw="0"+str(i-2)
258.                        else:
259.                            asw=str(i-2)
260.                        image.save(path_general+r'\datos\fotos_png_TE\output'+a
    sw+'.png', "png")
261.            legend.setLayerVisible(layer, False)
262.            iface.mapCanvas().refresh()
263.            layer.triggerRepaint()
264.        creacion_png()
265.        #print("--- %s seconds ---" % (time.time() - start_time))
266.        #Cierre del proyecto y de QGIS
267.        iface.actionExit().trigger()
268.        os.kill(os.getpid(), 9)
```

### 10.1.8.- Execute_qgis4_png_VD.py

```
import os
import time

def execute_qgis(path_general):
    execute_qgis=r"call C:\PROGRA~1\QGIS2~1.18\bin\qgis --nologo --project "+path_general+r"\qgis\creacion_png_VD.qgs"
    os.system(execute_qgis)
    #Suspendo el programa para que le de tiempo a QGIS a crear todas las imagenes
    time.sleep(180)
```

### 10.1.9.- Execute_qgis4_png_VE.py

```
import os
import time

def execute_qgis(path_general):
    execute_qgis=r"call C:\PROGRA~1\QGIS2~1.18\bin\qgis --nologo --project "+path_general+r"\qgis\creacion_png_VE.qgs"
    os.system(execute_qgis)
    #Suspendo el programa para que le de tiempo a QGIS a crear todas las imagenes
    time.sleep(180)
```

### 10.1.10.- Execute_qgis4_png_HR.py

```
import os
import time

def execute_qgis(path_general):
    execute_qgis=r"call C:\PROGRA~1\QGIS2~1.18\bin\qgis --nologo --project "+path_general+r"\qgis\creacion_png_HR.qgs"
    os.system(execute_qgis)
    #Suspendo el programa para que le de tiempo a QGIS a crear todas las imagenes
    time.sleep(180)
```

### 10.1.11.- Execute_qgis4_png_TE.py

```
import os
import time

def execute_qgis(path_general):
    execute_qgis=r"call C:\PROGRA~1\QGIS2~1.18\bin\qgis --nologo --project "+path_general+r"\qgis\creacion_png_TE.qgs"
    os.system(execute_qgis)
    #Suspendo el programa para que le de tiempo a QGIS a crear todas las imagenes
    time.sleep(180)
```

## 10.1.12.- Execute_qgis4_shp.py

```
1.  import os
2.  import time
3.
4.  def execute_qgis_shp(path_general):
5.      execute_qgis=r"call C:\PROGRA~1\QGIS2~1.18\bin\qgis --nologo --
    project "+path_general+r"\qgis\creacion_shp.qgs"
6.      os.system(execute_qgis)
7.      #Suspendo el programa para que le de tiempo a QGIS a crear el shapefile
8.      time.sleep(5400)
```

## 10.1.13.- New_layer.py

```
1.  from PyQt4.QtCore import *
2.  from qgis.core import *
3.  from qgis.utils import *
4.  import os
5.  import time
6.  import os.path
7.  import math
8.  import csv
9.  import numpy
10.
11. path_general=str((os.path.dirname(os.path.realpath(__file__)))[:-8])
12.
13. def crear_shape():
14.     #Creo un nuevo layer sobre el cual se guardara la informacion de las 73 ca
    pas, este tiene una proyection ETRS89 al igual que las demas capas transformad
    as
15.     vl = QgsVectorLayer("Point?crs=epsg:25830", "temporary_points", "memory")

16.     writer = QgsVectorFileWriter.writeAsVectorFormat(vl,path_general+r"\datos\
    shp\NewFile.shp","utf-8",None,"ESRI Shapefile")
17.     layer_total = QgsVectorLayer(path_general+r"\datos\shp\NewFile.shp", "SHP_
    V", "ogr")
18.
19.     name=os.listdir(path_general+r"\datos\grib1")[0].split('+')[0][2:]
20.     print name
21.     horiz=name[:4]+r"_"+name[4:-4]+r"_"+name[6:-2]+r"_"+name[8:]
22.     print horiz
23.
24.     cnt=0
25.     for fil in os.listdir(path_general+r"\datos\grib2"):
26.         #Creando los csv que contienen el componente U, V, TE y HR
27.         print fil
28.         create_u=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
    al+r"\datos\grib2"+"\\"+fil+r" -C -msg 1 -Csv -
    out "+path_general+r"\datos\shp\csv_u.csv"
29.         create_v=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
    al+r"\datos\grib2"+"\\"+fil+r" -C -msg 2 -Csv -
    out "+path_general+r"\datos\shp\csv_v.csv"
30.         create_t=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
    al+r"\datos\grib2"+"\\"+fil+r" -C -msg 3 -Csv -
    out "+path_general+r"\datos\shp\csv_tF.csv"
31.         create_h=path_general+r"\programas\ndfd\degrib\bin\degrib "+path_gener
    al+r"\datos\grib2"+"\\"+fil+r" -C -msg 4 -Csv -
    out "+path_general+r"\datos\shp\csv_h.csv"
32.         os.system(create_u)
```

