UNIVERSIDAD DE VALLADOLID


E.T.S.I. TELECOMUNICACIÓN

# TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN, MENCIÓN EN SISTEMAS DE TELECOMUNICACIÓN

# Recogida, automatización y representación de datos para la analítica de un sistema digital inteligente agro

Autor:

**D. Jaime Antolín Martínez**

Tutoras:

**Dña. Noemí Merayo Álvarez**

**Dña. Patricia Fernández del Reguero**

TÍTULO: **Recogida, automatización y representación de datos para la analítica de un sistema digital inteligente agro**

AUTOR: **Jaime Antolín Martínez**

TUTORAS: **Noemí Merayo Álvarez / Patricia Fernández del Reguero**

DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

**TRIBUNAL**

PRESIDENTE: Patricia Fernández del Reguero

SECRETARIO: Noemí Merayo Álvarez

VOCAL: Rubén M. Lorenzo Toledo

SUPLENTE: J. Carlos Aguado Manzano

SUPLENTE: Ramón J. Durán Barroso

## Resumen de TFG

El *Business Analytics* es una parte muy importante en el mundo de los negocios hoy en día. Este Trabajo de Fin de Grado quiere ayudar a Grupo Matarromera a llevar sus negocios al siguiente nivel. Gracias a sus sensores y a las APIs de estos, se puede construir una base de datos donde almacenar todos los datos y mostrarlos después en un visualizador de datos. También se pueden usar otras APIs, y así contribuir a mejorar la base de datos y añadir más información, que puede ser mostrada en cualquier momento. Estos datos también pueden ser usados junto a las IA, pudiendo así desarrollar sistemas automatizados.



## Palabras clave

Cesens, Modpow, Base de Datos, API (Interfaz de Programación de Aplicaciones), Visualizador de Datos, Microsoft Power Bi, Sentinel, Python, SQL.



## Abstract

A Business Analytics is an especially important part in the business world nowadays. This Final Degree Project wants to help Matarromera Group to take their business to a higher level. Thanks to this company's sensors and their APIs, a database can be built to store all the data from the field and show them in a dada viewer. Also, other sensors and APIs can be used, contributing to the performance of the database, and adding more data, which can be shown at any moment. These data also can be used with AI, developing automated systems.



## Keywords

Cesens, Modpow, Database, API (Application Programming Interface), Data Viewer, Microsoft Power Bi, Sentinel, Python, SQL.

# Agradecimientos

*A todos los compañeros que he tenido durante la carrera, quienes me han ayudado en momentos de necesidad.*

*A Grupo Matarromera por darme la oportunidad de realizar con ellos mis prácticas de empresa, y en especial a Adrián y Jaime, mis compañeros durante mi periplo en Emina.*

*A Noemí y Patricia, mis tutoras del TFG, por las horas que me dedicaron durante la realización de este proyecto.*

*Y por último, pero no menos importante, a mis padres, siempre animando, siempre apoyando por mal dadas que viniesen, pero que no desistieron y siempre estuvieron conmigo.*

*Y a mi abuela Veva, allí donde estes, estoy también es tuyo. Gracias.*

# Índice

# Índice de figuras

# Capítulo 1. Introducción

## 1.1. Motivación

Este Trabajo de Fin de Grado, realizado con la ayuda de Grupo Matarromera, se divide en tres partes bien diferenciadas: código y base de datos, visualización de datos e imágenes satélite.

La motivación para la realización de este proyecto surge a raíz de la necesidad por parte de Grupo Matarromera de crear un sistema automatizado para regar y fertilizar cultivos. Puesto que en su haber poseen sensores capaces de guardar los datos de la tierra de los cultivos, surgió la idea de poder almacenar todos estos datos en una única base de datos, a partir de la cual se mostrarían estos mismos datos para su análisis.

De ahí surgió la idea y la necesidad de este proyecto, para poder almacenar todos estos datos, y hacerlo de forma automática, y poder, en un futuro, aplicar una IA que logre cumplir el objetivo de este trabajo.

## 1.2. Objetivos

Los objetivos de este Trabajo de Fin de Grado son dos.

En primer lugar, la creación de un sistema automatizado mediante el cual se puedan recoger en tiempo real y de forma periódica diferentes tipos de datos provenientes de diferentes APIs, almacenándolos en una base de datos, para después mostrarlos con una herramienta de visualización de datos. Todo esto se puede resumir en que se busca la creación de un Business Analytics.

El segundo objetivo, relacionado con el Business Analytics, busca la creación de una base de datos con diferentes tipos de información para que, en un futuro, sea una plataforma optimizada para la aplicación de técnicas de IA.

## 1.3. Fases y métodos

La primera fase del proyecto se centró en el desarrollo del código para poder recoger los datos de las diferentes APIs disponibles, siendo esta parte la que más tiempo requirió. Primero se hicieron pruebas de cómo se podrían extraer los datos, y seguidamente se hicieron pruebas de cómo poder guardar los datos para su posterior manipulación.

El siguiente paso fue la construcción de la base de datos, su conexión a Python, y la carga actualizada de los datos en esta base de datos. Esta parte se hizo conjuntamente con la parte de código, de modo que se realizaban pruebas con la base de datos al mismo tiempo que se hacían con la extracción de datos.

La siguiente fase del proyecto fue la que involucró a Power Bi. Hasta que las partes de código y base de datos no estuvieron casi finalizadas, y con una gran certeza de que lo hecho en ellas estaba correcto, no se comenzó a hacer pruebas con el visualizador de datos.

Finalmente, la última fase fue el desarrollo del documento actual, donde se detalló todos los pasos seguidos durante la realización del proyecto.

## 1.4. Estructura de la memoria del TFG

En el Capítulo 2 se detallarán las herramientas empleadas durante la realización de este proyecto, tanto software como APIs, así como los lenguajes y plataformas de programación elegidas para este trabajo.

En el Capítulo 3 se detallará en profundidad el código escrito para este proyecto, indicando de forma meticulosa su funcionamiento. Además, se hablará de la creación de la base de datos y del proceso de automatización de recogida de datos de las APIs.

En el Capítulo 4 se mostrará en detalle el funcionamiento del entorno de visualización de datos desarrollado de los datos recogidos a partir de Microsoft Power Bi [1], la herramienta usada para la visualización de datos. Se indicará todo el proceso seguido, desde la carga de datos, hasta las herramientas elegidas para mostrar los datos proporcionadas por el software.

En el Capítulo 5 se mostrará información sobre la recogida de datos a partir de imágenes satélite, el trabajo que se ha realizado junto con la API Sentinel [1], la problemática encontrada y los resultados obtenidos.

El Capítulo 6 incluye las conclusiones de este Trabajo de Fin de Grado y las líneas futuras a seguir para un futuro desarrollo de este trabajo.

Finalmente se incluyen la Bibliografía y un Anexo, que contiene el código empleado en este proyecto.

# Capítulo 2. Herramientas y metodología

En este capítulo del documento se detallarán las principales herramientas utilizadas para la realización del proyecto. Se describirán tanto los lenguajes de programación utilizados como el software usado para editar el código, las diferentes APIs usadas, la base de datos donde se almacenan los datos y la herramienta usada para su visualización.

Algunos de los elementos aquí descritos pertenecen a Grupo Matarromera, empresa donde se realizaron las prácticas de empresa y quienes solicitaron la realización de este proyecto.

El esquema completo del proyecto es el que se muestra en la Figura 1. El orden de los pasos seguidos va de izquierda a derecha. En primer lugar se recogen los datos de las APIs. El segundo paso es almacenar estos datos, con ayuda de Python, en la base de datos. Finalmente estos datos se muestran en el *data viewer*.

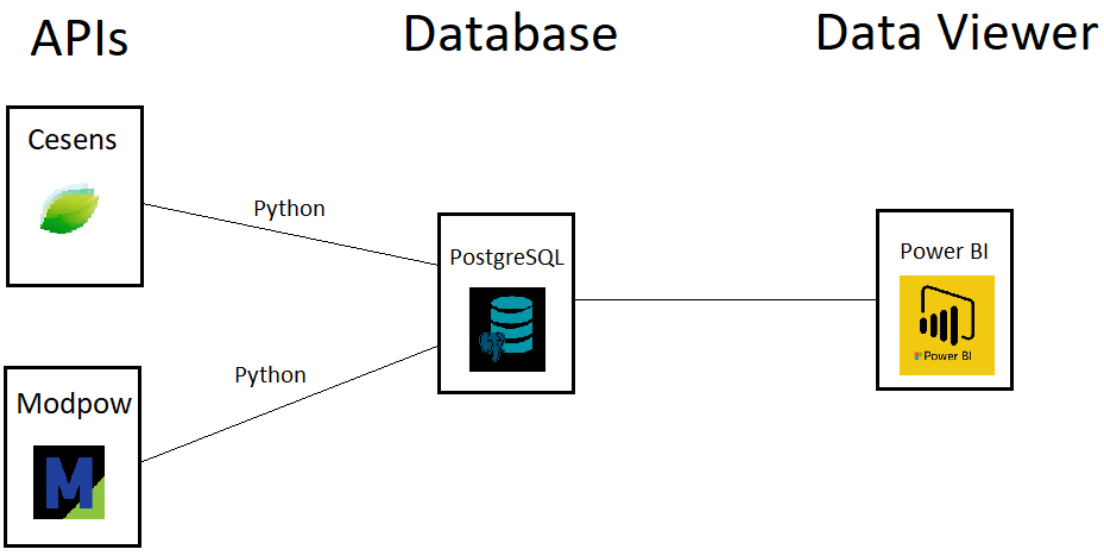


*Figura 1. Diagrama de bloques del proyecto.*

## 2.1. Python

El código escrito para la realización de este proyecto, con el cual se han obtenido todos los datos necesarios, así como almacenar estos, ha sido Python, más concretamente, la versión Python 3.10 [3].

Python es un lenguaje de programación simple y sencillo de utilizar, no necesita de compilación previa para su ejecución, ya que una vez se ejecuta va compilando el código línea a línea. Gracias a la gran cantidad de librerías que posee, así como a la inmensa cantidad de foros de ayuda un la red, la programación en este proyecto fue rápida de realizar.

El software utilizado para la edición del código fuente fue Visual Studio Code [4]. Es un programa de uso simple que gracias a sus constantes actualizaciones siempre se dispone de las librerías de Python al día.

## 2.2. APIs

Una API (*Application Programming Interface*, Interfaz de Programación de Aplicaciones en español) es una pieza de código que permite a diferentes aplicaciones comunicarse entre sí y compartir información y funcionalidades. Una API hace de intermediario entre dos sistemas, en el caso de este proyecto una de las partes eran los sensores colocados en las parcelas de cultivo de Carralancha y Valdecovo, y la otra parte es una interfaz a la que se puede solicitar los datos de la sensórica.

Las APIs utilizadas han sido Cesens y Modpow [5] [6], accediendo a los datos gracias a las credenciales proporcionadas por Grupo Matarromera, y por otro lado, para las imágenes satelitales Sentinel, de acceso público.

### 2.2.1. Cesens

Cesens es una API tanto climática como de suelo, es decir, devuelve datos de temperatura así como del estado del terreno.

Cuenta con más de 50 modelos predictivos de enfermedades para distintos cultivos, así como registros de tratamientos fitosanitarios. También integra imágenes satélite y de drones para obtener datos como el NDVI (*Normalized Difference Vegetation Index*, Índice de Vegetación de Diferencia Normalizada en castellano), necesidad de riego e índice de humedad.

Para este trabajo, se le ha dado a Cesens dos usos distintos. El primero ha sido para la obtención de todos los datos que conciernen a los cultivos, y su segundo uso ha sido para la elaboración de una comparativa de temperaturas con estaciones públicas disponibles en Cesens.

### 2.2.2. Modpow

Modpow es una API que proporciona la solución para la gestión profesional del riego y del fertirriego a través de estudios agronómicos y la información aportada por el software.

El funcionamiento de Modpow se basa en una serie de sensores, o sondas, enterrados en el cultivo a diferentes profundidades. A través de la información que estos recogen, un datalogger la procesa y genera la decisión de se es necesario o no el riego y la fertilización en el terreno.

### 2.2.3. Sentinel

Sentinel es un proyecto multi-satélite desarrollado por la ESA (*European Space Agency*) en el marco del Programa Copérnico. Este programa consta de siete misiones diferentes, cada una con un propósito propio.

## 2.3. PostgreSQL

PostgreSQL es una base de datos de código abierto que usa el lenguaje de programación SQL combinado con muchas características para el almacenamiento seguro de los datos.

A través de su software se han almacenado los datos de las APIs, y gracias a su compatibilidad, se han podido mostrar después a través de Power Bi [7].