```
        os.system(create_v)
        os.system(create_t)
        os.system(create_h)
        #Cargando dichos csv
        dat_u=numpy.genfromtxt(path_general+r"\datos\shp\csv_u.csv",delimiter=
',')
        dat_u = numpy.delete(dat_u,0, axis=0)
        dat_v=numpy.genfromtxt(path_general+r"\datos\shp\csv_v.csv",delimiter=
',')
        dat_v = numpy.delete(dat_v,0, axis=0)
        dat_tF=numpy.genfromtxt(path_general+r"\datos\shp\csv_tF.csv",delimite
r=',')
        dat_tF = numpy.delete(dat_tF,0, axis=0)
        dat_h=numpy.genfromtxt(path_general+r"\datos\shp\csv_h.csv",delimiter=
',')
        dat_h = numpy.delete(dat_h,0, axis=0)
        pr = layer_total.dataProvider()
        crsSrc = QgsCoordinateReferenceSystem(4258)
        crsDest = QgsCoordinateReferenceSystem(25830)
        xform = QgsCoordinateTransform(crsSrc, crsDest)

        #En la primera iteracion se llena lel nuevo .shp con los datos de los
ID y las coordenadas de cada uno de los puntos
        if cnt==0:
            layer_total.dataProvider().addAttributes( [ QgsField("ID", QVarian
t.Int)] )
            layer_total.dataProvider().addAttributes( [ QgsField("Horizonte",
QVariant.String,"string",13)] )
            layer_total.dataProvider().addAttributes( [ QgsField("X_ETRS89", Q
Variant.Double, "double", 12, 3)] )
            layer_total.dataProvider().addAttributes( [ QgsField("Y_ETRS89", Q
Variant.Double, "double", 12, 3)] )
            layer_total.dataProvider().addAttributes( [ QgsField("ETRS89_lat",
 QVariant.Double, "double", 6, 3)] )
            layer_total.dataProvider().addAttributes( [ QgsField("ETRS89_lon",
 QVariant.Double, "double", 6, 3)] )

            for rowu in dat_u:
                    rx=str(int(rowu[0]))
                    if len(rx)==1:
                       rx="0"+rx
                    ry=str(int(rowu[1]))
                    if len(rx)==1:
                       ry="0"+ry
                    ID_pnt=rx+'00'+ry
                    x=QgsPoint(float(rowu[3]),float(rowu[2]))
                    pt = xform.transform(x)
                    feat = QgsFeature()
                    feat.setGeometry(QgsGeometry.fromPoint(QgsPoint(pt)))
                    feat.setAttributes([int(ID_pnt), horiz, pt[0], pt[1],float
(rowu[2]),float(rowu[3])])
                    layer_total.dataProvider().addFeatures([feat])
        #Para cada iteracion de las capas se crean 6 campos nuevos que contien
en los datos de las direcciones de los vientos N.E, el total, la clase de feno
meno que es en escala de Bouefort y el azimut del viento
        layer_total.dataProvider().addAttributes( [ QgsField("C_U_"+str(cnt),
QVariant.Double, "double", 5, 3)] )
        layer_total.dataProvider().addAttributes( [ QgsField("C_V_"+str(cnt),
QVariant.Double, "double", 5, 3)] )
        layer_total.dataProvider().addAttributes( [ QgsField("C_T_"+str(cnt),
QVariant.Double, "double", 5, 3)] )
        layer_total.dataProvider().addAttributes( [ QgsField("C_A_"+str(cnt),
QVariant.Double, "double", 8, 1)] )
```

```
78.         layer_total.dataProvider().addAttributes( [ QgsField("C_TMP_"+str(cnt)
    , QVariant.Double, "double", 8, 3)] )
79.         layer_total.dataProvider().addAttributes( [ QgsField("C_H_"+str(cnt),
    QVariant.Double, "double", 5, 3)] )
80.         #Cargo los valores de los rows de las tablas u v y final
81.         fin_feats=layer_total.getFeatures()
82.         #Se presupone que los  datos siempre siguen el mismo orden, para ahorr
    ar tiempo de calculo
83.         cnt_attb=0
84.         for fin_feat in fin_feats:
85.             fin_attrs = fin_feat.attributes()
86.             puVal=float(dat_u[cnt_attb][4])
87.             pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_U_
    "+str(cnt)] : puVal}})
88.             pvVal=float(dat_v[cnt_attb][4])
89.             pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_V_
    "+str(cnt)] : pvVal}})
90.             ptFVal=float(dat_tF[cnt_attb][4])
91.             pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_TM
    P_"+str(cnt)] : ptFVal}})
92.             phVal=float(dat_h[cnt_attb][4])
93.             pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap()["C_H_
    "+str(cnt)] : phVal}})
94.             cnt_attb+=1
95. #           Ahora que tenemos el valor de U y V se puede calcular el Azimut, d
    esde el Sur, y el valor total
96.             Tval=math.sqrt(puVal**2+pvVal**2)
97.             dx=puVal
98.             dy=pvVal
99.             #Si alguno de estos valroes de 0 la funcion del azimut da error po
    r lo que se le da un valor minusculo
100.                    if dx==0:
101.                        dx=0.00000000001
102.                    if dy==0:
103.                        dy=0.00000000001
104.                    if dx>=0 and dy>=0:
105.                        c=0
106.                    if dx>=0 and dy<=0:
107.                        c=180
108.                    if dx<=0 and dy<=0:
109.                        c=180
110.                    if dx<=0 and dy>=0:
111.                        c=360
112.                    azi=c+math.atan((dx)/(dy))*180/math.pi
113.                    azi=azi-180.0
114.                    if azi<0:
115.                        azi=azi+360.0
116.                    if azi>360:
117.                        azi=azi-360.0
118.                    pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap(
    )["C_T_"+str(cnt)] : Tval}})
119.                    pr.changeAttributeValues({fin_feat.id() : {pr.fieldNameMap(
    )["C_A_"+str(cnt)] : azi}})
120.                cnt+=1
121.        crear_shape()
122.        iface.actionExit().trigger()
123.        os.kill(os.getpid(), 9)
```

## 10.1.14.- Geoserver_cURL.py

```
1.  import requests
2.  import os,glob
3.
4.
5.  def geoserver_layers():
6.      headers = {'Content-type': 'text/xml'}
7.      #Nombre del Workspace
8.      wk='AEMET'
9.      #Directorio del Workspace, se conecta al servidor de Vaersa
10.     dr_wkspc=r"\\arcgis.vaersa.org"+"\\"+wk
11.     #Compreuba dentro del directorio que capas hay disponibles
12.     dirs = [d for d in os.listdir(dr_wkspc) if os.path.isdir(os.path.join(dr_w
    kspc, d))]
13.     #Borra todas las capas con im?genes
14.     for img_st in dirs:
15.         #Para que salte las carpetas del shape y de estilos
16.         if img_st!='AEMET_shp' and img_st!='styles':
17.             #ahora hay que borrar el storage junto con la capa, lo ahce el rec
    urse
18.             r1 = requests.delete("http://arcgis.vaersa.org:8080/geoserver/rest
    /workspaces/"+wk+"/coveragestores/"+img_st+"?recurse=true",
19.                 #Todas las conexiones requieren tener de antemano el usuario y
     contrase?a de Geoserver, adem?s de los permisos para poder modificar los dato
    s.
20.                 #En este caso se han cubierto por seguridad.
21.                 auth=('******', '**********'),
22.                 headers=headers)
23.
24.     #Actualiza el .shp del servidor, no hace falta borrarla previamente porque
     el nombre del .shp no cambia
25.     headers = {'Content-
    type':  "application/zip", 'Accept': 'application/xml'}
26.     bakemono=file(r'\\arcgis.vaersa.org\datos_visor_previsiones\shp\shp_cmp.zi
    p','rb').read()
27.     r1 = requests.put("http://arcgis.vaersa.org:8080/geoserver/rest/workspaces
    /AEMET/datastores/AEMET_shp/file.shp",
28.         auth=('******', '**********'),
29.         data= bakemono,
30.         headers=headers)
31.     #Sube las im?genes de los vientos con flechas
32.     headers = {'Content-type': 'text/xml'}
33.     os.chdir(r"\\arcgis.vaersa.org\datos_visor_previsiones\png_VE")
34.     for filep in glob.glob("*.png"):
35.         if "output" not in filep:
36.             nombre=filep.split(r'.')[0]
37.             #Primero crea el CoverageStore, el de d?nde se van a obtener los d
    atos
38.             r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver/rest/w
    orkspaces/"+wk+"/coveragestores",
39.                 auth=('******', '**********'),
40.                 data= '<coverageStore> \
41.                     <name>'+nombre+r'_storage'+'</name> \
42.                     <workspace>'+wk+'</workspace> \
43.                     <enabled>true</enabled> \
44.                     <type>WorldImage</type> \
45.                     <url>datos_visor_previsiones/png_VE/'+filep+'</url> \
46.                     </coverageStore>'  ,
47.                 headers=headers)
```