## 2.4. Microsoft Power BI

Microsoft Power Bi es una plataforma unificada y escalable de inteligencia empresarial con funciones de autoservicio apta para grandes empresas.

Gracias a sus múltiples conectores se ha podido mostrar los datos desde PostgreSQL, y con su sistema de organización de datos se han podido mostrar de forma ordenada todos los datos recogidos de las APIs.

# Capítulo 3. Automatización y Recogida de Datos de las APIs

En este capítulo de la memoria se explicará la elaboración del código desarrollado en Python a través del cual se han obtenido los datos de las APIs, tanto de Modpow, como de Cesens para las estaciones públicas y privadas de Matarromera.

Se incluirán fragmentos de código, no el código completo. El código al completo se encuentra al final del documento, en el apartado Anexos.

## 3.1. Recogida de datos de la API de Cesens

En este apartado se hablará de Cesens, una API cuyos sensores devuelven datos relacionados con el suelo y también climatológicos. Existen dos tipos de estaciones para esta API, estaciones públicas y estaciones privadas. Para ambas estaciones, la forma de acceso a la misma es la que se muestra a continuación con el nombre y clave correspondientes (se han eliminado por seguridad):

```
log = {
        "nombre": "correo@email.com",
        "clave": "Contraseña"
    }
```

En primer lugar, se debe definir un cuerpo en formato *json* con las credenciales para acceder a los datos, con el *email* y la contraseña con la que se ha registrado en Cesens.

```
login=rqs.post("https://app.cesens.com/api/usuarios/login",data=js.dum
ps(log))
```

Tras este primer paso, haciendo uso de la función *request* y el método POST, se envían estas credenciales. Si todo ha ido bien, el código que devuelve Cesens es el 200, que significa que se ha hecho *login* correctamente, y se crea la cabecera para poder solicitar los datos.

```
if login.status_code==200:
      head='Token' + ' ' + login.json()['auth']
      cabeza_cesens = {
```

```
            "Authentication": head
        }
return cabeza_cesens
```

Ahora que la cabecera está creada, ya se puede empezar a solicitar los datos de las estaciones que se desee. La diferencia entre estaciones públicas y privadas reside en que las estaciones públicas son accesibles a través de cualquier cuenta de Cesens, mientras que las estaciones privadas solo son accesibles si la empresa que ha contratado los servicios de sus sensores habilita servicio a la cuenta que se esté usando.

### 3.1.1. Datos de las estaciones públicas

Las estaciones públicas han sido usadas para obtener datos meteorológicos de lugares cercanos a las tierras de Carralancha y Valdecovo. En concreto, estas ubicaciones han sido Medina del Campo, Olmedo y Rueda. La finalidad de obtener estos datos es poder realizar una comparativa entre los datos de estas estaciones y los datos de la estación climática de Carralancha.

En primer lugar, se ha obtenido y guardado en una tabla las ubicaciones de los sensores de estas ubicaciones para su posterior uso, usando el siguiente código:

```
ubimedina=rqs.get(urlubimedina,headers=cabeza_cesens)
datomedina=js.dumps(ubimedina.json())
infoubimedina=js.loads(datomedina)
ubiolmedo=rqs.get(urlubiolmedo,headers=cabeza_cesens)
datoolmedo=js.dumps(ubiolmedo.json())
infoubiolmedo=js.loads(datoolmedo)
ubirueda=rqs.get(urlubirueda,headers=cabeza_cesens)
datorueda=js.dumps(ubirueda.json())
infoubirueda=js.loads(datorueda)
id=infoubimedina['id']
nombre=infoubimedina['nombre']
descripcion=infoubimedina['descripcion']
latitud=infoubimedina['latitud']
longitud=infoubimedina['longitud']
filas.append([id, nombre, descripcion, latitud, longitud])
id=infoubiolmedo['id']
nombre=infoubiolmedo['nombre']
descripcion=infoubiolmedo['descripcion']
latitud=infoubiolmedo['latitud']
longitud=infoubiolmedo['longitud']
filas.append([id, nombre, descripcion, latitud, longitud])
id=infoubirueda['id']
nombre=infoubirueda['nombre']
descripcion=infoubirueda['descripcion']
latitud=infoubirueda['latitud']
longitud=infoubirueda['longitud']
```

```
filas.append([id, nombre, descripcion, latitud, longitud])
filas.sort()
with open('Ubicaciones_Temperaturas.json','w') as json_file:
      js.dump(filas, json_file, indent=4)
read_file=pd.read_json(r'C:/Users/jaime/Desktop/Database/Ubicaciones_T
emperaturas.json')
read_file.to_csv(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Tempera
turas.csv',index=None,header=headerubis,sep=";",decimal='.',float_form
at='%.5f')
```

Tras esto, se han solicitado los datos de temperatura de las diferentes estaciones. El método a continuación mostrado es solo para la estación de Medina del Campo, pero es igual para las estaciones de Olmedo y Rueda. Lo que se ha hecho ha sido, mediante la función *request* y el método GET, es llamar a la API de Cesens, que ha devuelto en formato *json* los datos solicitados. Tras esto se guardan en formato tabla para hacer el cambio a *csv*.

```
ubimedinacyl = rqs.get(urlmedinacyl, headers=cabeza_cesens)
datamedinacyl = js.dumps(ubimedinacyl.json())
infomedinacyl = js.loads(datamedinacyl)
ubiolmedocyl = rqs.get(urlolmedocyl, headers=cabeza_cesens)
dataolmedocyl = js.dumps(ubiolmedocyl.json())
infoolmedocyl = js.loads(dataolmedocyl)
ubiruedaaemet = rqs.get(urlruedaaemet, headers=cabeza_cesens)
dataruedaaemet = js.dumps(ubiruedaaemet.json())
inforuedaaemet = js.loads(dataruedaaemet)
for itemmedina in infomedinacyl:
      idubicacion=37
      nombremetrica='Temperatura'
      fecha_unix=itemmedina
      fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
      fecha=fecha_normal.strftime('%Y-%m-%d %H:%M:%S')
      valor=infomedinacyl[itemmedina]
      filasmedina.append([idubicacion,nombremetrica,fecha,valor])
if len(filasmedina)!=0:
      with open('Temperatura_Medina_Inforiego.json','w') as json_file:
            js.dump(filasmedina, json_file, indent=4)
      path_json=r'C:/Users/jaime/Desktop/Database/Temperatura_Medina_I
      nforiego.json'
      read_file = pd.read_json(path_json)
      path_csv=r'C:/Users/jaime/Desktop/Database/Temperatura_Medina_In
      foriego.csv'
      read_file.to_csv(path_csv,index=None,header=header,sep=";",decim
      al='.',float_format='%.5f')
```

### 3.1.2. Datos de las estaciones privadas

Las estaciones privadas pertenecen a Grupo Matarromera, y como ya se ha dicho antes, se necesitan permisos especiales para poder acceder a los datos. Las estaciones se ubican en dos parcelas diferentes: Carralancha y Valdecovo. Carralancha posee las

estaciones de: Almendro 1 Pozaldez, Almendro 2 Pozaldez, Olivo 1 Pozaldez, Olivo 2 Pozaldez, Vid Pozaldez y estación Pozaldez clima. Valdecovo posee las estaciones de: Valdecovo 1, Valdecovo 2, Valdecovo 3 y Valdecovo 4. La lista de métricas que devuelven valores es la siguiente:

1. Temperatura
2. Humectación en hoja (porcentaje)
3. Humedad relativa
4. Temperatura en suelo
5. Presión atmosférica
6. Velocidad del viento
7. Dirección del viento
8. Precipitaciones
9. Temperatura interna
10. Batería (voltaje)
11. Integral térmica Winkler (10ºC)
12. Precipitaciones diarias
13. Temperatura media
14. Horas frío (7ºC)
15. Diferencia temperatura día-noche
16. Punto de rocío
17. Radiación solar
18. Moteado (manzano y peral)
19. Racha de viento máxima
20. Contenido volumétrico (conector 1)
21. Contenido volumétrico (conector 2)
22. Contenido volumétrico (conector 3)
23. Mildiu (vid)
24. Oídio (vid)
25. Botrytis (vid)
26. Temperatura máxima
27. Evapotranspiración
28. Radiación solar directa
29. Radiación solar directa neta

30. Agua útil
31. Presión de vapor
32. Insolación diaria
33. Índice heliotérmico
34. Índice de posibilidades heliotérmicas
35. Índice bioclimático de Hidalgo
36. Temperatura media diurna
37. Humedad relativa media diurna
38. Presión atmosférica normalizada
39. Índice de frescor nocturno
40. Temperatura media del ciclo
41. Dendrómetro (conector 4)
42. Índice de sequía
43. Transpiración
44. Precipitaciones acumuladas
45. Conductividad eléctrica aparente (conector 1)
46. Conductividad eléctrica aparente (conector 2)
47. Conductividad eléctrica aparente (conector 3)
48. Tendencia dendrometría (conector 4)
49. Contracción diaria (conector 4)
50. Crecimiento diario (conector 4)
51. Variación de temperatura diaria
52. Humedad relativa interna
53. Mancha negra (peral)
54. Agua útil (porcentaje)
55. Evapotranspiración acumulada
56. Oidiopsis (tomate)
57. Fuego bacteriano (peral)
58. Polilla del racimo (vid)
59. Riego (pulsos)
60. Caudal de riego (cantidad total)
61. Riego diario (cantidad total)
62. Riego acumulado (cantidad total)
63. Cobertura (GSM)

64. Heladas
65. Contenido volumétrico (10 cm)
66. Contenido volumétrico (20 cm)
67. Contenido volumétrico (40 cm)
68. Contenido volumétrico (60 cm)
69. Contenido volumétrico (80 cm)
70. Temperatura en suelo (10 cm)
71. Temperatura en suelo (20 cm)
72. Temperatura en suelo (40 cm)
73. Temperatura en suelo (60 cm)
74. Temperatura en suelo (80 cm)
75. Salinidad (conector 1)
76. Salinidad (conector 2)
77. Salinidad (conector 3)
78. Temperatura en suelo (conector 2)
79. Temperatura en suelo (conector 3)
80. Recomendación de riego
81. Índice de calor
82. Oídio (pimiento)
83. Mancha negra (tomate)
84. Mancha hoja (almendro)
85. NDVI medio
86. Déficit de presión de vapor
87. Balance hídrico
88. Mildiu de la patata
89. Riego (cantidad total)
90. Conductividad eléctrica a 25 °C (conector 1)
91. Conductividad eléctrica a 25 °C (conector 2)
92. Conductividad eléctrica a 25 °C (conector 3)
93. Potencial hídrico (en bruto)
94. Potencial hídrico (almendra)
95. Potencial hídrico (uva)
96. Potencial hídrico (nuez)
97. Tiempo diario de humectación en hoja

98. Piojo rojo California (cítricos)
99. Dollar Spot (césped)

Una vez hecho el *login*, se podrá obtener la cabecera para poder obtener los datos. En primer lugar, se obtendrán las ubicaciones para poder guardarlas en una tabla para su posterior uso. Antes de empezar, se comprueba si el fichero con esta tabla existe, para ahorrar tiempo en la recogida de datos, mediante el siguiente código:

```
if os.path.isfile('Ubicaciones_Cesens.csv')==False:
```

Si no existiera, pedirían los datos con la función *request* y el método GET. Una vez obtenidos, se guardaría en un array para posteriormente guardarlos en un fichero en formato tabla, tal y como se muestra en el siguiente código:

```
ubi=rqs.get("https://app.cesens.com/api/ubicaciones/",headers=cabeza_c
esens)
data1=js.dumps(ubi.json())
info=js.loads(data1)
for registro in info:
      if(nombre1 in registro['nombre'] or nombre2 in
registro['nombre']):
            id=registro['id']
            nombre=registro['nombre']
            descripcion=registro['descripcion']
            latitud=registro['latitud']
            longitud=registro['longitud']
            filas.append([id, nombre, descripcion, latitud, longitud])
filas.sort()
with open('Ubicaciones_Cesens.json','w') as json_file:
      js.dump(filas, json_file, indent=4)
read_file=pd.read_json(r'C:/Users/jaime/Desktop/Database/Ubicaciones_C
esens.json')
read_file.to_csv(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Cesens.
csv',index=None,header=header,sep=";",decimal='.',float_format='%.5f')
```

Lo siguiente en hacer es obtener el listado completo de las métricas. Al igual que para obtener las ubicaciones, se comprueba si el fichero que contiene la lista de métricas ya existe:

```
if os.path.isfile('Metricas_Cesens.csv')==False:
```

Si no existiera, haría uso de la cabecera de Cesens previamente creada para solicitar los datos de la API haciendo uso de *request* y GET. Se usa un bucle para obtener las métricas ya que estas están numeradas del 1 al 352, bien es cierto que faltan bastantes entre medias, pero es un proceso que se hace rápido y que no satura los servidores a los que se llama. Una vez hecho esto, se guardan los datos para poder almacenarlos en un fichero con formato tabla:

```
for x in range (1,353):
      url="https://app.cesens.com/api/metricas/"+repr(x)
      ubi=rqs.get(url, headers=cabeza_cesens)
      data1=js.dumps(ubi.json())
      info=js.loads(data1)
      if(error1 not in info and info!=error2 and info!=error3):
            id=info['id']
            nombre=info['nombre'].replace(" ","")
            acronimo=info['acronimo']
            unidad=info['unidad']
            metbien=info['nombre']
            filas.append([id, nombre.lower(),acronimo,unidad,metbien])
filas.sort()
with open('Metricas_Cesens.json','w') as json_file:
      js.dump(filas, json_file, indent=4)
read_file=pd.read_json(r'C:/Users/jaime/Desktop/Database/Metricas_Cese
ns.json')
read_file.to_csv(r'C:/Users/jaime/Desktop/Database/Metricas_Cesens.csv
',index=None,header=header,sep=";",decimal='.',float_format='%.5f')
```

Una vez obtenidas las litas de ubicaciones y métricas, se obtienen los datos. Para obtener los datos se hacen llamadas a cada métrica por cada ubicación existente. Primero, para poder obtener los datos, es necesario volver a obtener las ubicaciones mediante el siguiente código:

```
ubi=rqs.get("https://app.cesens.com/api/ubicaciones/",headers=cabeza_c
esens)
data1=js.dumps(ubi.json())
info=js.loads(data1)
```

Una vez obtenidas las ubicaciones, se hace uso del bucle visto anteriormente para poder obtener las diferentes ubicaciones. A la hora de obtener las métricas, existen dos que causan fallo y que no se ha podido resolver, ya que es problema de Cesens, y son las métricas 902 y 163, de ahí que exista un filtro para esas dos métricas. Una vez empezado el bucle, se hace la llamada con *request* y GET para obtener los datos. Las fechas de estos datos van desde principios de 2022 hasta el día de hoy. Tras las llamadas para obtener los datos, se guardan en un array y se guardan posteriormente n un fichero con formato tabla, proceso completo que se muestra en el siguiente código:

```
for x in range (1,353):
      if(x!=90 and x!=163):
            new_url2 = 'https://app.cesens.com/api/metricas/' + repr(x)
            metricas=rqs.get(new_url2, headers=cabeza_cesens)
            data3=js.dumps(metricas.json())
            info3=js.loads(data3)
            if(error1 not in info3 and info3!=error2 and
info3!=error3):
                  nombremetrica=info3['nombre'].replace(" ","")
                  new_archivo = nombremetrica.lower()+ '.json'
                  new_archivo_csv = nombremetrica.lower() + '.csv'
                  for item in info:
```

```
                        if(nombre1 in item['nombre'] or nombre2 in
item['nombre']):
                        new_url1='https://app.cesens.com/api/datos/' +
repr(item['id']) + '/' + repr(x)
                         datos1=rqs.get(new_url1,headers=cabeza_cesens)
                         data2=js.dumps(datos1.json())
                         info2=js.loads(data2)
                        if(error1 not in info2 and info2!=error2 and
info2!=error3):
                              for item4 in info2:
                                    if(info2[item4] != None):
                                          idubi=repr(item['id'])
                                          nombremet=nombremetrica

                                          fecha_unix=item4
                                          fecha_normal=dt.datetime.from
                                          timestamp(int(fecha_unix))

      fecha=fecha_normal.strftime('%Y-%m-%d %H:%M:%S')
                                          valor=info2[item4]

                                          filas.append([idubi,nombremet
                                          .lower(),fecha,valor])
            if len(filas)!=0:
                  with open(new_archivo,'w') as json_file:
                        js.dump(filas, json_file, indent=4)
                  path_json=r'C:/Users/jaime/Desktop/Database/'+new_arc
            hivo
                   read_file = pd.read_json(path_json)
                   path_csv=r'C:/Users/jaime/Desktop/Database/'+new_arc
                   hivo_csv
                   read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
```

En resumen, para obtener los datos de todos los parámetros de la API Cesens, tanto en estaciones privadas como públicas, primero se obtiene la cabecera tras el *login*, para después obtener los datos de las diferentes ubicaciones, las diferentes métricas, y finalmente los datos para usarlos finalmente en el *business analytics*.

## 3.2. Recogida de datos de la API de Modpow

En este apartado se detallará el código empleado para obtener los datos de la API de Modpow, una API cuyos sensores extraen datos del suelo, tales como minerales y nutrientes. Lo primero que se hace es iniciar sesión en servidor. Es necesario para ello tener permisos otorgados por la empresa que contrate los servicios de Modpow, es este caso Grupo Matarromera. Primero de todo se deben indicar las credenciales de inicio de sesión.

```
log = {
      "login":"correo@email.com",
      "password":"Contraseña",
      "device":{"vendor":"API","platform":"API","type":"API"}
}
```

Una vez introducidas las credenciales, con el método POST de la función *request* se mandan, y si todo ha ido bien, se recibe el código 200, tal y como se muestra a continuación:

```
login=rqs.post("https://apiv2.wireless-
monitoring.net/ioecrops/extra/login", data=js.dumps(log))
```

Como ya se ha mencionado, si la respuesta obtenida es el código 200, se procede a la creación de la cabecera, que será utilizada para poder obtener el resto de los datos de Modpow:

```
if login.status_code==200:
      head=login.json()['token']
      cabeza_modpow = {
            "Authorization": head
}
```

Una vez iniciada sesión correctamente, ya se pueden solicitar los datos que se precisen. En primer lugar se obtienen las ubicaciones que se usarán, seguidamente se solicitan las diferentes métricas que recogen los sensores, y finalmente los datos de estas métricas en cada una de las ubicaciones recibidas.

En el caso de las ubicaciones, estas se dividen en función de la parcela. Para Carralancha se tienen las ubicaciones de Almendros, Olivos y Viñedo. Para Valdecovo, las ubicaciones son Arbequina, Picual Pinar y Picual Castillo. Lo primero que se hace en el código es comprobar si el fichero que contiene la tabla de ubicaciones existe, de modo que, si ya está creado, no se vuelven a solicitar las ubicaciones para ahorrar tiempo y no saturar los servidores de Modpow:

```
if os.path.isfile('Ubicaciones_Modpow.csv')==False:
```

Tras este primer paso, gracias a la función *request* y el método GET, se obtienen los datos de las diferentes ubicaciones, guardando estas en un array para posteriormente guardar los datos en un fichero en formato tabla, tal como se muestra a continuación:

```
ubi=rqs.get("https://apiv2.wireless-
monitoring.net/ioecrops/devices",headers=cabeza_modpow)
data1=js.dumps(ubi.json())
info=js.loads(data1)
for item in info['devices'] :
```

```
      newurl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']
      ubi2=rqs.get(newurl,headers=cabeza_modpow)
      data2=js.dumps(ubi2.json())
      info2=js.loads(data2)
      for item2 in info2['devices']:
            id=item2['id']
            nombre=item2['deployment']['label']
            descripcion=item2['label']
            latitud=item2['location']['latitude']
            longitud=item2['location']['longitude']

            filas.append([int(id),nombre,descripcion,latitud,longitud])
filas.sort()
with open('Ubicaciones_Modpow.json','w') as json_file:
      js.dump(filas, json_file, indent=4)
read_file=pd.read_json(r'C:/Users/jaime/Desktop/Database/Ubicaciones_M
odpow.json')
read_file.to_csv(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Modpow.
csv',index=None,header=header,sep=";",decimal='.',float_format='%.5f')
```

Una vez obtenidas las ubicaciones, se procede a obtener las diferentes métricas. Las métricas que devuelven datos son las siguientes:

1. Battery Level
2. Liquid Flow (Liters/5min)
3. Watering Flow (Liters/5min)
4. Soil Moisture - 20cm
5. Watering EC
6. Soil Moisture - 40cm
7. Watering Temperature
8. Soil Moisture - 60cm
9. Soil Moisture – Roots
10. Soil Moisture - Roots 25cm
11. Soil EC – Roots
12. Soil Temp. – Roots
13. Soil Moisture – Drainage
14. Soil Moisture - Drainage 50cm
15. 25cm - Sol. EC
16. 30cm - Sol. EC
17. 25cm - Sol. pH
18. 30cm - Sol. pH
19. 25cm - Sol. N

20. 30cm - Sol. N
21. 25cm - Sol. P
22. 30cm - Sol. P
23. 25cm - Sol. K
24. 30cm - Sol. K
25. 25cm - Sol. Ca
26. 30cm - Sol. Ca
27. 25cm - Sol. Na
28. 30cm - Sol. Na
29. 50cm - Sol. EC
30. 60cm - Sol. EC
31. 50cm - Sol. pH
32. 60cm - Sol. pH
33. 50cm - Sol. N
34. 60cm - Sol. N
35. 50cm - Sol. P
36. 60cm - Sol. P
37. 50cm - Sol. K
38. 60cm - Sol. K
39. 50cm - Sol. Ca
40. 60cm - Sol. Ca
41. 50cm - Sol. Na
42. 60cm - Sol. Na
43. Water – EC
44. Water – pH
45. Water – Na
46. 1 Branch Growth
47. 2 Branch Growth
48. 3 Branch Growth
49. 4 Branch Growth
50. 1 Fruit Growth
51. 2 Fruit Growth
52. 3 Fruit Growth
53. 4 Fruit Growth

Para obtener estas métricas, al igual que para obtener las ubicaciones, se hace uso de la función *request* y el método GET. Previo paso a solicitar los datos se comprueba la existencia del fichero que contiene las métricas, si existe no se solicitan. En caso de que sea necesario recoger las métricas, tras la llamada para obtenerlas, se almacenan los datos en un array para después guardarlos en un archivo en formato tabla, tal y como se muestra en el fragmento de código siguiente:

```
if os.path.isfile('Metricas_Modpow.csv')==False:
      met=rqs.get("https://apiv2.wireless-
      monitoring.net/ioecrops/devices",headers=cabeza_modpow)
      data1=js.dumps(met.json())
      info=js.loads(data1)
      for item in info['devices']:
            newurl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"
            met2=rqs.get(newurl,headers=cabeza_modpow)
            data2=js.dumps(met2.json())
            info2=js.loads(data2)
            for item2 in info2['datastreams']:
                  newurl2=newurl+'/'+repr(item2['id'])
                  met3=rqs.get(newurl2,headers=cabeza_modpow)
                  data3=js.dumps(met3.json())
                  info3=js.loads(data3)
                  for item3 in info3['datastreams']:
                        id=repr(item3['id'])
                        nombre=item3['label'].replace(" ","")
                        propiedades=item3['property']
                        unidad=item3['units']
                        nombrebien=item3['label']
                        filas.append([int(id),nombre.lower(),propiedade
                        s,unidad,nombrebien])

      with open('Metricas_Modpow.json','w') as json_file:
            js.dump(filas2, json_file, indent=4)
      read_file=pd.read_json(r'C:/Users/jaime/Desktop/Database/Metrica
      s_Modpow.json')
      read_file.to_csv(r'C:/Users/jaime/Desktop/Database/Metricas_Modp
      ow.csv', index = None,
      header=header,sep=";",decimal='.',float_format='%.5f')
```

Tras la obtención de las ubicaciones y las métricas, se deben obtener los datos relativos a estas. En este caso, el código presenta dos variantes en función de si ya hay ficheros creados con los datos o si todavía no hay nada guardado.

Si no existen datos guardados, se solicitan estos desde el 01/04/2022 hasta el día presente. Modpow presenta un problema, y es que solo devuelve datos de una semana como máximo, es decir, si se solicitan datos desde la fecha indicada hasta hoy, solo nos devolverá los datos de la última semana. Para solventar este problema, se ha hecho uso de bucles en los que las fechas van cambiando para poder recoger todos los datos posibles,

y de esta forma tener una mejor base de datos, más completa. Esto provoca problemas y saturación en los servidores de Modpow, de ahí que existan dos formas de recoger estos datos. Tras esta ejecución que puede durar incluso más de cuatro horas, se guarda en un archivo txt la fecha de ejecución en formato UNIX, dando lugar así a la segunda forma de recoger los datos. Esta primera forma de ejecución solo se lleva a cabo una única vez, cuando la base de datos está vacía y no hay dato alguno, y por consiguiente no existe el fichero que contiene la fecha de la última ejecución.