```
48.             #Seguidamente crea el layer, en el caso de las im?genes s?lo hace
    falta el nombre de la capa, t?tulo, sistema de referencia y extensi?n, la cual
     se calcula autom?ticamente gracias al comando del POST.
49.             #La parte de recalculate se encarga de calcular la extension.
50.             r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver/rest/w
    orkspaces/"+wk+"/coveragestores/"+nombre+r'_storage'+"/coverages?recalculate=n
    ativebbox,latlonbbox",
51.                 auth=('******', '**********'),
52.                 data= '<coverage> \
53.                       <name>'+nombre+r'_coverage'+'</name>\
54.                       <title>'+nombre+r'_coverage'+'</title>\
55.                       <srs>EPSG:25830</srs>\
56.                       </coverage>' ,
57.                 headers=headers)
58.     #Sube las im?genes de las Temperaturas
59.     os.chdir(r"\\arcgis.vaersa.org\datos_visor_previsiones\png_TE")
60.     for filep in glob.glob("*.png"):
61.         if "output" not in filep:
62.             nombre=filep.split(r'.')[0]
63.             r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver/rest/w
    orkspaces/"+wk+"/coveragestores",
64.                 auth=('******', '**********'),
65.                 data= '<coverageStore> \
66.                      <name>'+nombre+r'_storage'+'</name> \
67.                      <workspace>'+wk+'</workspace> \
68.                      <enabled>true</enabled> \
69.                      <type>WorldImage</type> \
70.                      <url>datos_visor_previsiones/png_TE/'+filep+'</url> \
71.                      </coverageStore>'  ,
72.                 headers=headers)
73.             #Create Layer
74.             #La parte de recalculate se encarga de calcular la extension.
75.             r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver/rest/w
    orkspaces/"+wk+"/coveragestores/"+nombre+r'_storage'+"/coverages?recalculate=n
    ativebbox,latlonbbox",
76.                 auth=('******', '**********'),
77.                 data= '<coverage> \
78.                       <name>'+nombre+r'_coverage'+'</name>\
79.                       <title>'+nombre+r'_coverage'+'</title>\
80.                       <srs>EPSG:25830</srs>\
81.                       </coverage>' ,
82.                 headers=headers)
83.     #Sube las im?genes de las Humedades
84.     os.chdir(r"\\arcgis.vaersa.org\datos_visor_previsiones\png_HR")
85.     for filep in glob.glob("*.png"):
86.         if "output" not in filep:
87.             nombre=filep.split(r'.')[0]
88.             r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver/rest/w
    orkspaces/"+wk+"/coveragestores",
89.                 auth=('******', '**********'),
90.                 data= '<coverageStore> \
91.                      <name>'+nombre+r'_storage'+'</name> \
92.                      <workspace>'+wk+'</workspace> \
93.                      <enabled>true</enabled> \
94.                      <type>WorldImage</type> \
95.                      <url>datos_visor_previsiones/png_HR/'+filep+'</url> \
96.                      </coverageStore>'  ,
97.                 headers=headers)
98.             #Create Layer
99.             #La parte de recalculate se encarga de calcular la extension.
100.                r1 = requests.post("http://arcgis.vaersa.org:8080/geoserver
    /rest/workspaces/"+wk+"/coveragestores/"+nombre+r'_storage'+"/coverages?recalc
    ulate=nativebbox,latlonbbox",
```