La segunda forma de recoger los datos consiste comprobar la existencia del fichero donde se ha guardado la fecha de la última ejecución. Si existe, se abre y se lee el contenido, que será la fecha en formato UNIX (un número de diez cifras). El resto del proceso de obtención de datos es similar al descrito en el anterior párrafo, pero en lugar de solicitar los datos desde el 01/04/2022, se solicitan desde la fecha contenida en el txt.

Tras esto, en ambos casos, los datos se guardan en un array para después ser contenidos en archivos *csv* en formato tabla. No se ha mostrado ningún fragmento de código debido a la longitud que tiene en esta parte del proyecto, y al igual que se dijo al inicio del capítulo, todo el código está disponible en los Anexos, y en particular, esta parte del código estará en el apartado de código de datos de Modpow.

## 3.3. Diseño de base de datos

El software utilizado para almacenar los datos recogidos de las APIs es PostgreSQL. Para poder utilizarlo, en primer lugar, después de instalar el software, es crear la base de datos. Para ello, basta con hacer *click* derecho donde se indica en la Figura 2, y elegir el nombre en el cuadro de texto de la Figura 3.

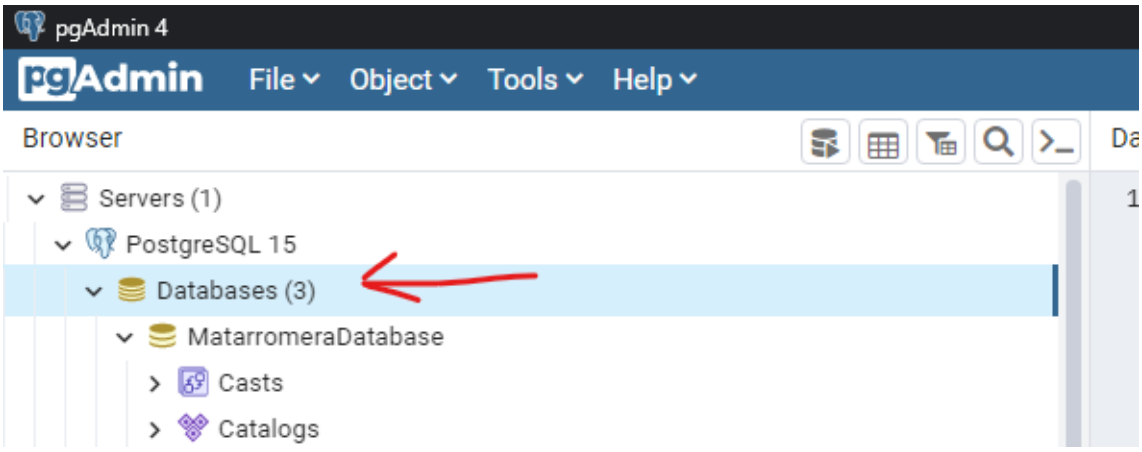


*Figura 2. Lugar donde hacer click derecho para crear la base de datos*

.

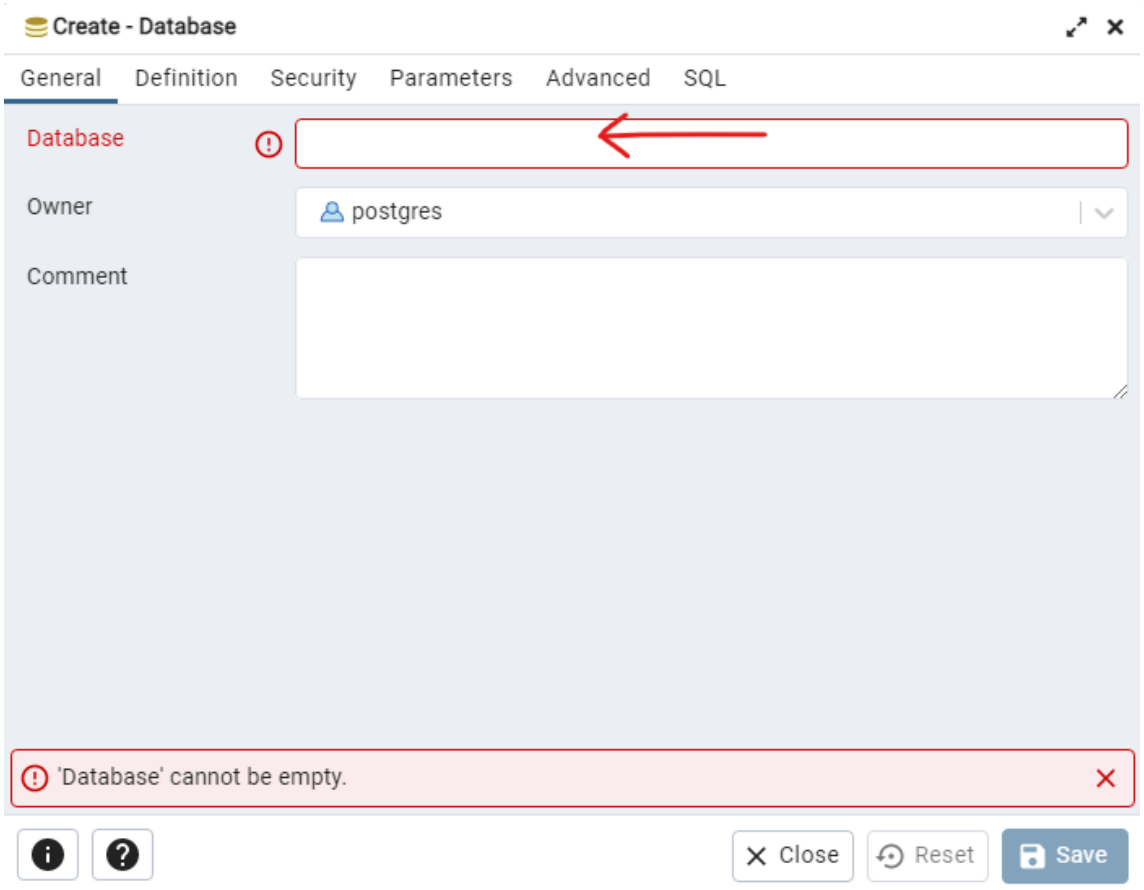


*Figura 3. Lugar donde escribir el nombre de la base de datos.*

Tras realizar estos dos pasos, la base de datos ya estaría creada. El siguiente paso es crear el acceso a la base de datos a través de Python. Para ello, en el código se deben

indicar las credenciales de la base de datos, y después, con la función *connection*, la conexión estaría realizada.

```
try:
      connection=psql.connect(
            host=host,
            user=user,
            password=password,
            database=nombredatabase
      )
      connection.autocommit=True
      return(connection)
except Exception as ex:
      print(ex)
```

Para poder guardar los datos en la base de datos, es necesario, mediante código SQL, crear sentencias para conectar a la base de datos, leer los archivos y copiar y pegar la información a PostgreSQL. Estas sentencias son similares para todos los archivos *csv*. Para ello, en primer lugar se debe crear la conexión a la base de datos mediante la siguiente sentencia:

```
cursor=conexion.cursor()
```

Después, se debe crear la sentencia SQL. Esta sentencia lleva incluida una sentencia *if* para comprobar que la tabla no exista, de forma que si no existe se crea la tabla:

```
tubicacion="CREATE TABLE IF NOT EXISTS ubicacion_cesens(id INT PRIMARY
KEY,nombre VARCHAR(50),descripcion VARCHAR(50),latitud FLOAT,longitud
FLOAT)"
cursor.execute(tubicacion)
```

Una vez creada la tabla, se debe abrir el archivo para poder copiar su contenido en la tabla, y posteriormente, subirla a la base de datos de PostgreSQL, tal y como se muestra a continuación:

```
with
open('C:/Users/Jaime/Desktop/Database/Ubicaciones_Cesens.csv','r',enco
ding="utf8") as f:
      next(f)
      cursor.copy_from(f,'ubicacion_cesens',sep=';')
```

Finalmente, se debe cerra la conexión con la base de datos. Si todo ha ido bien, la información y las tablas estarán guardadas en la base de datos. Para comprobar esto, se puede hacer *click* derecho en *Tables*, dentro de la base de datos, y pulsar en *Refresh*, como se muestra en la Figura 4.

```
cursor.close()
```

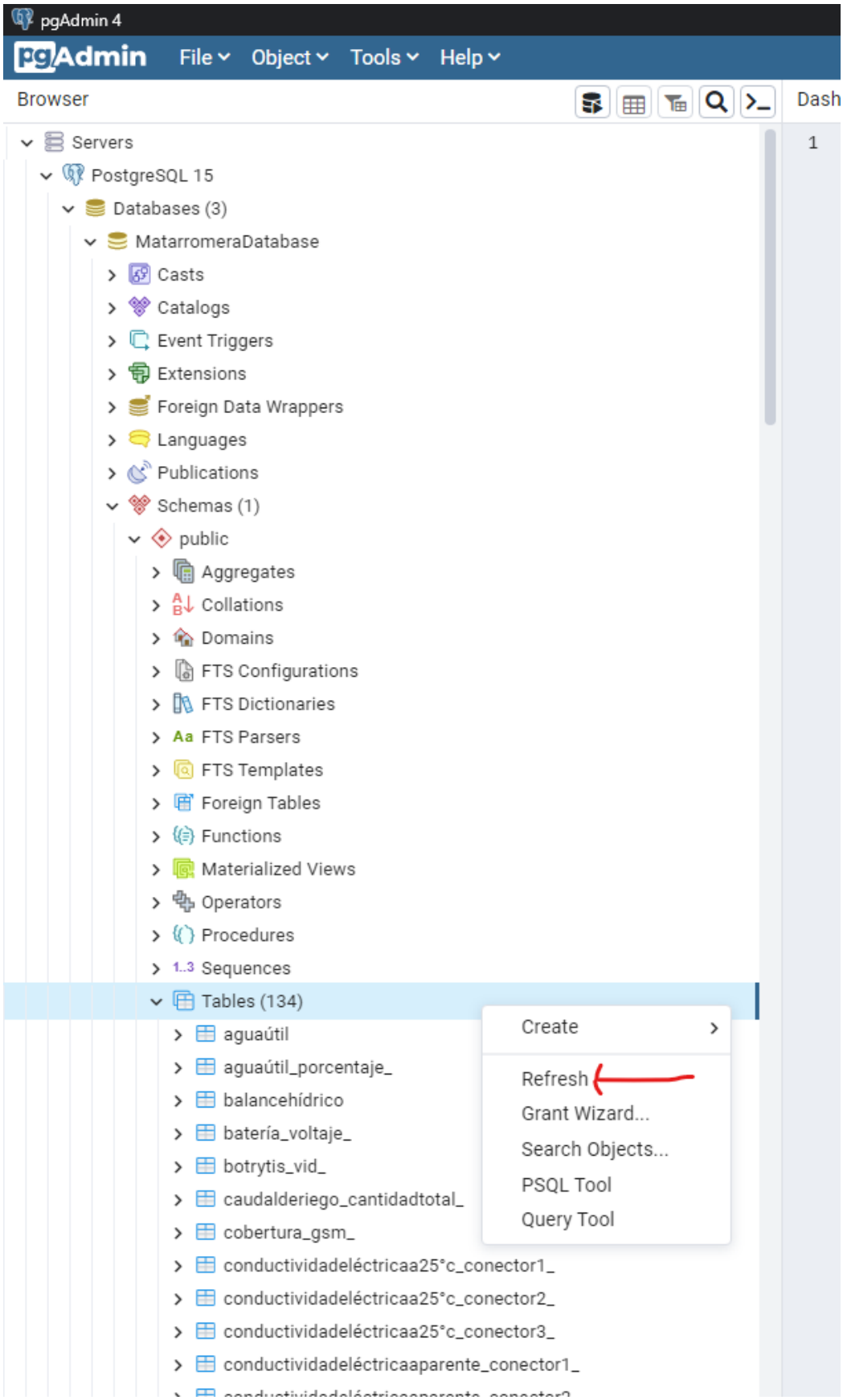


*Figura 4. Botón Refresh en PostgreSQL.*

Finalmente, como se puede ver en la Figura 5, en la Figura 6, y en la Figura 7, están los diagramas de tablas y las relaciones entre sí. En la Figura 5, la relativa a Cesens, se tienen dos *Primary Keys*, una en la tabla de ubicaciones y otra en la tabla de métricas. También hay dos *Foreign Keys*, dos en metubis (tabla que contiene las métricas relacionadas a cada ubicación) y dos en datos cesens, que son las usadas para relacionar estas tablas con las tablas de ubicaciones y métricas. En la Figura 6, la perteneciente a Modpow, se ven dos *Primary Keys*, pertenecientes a las tablas de ubicaciones y métricas, y dos *Foreign Keys*, ambas en la tabla de datos de Modpow. Finalmente, en la Figura 7, la de las temperaturas, no hay ni *Primary Keys* ni *Foreign Keys*.