```
                        auth=('******', '**********'),
                        data= '<coverage> \
                              <name>'+nombre+r'_coverage'+'</name>\
                              <title>'+nombre+r'_coverage'+'</title>\
                              <srs>EPSG:25830</srs>\
                              </coverage>' ,
                        headers=headers)
```

## 10.2.- Código del visor WEB

### 10.2.1.- Visor.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Image ArcGIS MapServer</title>
    <link rel="stylesheet" href="https://openlayers.org/en/v4.2.0/css/ol.css" type="text/css">
    <!--
- The line below is only needed for old environments like Internet Explorer and Android 4.x -->
    <script src="https://cdn.polyfill.io/v2/polyfill.min.js?features=requestAnimationFrame,Element.prototype.classList,URL"></script>
    <script src="https://openlayers.org/en/v4.2.0/build/ol.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/1.3.3/FileSaver.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/proj4js/2.3.12/proj4-src.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/proj4js/2.3.12/proj4.js"></script>
    <script src="https://code.jquery.com/jquery-2.2.3.min.js"></script>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

    <!-- Optional theme -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

    <!-- Latest compiled and minified JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

    <script type = "text/javascript" src = "visor.js"></script>
  </head>
  <body>
    <div id="cont_map" >
        <div id="cargando">
            <img src="loading2.gif" style="width: 300px;height: auto;position: absolute;z-index:2;top:100px;">
            <div style="height: 500px;width: 300px;background-color:#A9A9A9;opacity:0.5;position:absolute;z-index:1;"></div>
```

```
30.         </div>
31.         <div id="map" class="map"></div>
32.         <div id="capas" style="border: 1px solid black;width: 300px;padding:10
    px;float: left;">
33.             <p>Activar/Desactivar Capas</p>
34.             <label><input type="checkbox" checked name="vehicle" onclick="capa
    _vis();"> Municipios</label><br>
35.             <label><input type="checkbox" checked name="vehicle" onclick="capa
    _vis();"> Variable</label><br>
36.             <label><input type="checkbox" checked name="vehicle" onclick="capa
    _vis();"> Dirección del viento</label><br>
37.         </div>
38.         <div id="contendor_barra">
39.             <div style="padding-bottom:5%">
40.                 <input type="range" min="0" max="48" value="0" class="slider"
    id="valor_Rango" title="Texto del hover">
41.             </div>
42.             <p>Horizonte de la Previsión: <span id="fecha_img"></span></p>
43.             <p>Previsión elaborada el: <span id="fecha_creacion"></span></p>
44.         </div>
45.         <div id="contendor_Leyenda" style="border: 1px solid black;width: 300p
    x;height: 312px;padding:10px;float: left;">
46.             <p>Leyenda (Km/h)</p>
47.             <img src="leyenda_VE.PNG" >
48.         </div>
49.         <select id="seleccion_visor" style="position:absolute;left:100px;top:5
    05px;">
50.           <option >Viento</option>
51.           <option >Temperatura</option>
52.           <option >Humedad</option>
53.         </select>
54.     </div>
55.     <style>
56.     label{
57.      font-weight : normal;
58.     }
59.     #cont_map {
60.         margin:0;
61.         padding:0;
62.         height: 500px;
63.         width: 600px;
64.     }
65.     #map{
66.         height: 500px;
67.         width: 300px;
68.         border: 1px solid black;
69.         z-index:0;
70.         float: left;
71.     }
72.     #contendor_barra {
73.         float:left;
74.         border: 1px solid black;
75.         padding:10px;
76.         width: 300px;
77.     }
78.
79.     .slider {
80.         -webkit-appearance: none;
81.         width: 100%;
82.         /*height: 5px;*/
83.         border-radius: 5px;
84.         background: #d3d3d3;
85.         outline: none;
```

```
        opacity: 0.7;
        -webkit-transition: .2s;
        transition: opacity .2s;
    }


    .slider:hover {
        opacity: 1;
    }

    .slider::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 10px;
               height: 15px;
               border-radius: 30%;
               background: #4CAF50;
               cursor: pointer;
          }

          .slider::-moz-range-thumb {
              width: 25px;
              height: 25px;
              border-radius: 50%;
              background: #4CAF50;
              cursor: pointer;
          }
          </style>
        </body>
      </html>
```