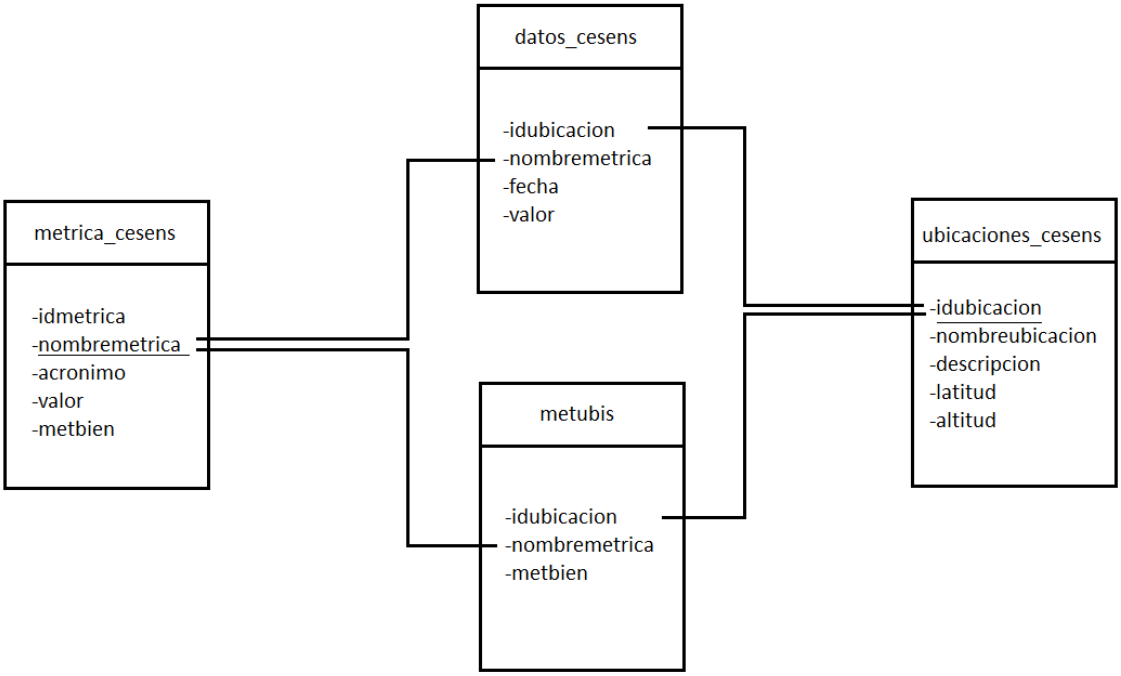


*Figura 5. Diagrama de tablas de Cesens*

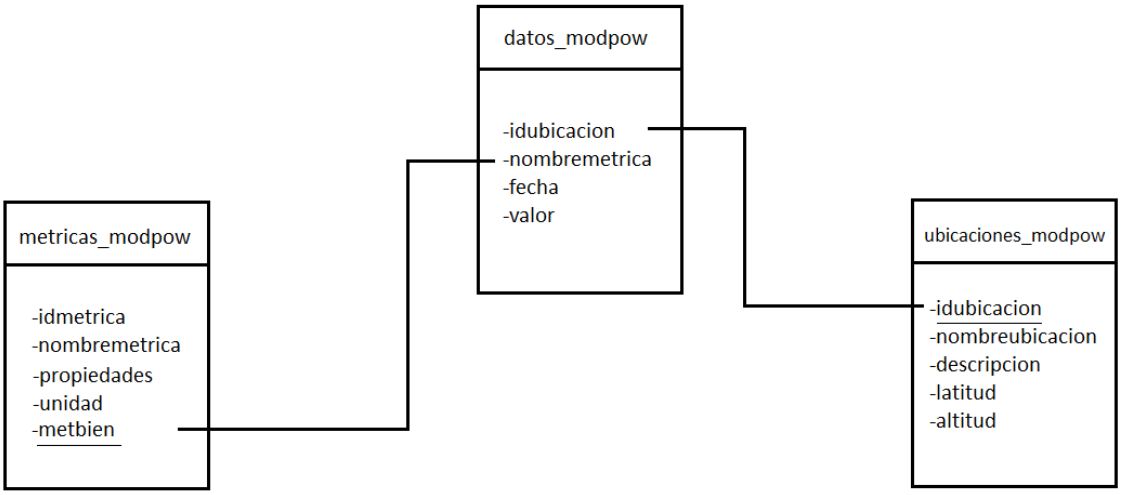


*Figura 6. Diagrama de tablas de Modpow*

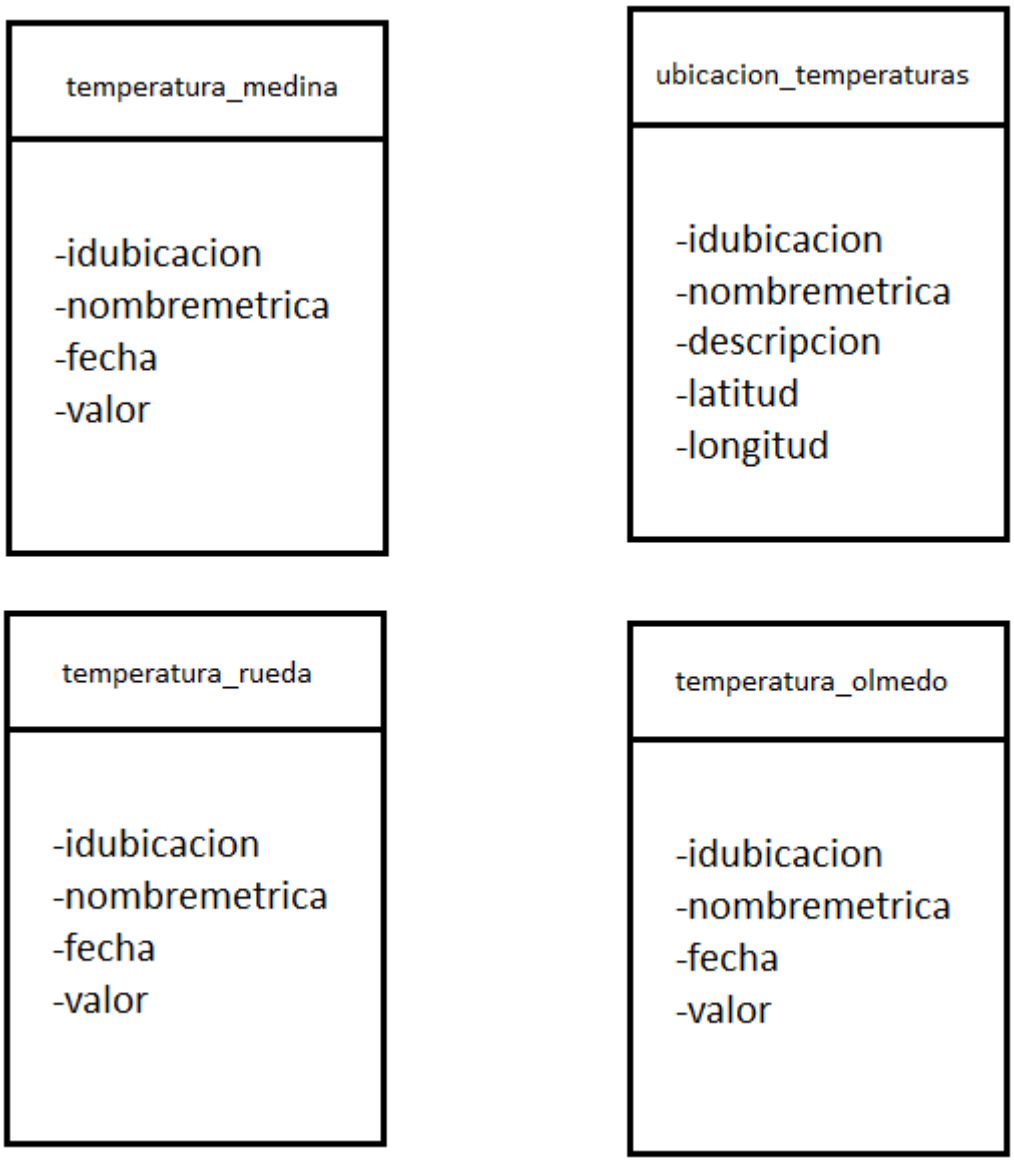


*Figura 7. Diagrama de tablas de Temperaturas*

## 3.4. Automatización de la recogida de datos

Para poder automatizar todo el proceso de recogida de datos, de forma que se pueda programar en qué momento del día quiere realizarse, se ha usado un archivo bat. En este

archivo simplemente se ha escrito la ruta del Ejecutable de Python y la ruta del archivo principal del proyecto, de forma que cada vez que se ejecute se recojan los datos.

Por si solo el archivo bat no puede iniciarse automáticamente. Es necesario usar el programador de tareas de Windows. Para ello, haciendo *click* en el botón de Windows, se escribe Programador de Tareas, abriéndose la ventana mostrada en la Figura 8.

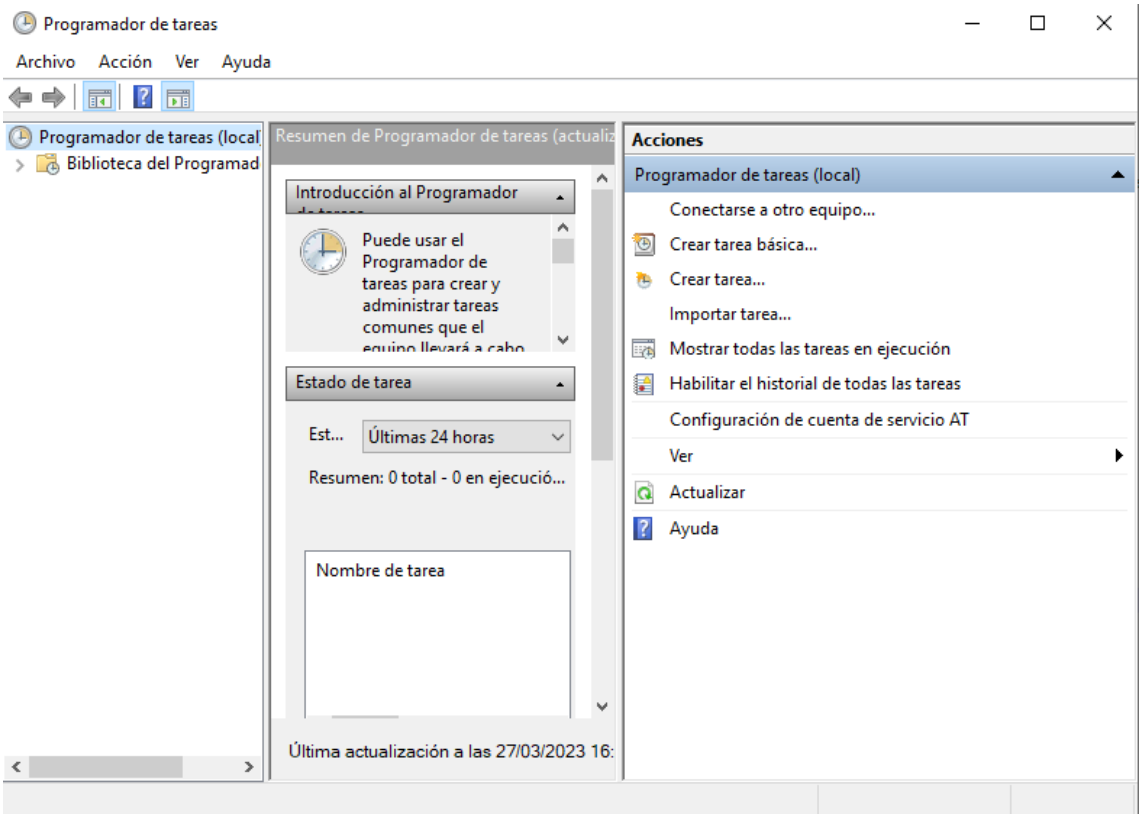


*Figura 8. Programador de tareas de Windows.*

Para crear la tarea con el archivo bat, es necesario clickar en Acción, y después en Crear Tarea, como se ve en la Figura 9.