## 10.2.2.- Visor.js

```
var tod_features;
var horizonte;
var source_layer;
var scale_value=1
var ck_img;
var ck_fle;
var ck_mun;
var cnt_carga;
var layerCounter = 0;

window.onload = function() {
    var slider = document.getElementById("valor_Rango");
    var output = document.getElementById("fecha_img");
    var output_crea = document.getElementById("fecha_creacion");
    var myProjectionName = 'EPSG:25830';
    proj4.defs(myProjectionName, "+proj=utm +zone=30 +ellps=GRS80 +units=m +no_defs");

    var mousePositionControl = new ol.control.MousePosition({
      coordinateFormat: ol.coordinate.createStringXY(0),
      projection: 'EPSG:25830',

    });
    urlsig="http://arcgis.vaersa.org/ArcGIS/rest/services/contorno_municipios/MapServer"
```

```
    layer_muni=new ol.layer.Image({
        title: 'Municipios',
        name: 'Municipios',
        source: new ol.source.ImageArcGISRest({
            url: urlsig,
            crossOrigin: 'anonymous',
            rendererOptions: { zIndexing: true },
            params:{LAYERS:"show:0", }
        })
    })
    layer = new ol.layer.Image({
          title: 'Imagen',
          name: 'Imagen de Intensidades',
        })
    flechas = new ol.layer.Image({
          title: 'Imagen',
          name: 'Flechas',
        })

    var layers = [
            layer,
            layer_muni,
            flechas

        ];

    map = new ol.Map({
        controls: [mousePositionControl],

        layers: layers,
        logo: false,
        target: 'map',

        view: new ol.View({
            projection: 'EPSG:25830',
            center: ol.proj.transform([-
0.5, 39.5],'EPSG:4326','EPSG:25830'),
            zoom: 8,
            zoomFactor: 1.8
        }),
    });



    //Se hace Zoom a la Comunidad Valenciana
    map.getView().fit([626000, 4190000, 815000, 4519000], map.getSize());
    //Rescato el primer punto de los 11k y solo el atributo del horizonte para
 saber cuando se calcularon los datos
    var url = "http://localhost:8080/geoserver/wfs?service=wfs&version=1.0.0&r
equest=GetFeature&typeName=AEMET:shp_geoserver&outputformat=text/javascript&fo
rmat_options=callback:getJson&cql_filter=ID=1001&propertyName=Horizonte";
    $.ajax({jsonp: false,jsonpCallback: 'getJson',type: 'GET',url: url,async:
false,dataType: 'jsonp',success: function(response) {
        tod_features=response.features;
        horizonte=tod_features[0].properties.Horizonte
        var temp_part=horizonte.split('_')
        var hor_js= new Date( Date.UTC(temp_part[0],temp_part[1]-
1,temp_part[2],temp_part[3]) );
        hor_js.setHours(hor_js.getHours())
        var any_c=hor_js.getFullYear().toString()
        var mes_c=(hor_js.getMonth()+1).toString()
        if (mes_c.length==1){
            mes_c="0"+mes_c
```

```
        }
        var dia_c=hor_js.getDate().toString()
        if (dia_c.length==1){
            dia_c="0"+dia_c
        }
        var h_c=hor_js.getHours().toString()
        if (h_c.length==1){
            h_c="0"+h_c
        }
        var fecha_img_viento=any_c+"/"+mes_c+"/"+dia_c+" "+h_c+"h"
        output_crea.innerHTML = fecha_img_viento;
        //Una vez se sabe el horizonte sobre el que se trabajara, se puede hac
    er peticiones a las capas de las intensidades y flechas.
        carga_vientos()
    }});
    output.innerHTML = slider.value;
    slider.setAttribute("title", slider.value);
    function carga_vientos(){
        document.getElementById('cargando').style.display = 'block'
        //COGE EL HORIZONTE DEL SHP
                var temp_part=horizonte.split('_')

                var hor_js= new Date(temp_part[0],temp_part[1]-
    1,temp_part[2],temp_part[3] );
                hor_js.setHours(parseInt(hor_js.getHours()) + parseInt(slider.v
    alue))
                var any_c=hor_js.getFullYear().toString()
                var mes_c=(hor_js.getMonth()+1).toString()
                if (mes_c.length==1){
                    mes_c="0"+mes_c
                }
                var dia_c=hor_js.getDate().toString()
                if (dia_c.length==1){
                    dia_c="0"+dia_c
                }
                var h_c=hor_js.getHours().toString()
                if (h_c.length==1){
                    h_c="0"+h_c
                }

                if (document.getElementById("seleccion_visor").value=='Viento')
    {
                    prefijo='VD'
                }else if (document.getElementById("seleccion_visor").value=='Te
    mperatura'){
                    prefijo='TE'