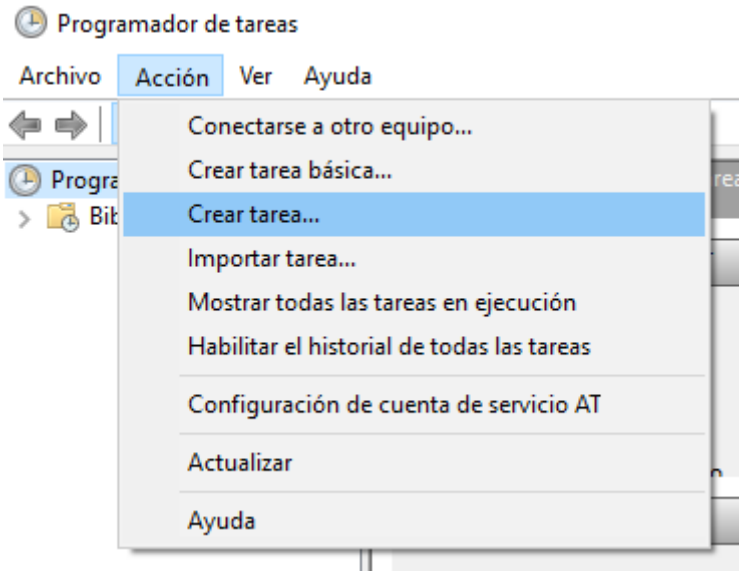


*Figura 9. Crear tarea.*

Tras esto, se abrirá la ventana mostrada en la Figura 10, donde debemos indicar el nombre de la tarea y una descripción de esta.

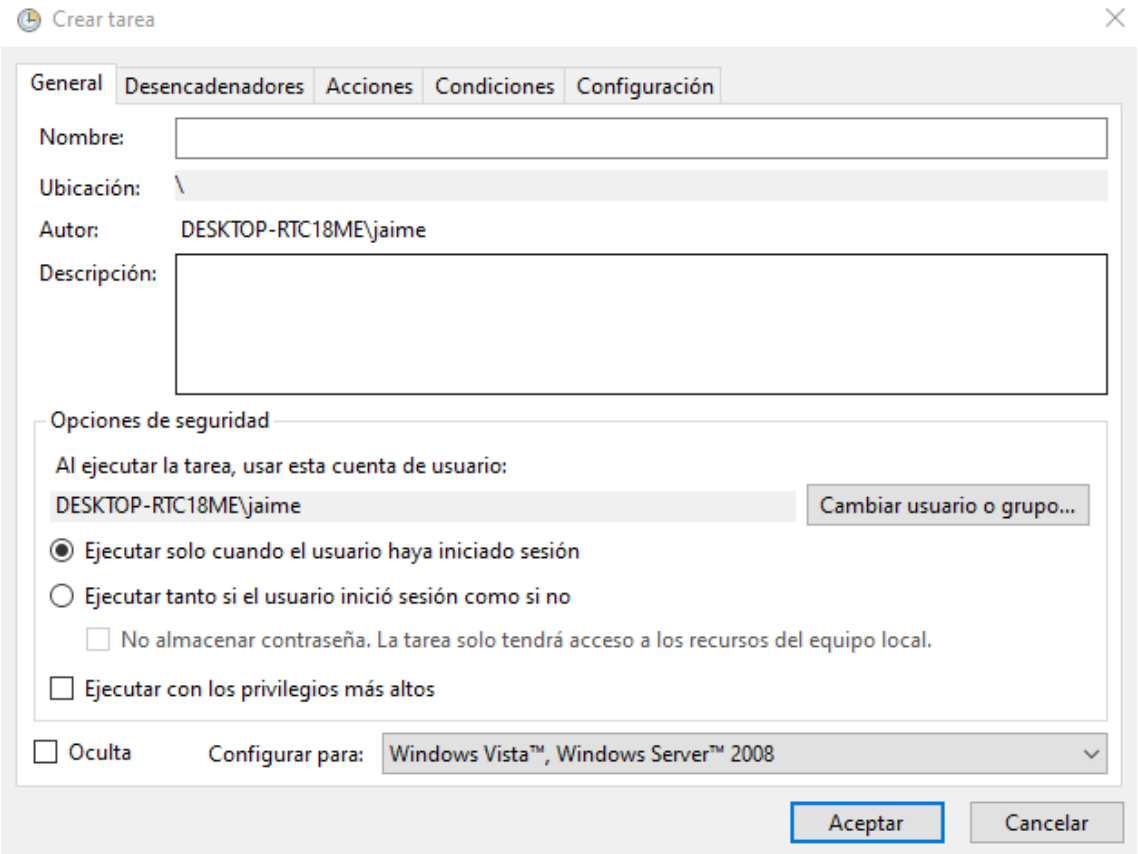


*Figura 10. Nombre y descripción de la tarea a crear.*

Después, se selecciona la pestaña Desencadenadores y Nuevo, abriéndose la ventana de la Figura 11. Aquí es donde se debe elegir la frecuencia de repetición de la tarea, que puede ir desde una sola vez, a diaria o mensualmente.

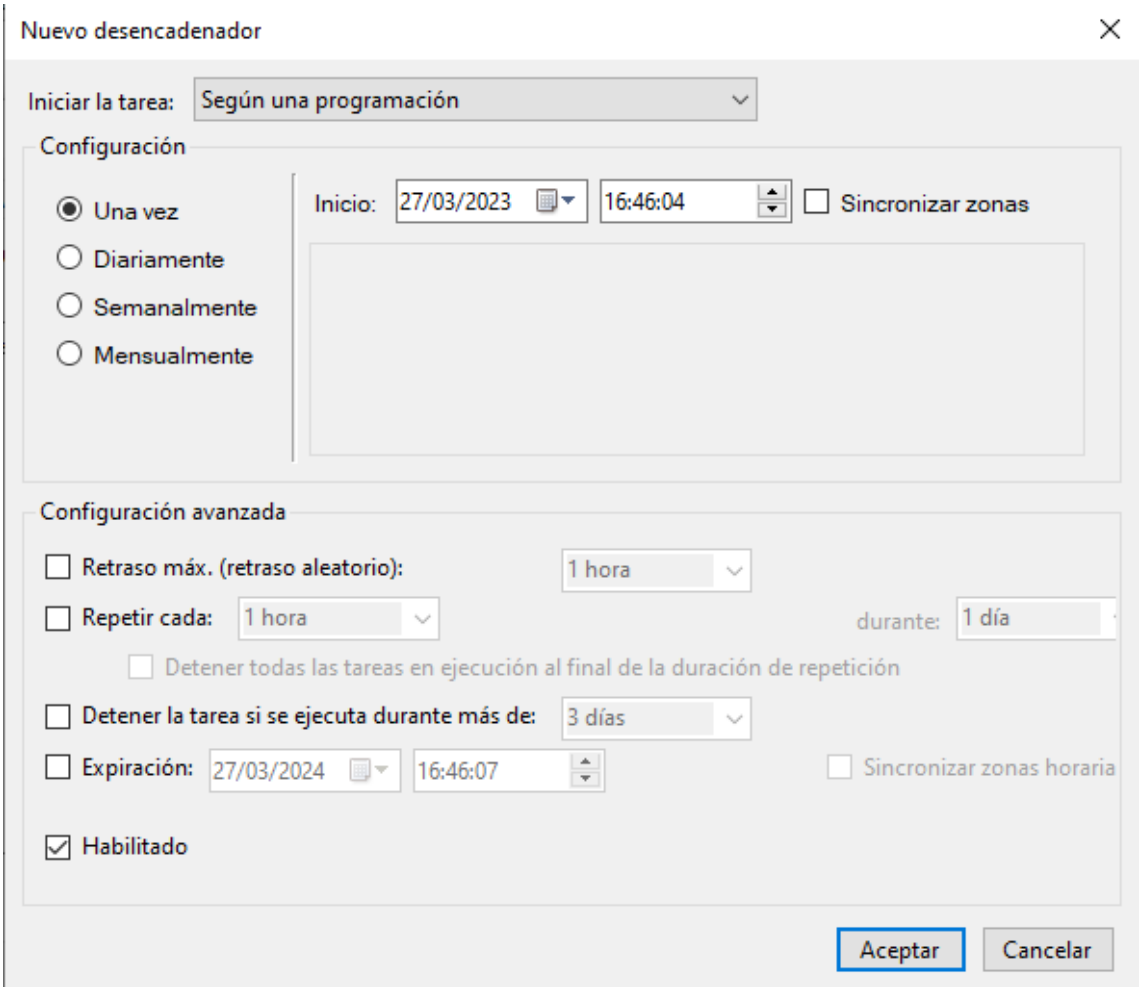


*Figura 11. Programador de la tarea.*

Volviendo a la Figura 10, si ahora se selecciona Acciones y Nuevo, se puede escribir la ruta del archivo bat en Programa o script, tal como se aprecia en la Figura 12.

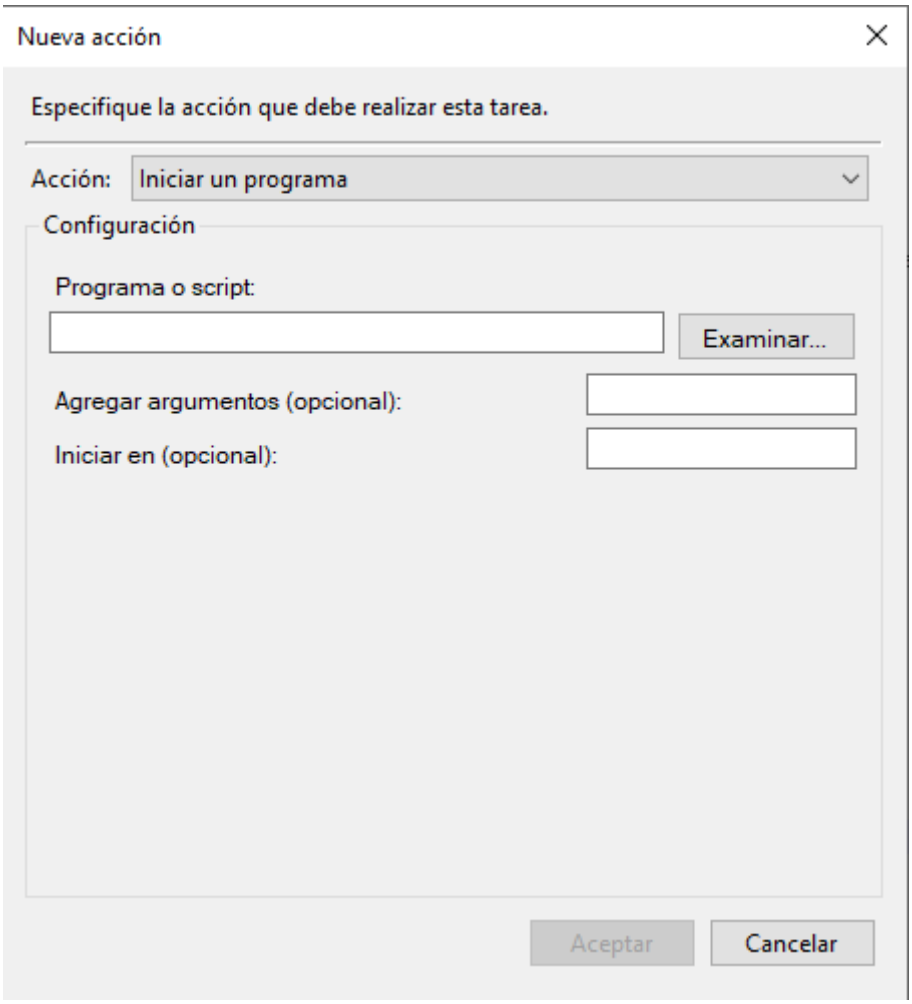


*Figura 12. Seleccionar la ruta del archivo bat.*

Siguiendo estos pasos, se puede automatizar la recogida de datos, de forma que se puede mantener siempre actualizada la base de datos sin tener que hacerlo manualmente.

# Capítulo 4. Visualización de datos mediante interfaz de Power Bi

En este apartado se detallará el proceso mediante el cual se muestran los datos a través del diseño de un panel en Microsoft Power Bi, desde la creación del modelo, hasta la conexión e inclusión de las tablas desde la base de datos.

Para el desarrollo de esta parte del proyecto, se dividió el modelo de Power Bi en tres esquemas bien diferenciados para recoger los datos de las diferentes APIs: Cesens, Modpow y Temperaturas.

## 4.1. Microsoft Power BI

Microsoft Power Bi es un software descargable desde Microsoft Store. Requiere de un inicio de sesión con credenciales válidas de Microsoft para poder ser usado.

Para poder obtener los datos, es necesario conectarse a la base de datos previamente creada y en la cual deben de estar los datos que se quieran representar. Para ello basta con seleccionar en el botón Obtener Datos de la Figura 13, después en Más como se indica en la Figura 14, y finalmente en Base de Datos; se selecciona Base de datos PostgreSQL como se ve en la Figura 15, que es la base de datos elegida para este proyecto, y finalmente se hace *click* en conectar.

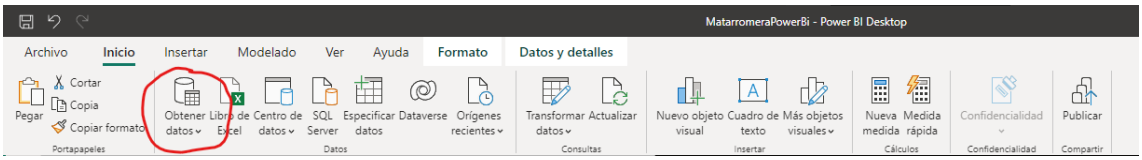


*Figura 13. Panel de opciones de Microsoft Power Bi.*

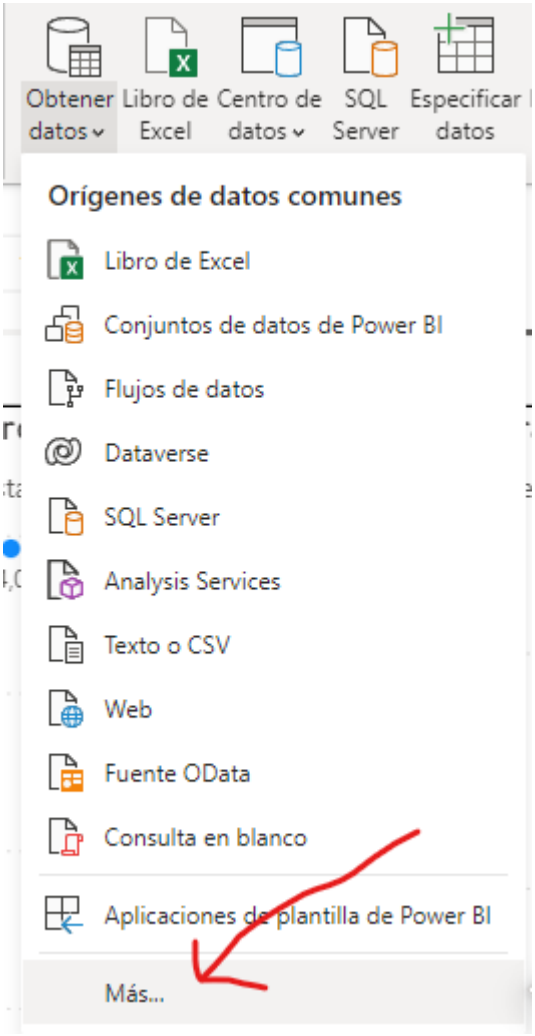


*Figura 14. Opciones para obtener datos.*

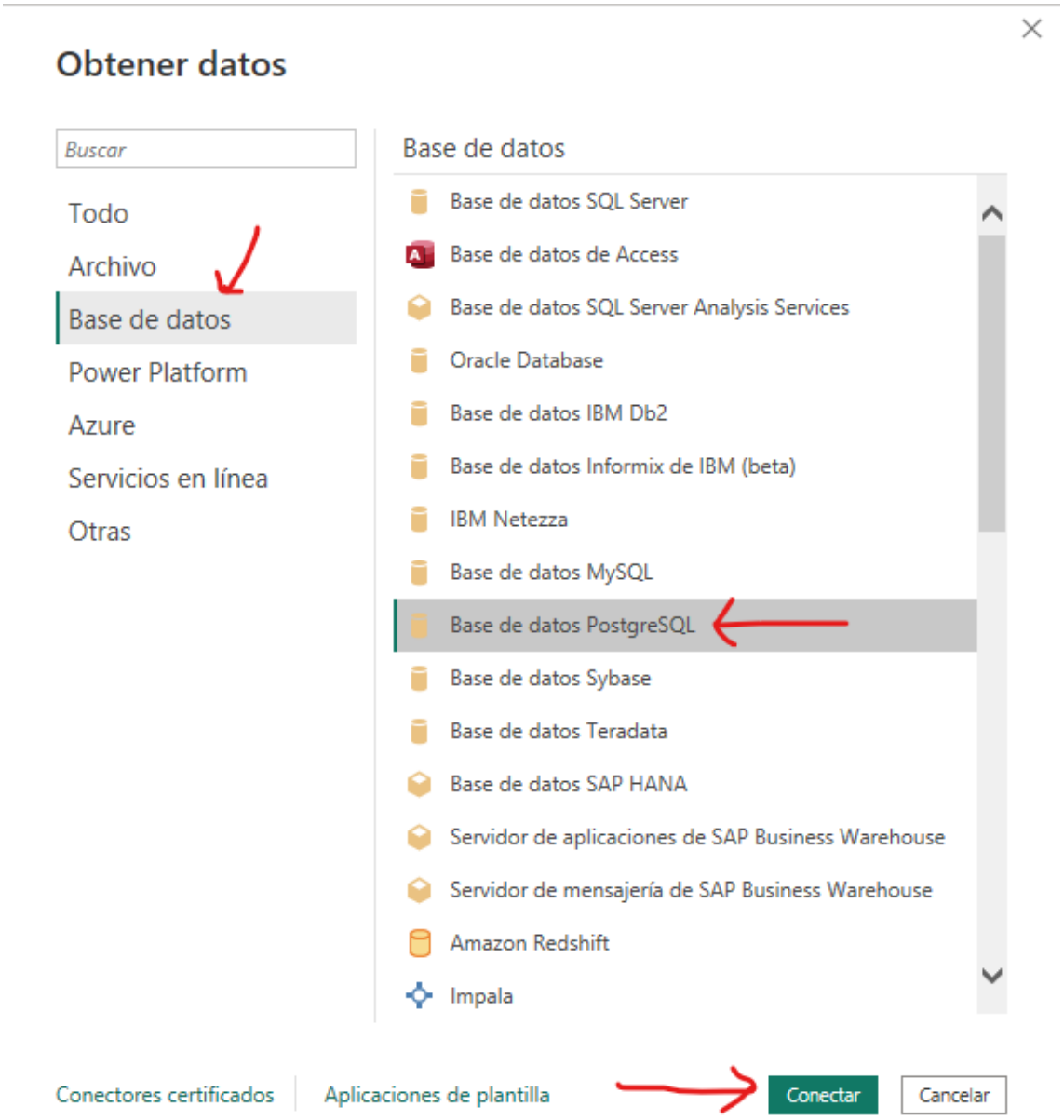


*Figura 15. Configuración usada para el proyecto.*

Tras esto, Microsoft Power Bi pedirá las credenciales de la base de datos de PostgreSQL, teniendo que indicar el Servidor y el nombre de la Base de datos, tal como se observa en la Figura 16. El Modo Conectividad de datos se deja en Importar y no se modifican ninguna de las opciones avanzadas. Tras esto se selecciona aceptar y se mostrará una nueva ventana donde se pedirán el Nombre de usuario de PostgreSQL y la Contraseña usada para la base de datos como se muestra en la Figura 17.


*Figura 16. Ventana para indicar la base de datos y el servidor a usar.*

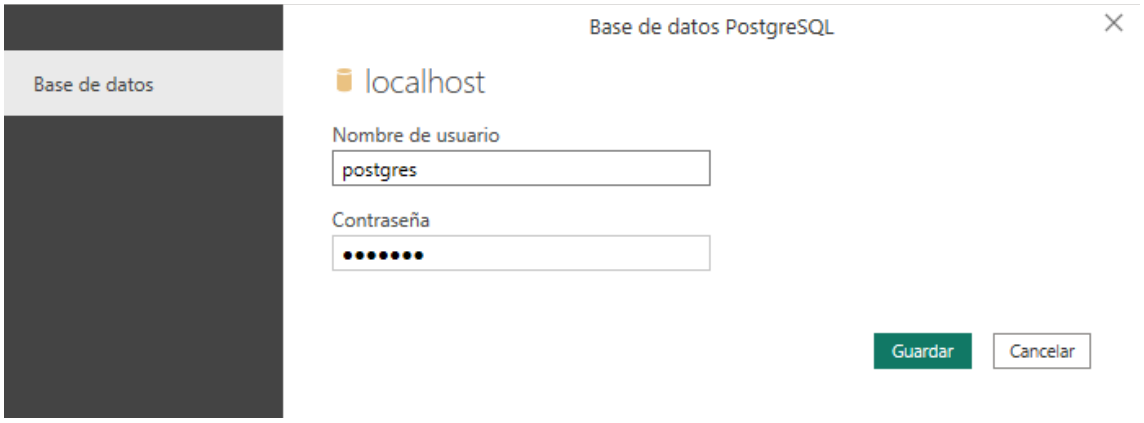


*Figura 17. Ventana para indicar el nombre de usuario y la contraseña de PostgreSQL.*

Tras esto, se abrirá la última ventana antes de poder empezar a trabajar con Power Bi. Esta ventana pedirá al usuario que seleccione todas las tablas contenidas en la base de datos indicada con las que se quiera trabajar, tal como se muestra en la Figura 18. Una vez seleccionadas las tablas, se da en la opción de Cargar, y estas tablas se cargarán en el modelo de Power Bi y ya se podrá empezar a diseñar de forma gráfica el modelo de visualizador de datos.

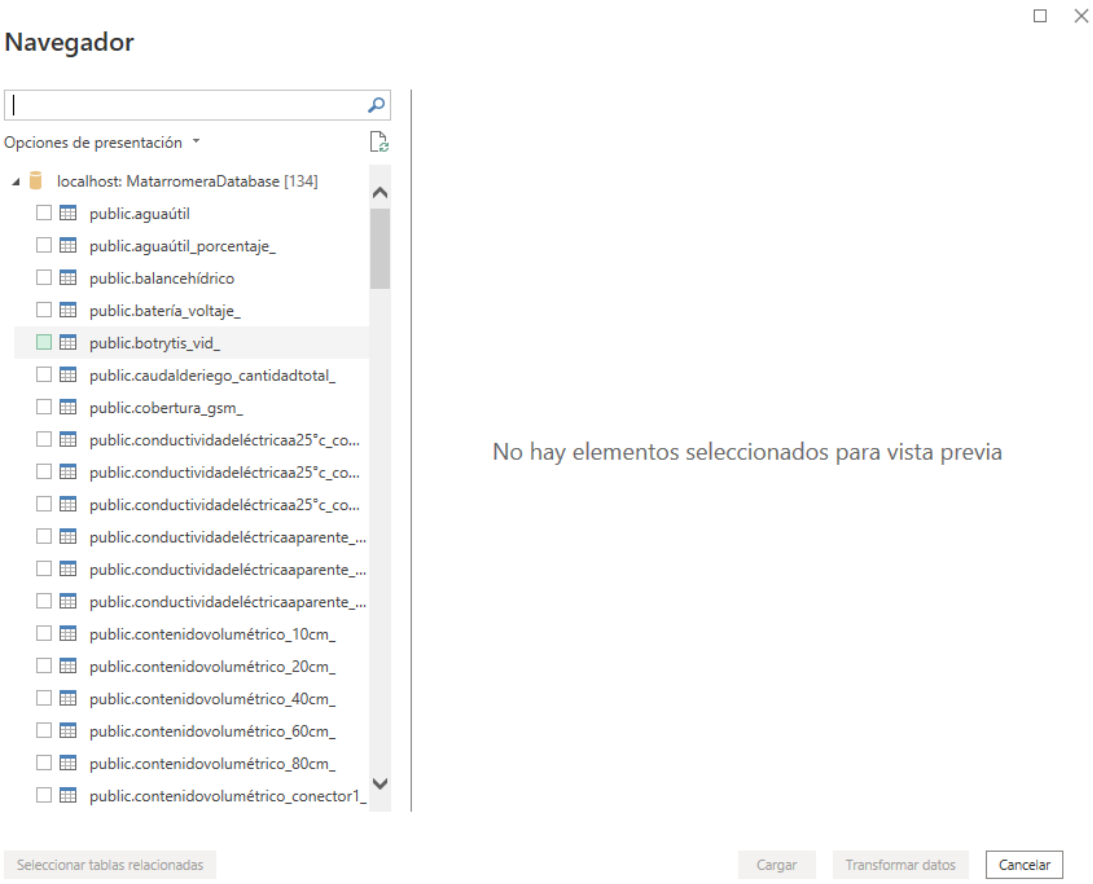


*Figura 18. Ventana que muestra las tablas de la base de datos seleccionada.*

Una vez realizados todos estos pasos, se procederá a crear diferentes ventanas en Power Bi para presentar los datos de las diferentes APIs, tal y como se mostrará a continuación.

Para poder actualizar los datos, se debe seleccionar el botón Actualizar, mostrado en la Figura 19, que conectará con la base de datos y actualizará el contenido de todas las tablas.