                }else{
                    prefijo='HR'
                }

                var fecha_img_viento=any_c+"_"+mes_c+"_"+dia_c+"_"+h_c
                source_layer= new ol.source.ImageWMS({
                    url: 'http://localhost:8080/geoserver/AEMET/wms',
                    params: {'LAYERS': 'AEMET:'+prefijo+'_'+fecha_img_viento+'_
    coverage'},
                    ratio: 1,
                    serverType: 'geoserver'
                  })
                //MUY IMPORANTE, le dice al OL que el source de la capa (layer)
     ha cambiado
                layer.setSource(source_layer)
                if (document.getElementById("seleccion_visor").value=='Viento')
    {
```

```
                    var sldBody =
                    '<?xml version="1.0" encoding="ISO-8859-1"?>\
                    <StyledLayerDescriptor version="1.0.0"\
                    xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"\
                    xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"\
                    xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">\
                      <NamedLayer>\
                        <Name>AEMET:shp_geoserver</Name>\
                        <UserStyle>\
                          <Title>Point Stacker</Title>\
                          <Abstract>A sample style that aggregates points</Abstract>\
                          <FeatureTypeStyle>\
                            <Rule>\
                              <TextSymbolizer>\
                                 <Label><![CDATA[ ]]></Label> <!-- fake label -->\
                                 <Graphic>\
                                   <Mark>\
                                     <WellKnownName>ttf://Segoe UI Symbol#0x2193</WellKnownName>\
                                     <Fill>\
                                       <CssParameter name="fill">#000000</CssParameter>\
                                     </Fill>\
                                   </Mark>\
                                 <Size>20</Size>\
                                 <Rotation>\
                                   <PropertyName>C_A_'+slider.value.toString()+'</PropertyName>\
                                 </Rotation>\
                               </Graphic>\
                               <LabelPlacement>\
                                   <PointPlacement>\
                                     <AnchorPoint>\
                                       <AnchorPointX>1.0</AnchorPointX>\
                                       <AnchorPointY>1.0</AnchorPointY>\
                                     </AnchorPoint>\
                                     <Displacement>\
                                       <DisplacementX>0</DisplacementX>\
                                       <DisplacementY>-5</DisplacementY>\
                                     </Displacement>\
                                   </PointPlacement>\
                               </LabelPlacement>\
                             </TextSymbolizer>\
                           </Rule>\
                         </FeatureTypeStyle>\
                       </UserStyle>\
                     </NamedLayer>\
                    </StyledLayerDescriptor>' ;
                    source_flechas= new ol.source.ImageWMS({
                        url: 'http://localhost:8080/geoserver/AEMET/wms',
                        params: {'LAYERS': 'AEMET:shp_geoserver', 'SLD_BODY' : sldBody},
                        ratio: 1,
                        serverType: 'geoserver'
                      })
                    //MUY IMPORANTE, le dice al OL que el source de la capa (layer) ha cambiado
                    flechas.setSource(source_flechas)
```

```
189.                }
190.                for (i=0; i<layers.length; i++) {
191.                 laker=layers[i]
192.
193.                 laker.getSource().on('imageloadstart', function(event) {
194.                     layerCounter++;
195.                 });
196.                 laker.getSource().on('imageloadend', function(event) {
197.                     layerCounter--;
198.                     if (layerCounter == 0) {
199.                         document.getElementById('cargando').style.display = 'n
    one'
200.                     }
201.                 });
202.                }
203.
204.                slider.setAttribute("title", fecha_img_viento);
205.                //Para tener la fecha en formato local
206.                var hor_js= new Date( Date.UTC(temp_part[0],temp_part[1]-
    1,temp_part[2],temp_part[3]) );
207.                hor_js.setHours(parseInt(hor_js.getHours()) + parseInt(slider.v
    alue))
208.                var any_c=hor_js.getFullYear().toString()
209.                var mes_c=(hor_js.getMonth()+1).toString()
210.                if (mes_c.length==1){
211.                    mes_c="0"+mes_c
212.                }
213.                var dia_c=hor_js.getDate().toString()
214.                if (dia_c.length==1){
215.                    dia_c="0"+dia_c
216.                }
217.                var h_c=hor_js.getHours().toString()
218.                if (h_c.length==1){
219.                    h_c="0"+h_c
220.                }
221.                var fecha_img_viento=any_c+"_"+mes_c+"_"+dia_c+"_"+h_c
222.                output.innerHTML = fecha_img_viento.replace("_", "/").replace("
    _", "/").replace("_", " ")+"h";
223.                slider.setAttribute("title", fecha_img_viento);
224.                //document.getElementById('cargando').style.display = 'none'
225.
226.            }
227.            //FUNCIONES QUE CONTROLAN EL SLIDER
228.            output.innerHTML = slider.value;
229.            slider.setAttribute("title", slider.value);
230.            slider.onchange = function() {
231.              carga_vientos()
232.            }
233.            slider.oninput = function() {
234.              carga_vientos()
235.            }
236.            //Cambia entre diferentes mapas, Vientos, Temperaturas o humedades

237.            document.getElementById("seleccion_visor").addEventListener("change
    ", function (){
238.                carga_vientos()
239.            if (document.getElementById("seleccion_visor").value=='Viento'){
240.                document.getElementById('contendor_Leyenda').getElementsByTagNa
    me('img')[0].src='leyenda_VE.PNG'
241.                document.getElementById('contendor_Leyenda').getElementsByTagNa
    me('p')[0].innerHTML='Leyenda (Km/h)'
242.                document.getElementById('capas').getElementsByTagName('input')[
    2].disabled = false;
```

```
243.                document.getElementById('capas').getElementsByTagName('input')[
    2].checked = true;
244.                document.getElementById('capas').getElementsByTagName('label')[
    2].style.opacity=1;
245.                flechas.setVisible(true)
246.            }else if (document.getElementById("seleccion_visor").value=='Temper
    atura'){
247.                document.getElementById('contendor_Leyenda').getElementsByTagNa
    me('img')[0].src='leyenda_TEv2.PNG'
248.                document.getElementById('contendor_Leyenda').getElementsByTagNa
    me('p')[0].innerHTML='Leyenda (ยบC)'
249.                document.getElementById('capas').getElementsByTagName('input')[
    2].disabled = true;
250.                document.getElementById('capas').getElementsByTagName('input')[
    2].checked = false;
251.                document.getElementById('capas').getElementsByTagName('label')[
    2].style.opacity=0.6;
252.                flechas.setVisible(false)
253.            }else{
254.                document.getElementById('contendor_Leyenda').getElementsByTagNa
    me('img')[0].src='leyenda_HR.PNG'
255.                document.getElementById('contendor_Leyenda').getElementsByTagNa
    me('p')[0].innerHTML='Leyenda (%)'
256.                document.getElementById('capas').getElementsByTagName('input')[
    2].disabled = true;
257.                document.getElementById('capas').getElementsByTagName('input')[
    2].checked = false;
258.                document.getElementById('capas').getElementsByTagName('label')[
    2].style.opacity=0.6;
259.                flechas.setVisible(false)
260.            }
261.                });
262.        }
263.
264.        function capa_vis(){
265.
266.            if (document.getElementById('capas').getElementsByTagName('input')[
    2].checked) {
267.                flechas.setVisible(true)
268.            }else{
269.                flechas.setVisible(false)
270.            }
271.            if (document.getElementById('capas').getElementsByTagName('input')[
    0].checked) {
272.                layer_muni.setVisible(true)
273.            }else{
274.                layer_muni.setVisible(false)
275.            }
276.            if (document.getElementById('capas').getElementsByTagName('input')[
    1].checked) {
277.                layer.setVisible(true)
278.            }else{
279.                layer.setVisible(false)
280.            }
281.
```