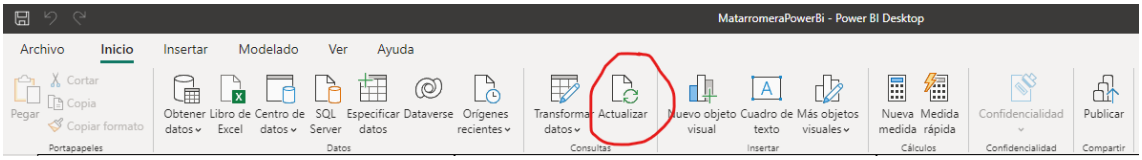


*Figura 19. Botón de Actualizar.*

### 4.1.1. Interfaz para representar datos de la API Cesens

Para la realización de este apartado, previamente a introducir gráficas, se juntó en una sola tabla todas las tablas de la base de datos original y relacionadas con los datos de la API de Cesens, de forma que todos los datos fuesen accesibles desde una sola tabla en lugar de las más de 100 que proporciona esta API. Para poder crear esta tabla, se debe seleccionar el apartado Vista de datos, en la parte izquierda de la ventana de Power Bi, tal como se muestra en la Figura 20.


*Figura 20. Lugar en el que se encuentra Visor de datos.*

Tras esto, se debe seleccionar Nueva tabla, en la parte superior de la ventana, como se ve en la Figura 21. Se abrirá a continuación una ventana en la que se podrá crear la nueva tabla. Para ello, en el cuadro de texto indicado en la Figura 18, se debe indicar el nombre de la nueva tabla junto con la sentencia UNION(), que indica que se quiere unir las diferentes tablas que se introduzcan entre los paréntesis. Todas las tablas que se quieran unir deben tener el mismo formato, es decir, el mismo número de columnas y que estas columnas tengan el mismo formato de datos. En la Figura 22 se muestra el cuadro de texto con las tablas de Cesens listas para ser unidas.


*Figura 21. Lugar en el que se encuentra Nueva tabla.*

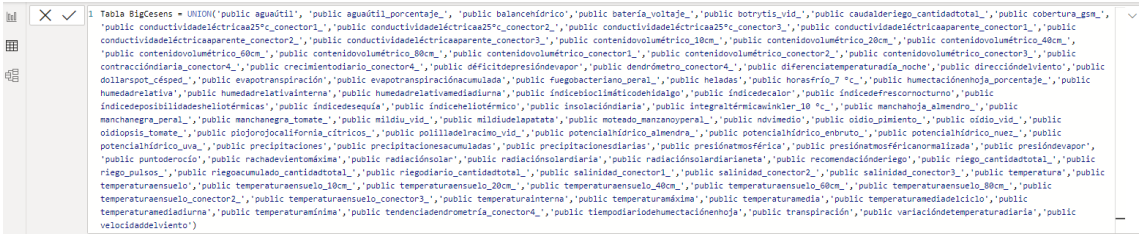


*Figura 22. Cuadro de texto en el que se indican las tablas que se quieren unir.*

A continuación, se debe crear el esquema de tablas del modelo. Para ello se debe seleccionar justo debajo de Visor de datos, en Vista del modelo tal como se aprecia en la Figura 23.


*Figura 23. Vista del modelo.*

Una vez seleccionado el Visor del modelo, se mostrará una ventana vacía, en la cual se deben arrastrar las tablas del modelo, las cargadas desde la base de datos o las que se creen, desde donde se indica en la Figura 24, a la derecha de la pantalla.


*Figura 24. Panel con todas las tablas cargadas desde la base de datos.*

Ahora, se selecciona Administrar relaciones, en la parte superior de la ventana (Figura 25). Una vez hecho este proceso, se abre una ventana nueva que muestra las relaciones entre tablas ya creadas, y se da la opción de crear nuevas relaciones como se ve en la Figura 26.

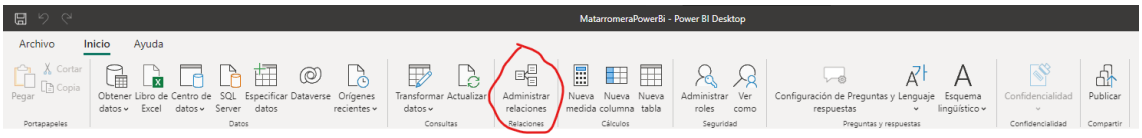


*Figura 25. Lugar en el que se encuentra Administrar relaciones.*

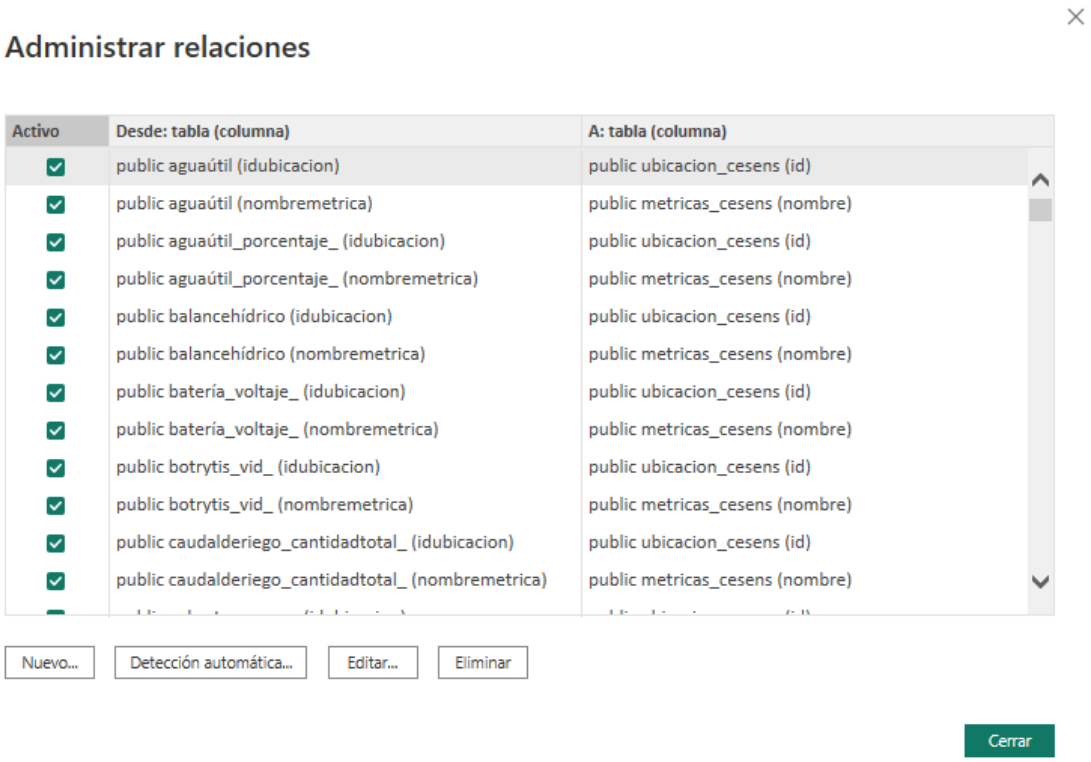


*Figura 26. Ventana de Administrar relaciones.*

Después de crear las relaciones necesarias, se creará el esquema de tablas con el que se trabajará. Todo el proceso anteriormente indicado es común a todas las ventanas de Power Bi. El resultado para los datos de la API de Cesens es el que se ve en la Figura 27. La tabla BigCesens es la tabla que contiene todas las tablas de Cesens con todos sus datos. La relación con ubicaciones_cesens, que contiene los datos de las ubicaciones de los sensores de Cesens, es de una a varios, es decir, una ubicación de esta tabla dará acceso

a los datos de BigCesens que contengan esta ubicación. Se complementa también con metricas_cesens, que contiene las id de las ubicaciones de Cesens, y al igual que ubicaciones_cesens, su relación es de uno a varios, de modo que una id de una métrica dará acceso a los datos de BigCesens con esa id de metrica. También existe una tabla llamada metubis, usada principalmente para los filtros, ya que contiene, además de la id de la métrica y su nombre simplificado, el nombre completo de esta.

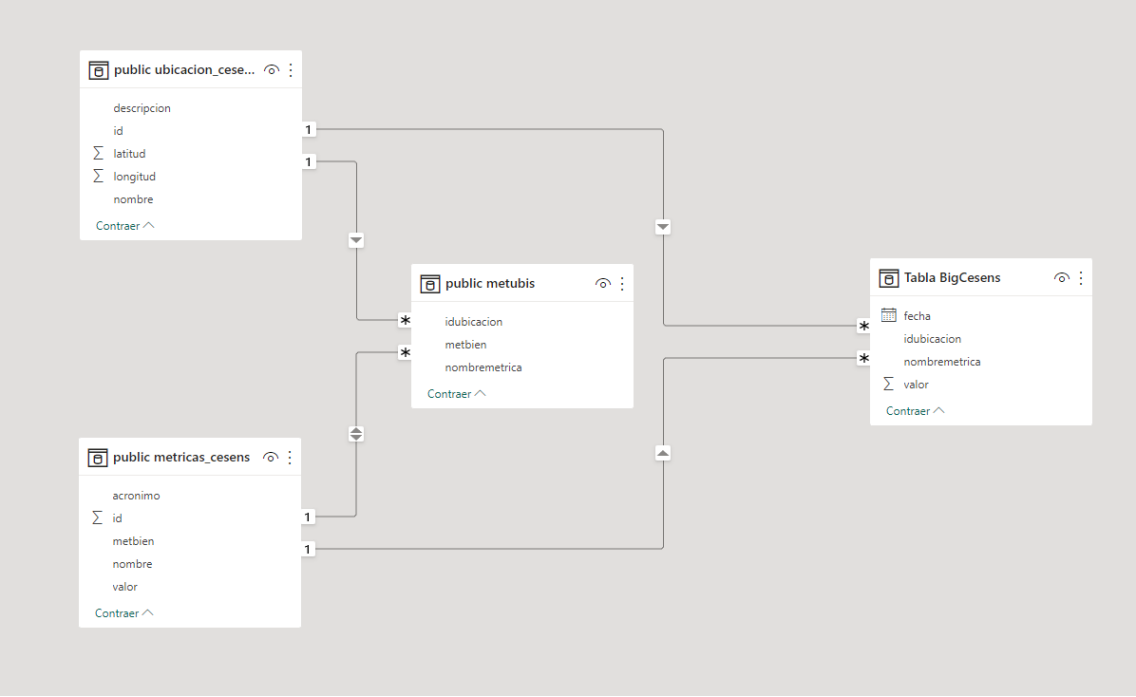


*Figura 27. Esquema de tablas de Cesens.*

Una vez realizados todos estos pasos, comunes a las demás ventanas tal y como ya se ha dicho antes, se procede a crear el informe. Se selecciona Vista, Figura 28, de informe y una vez seleccionado se estará en disposición de elegir las gráficas y filtros que se deseen para el visionado de datos. Estas gráficas se seleccionan desde el panel Visualizaciones, al lado de las tablas cargadas desde la base de datos tal como se muestra en la Figura 29.

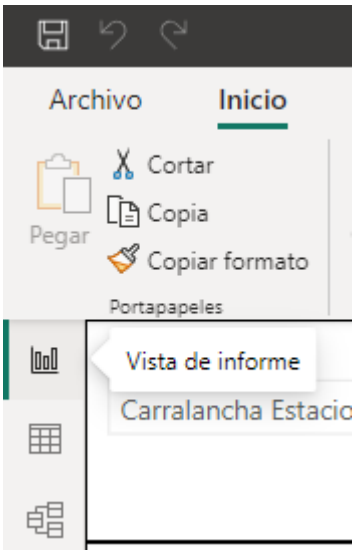


*Figura 28. Vista de informe.*

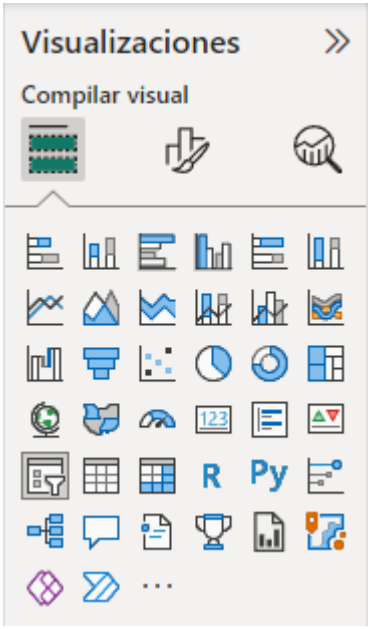


*Figura 29. Panel con todos los filtros y gráficos disponibles en Power Bi.*

Para esta ventana relacionada con los datos de la API de Cesens, mostrada en la Figura 30, se han seleccionado tres filtros en la parte superior de la ventana: el primero a la izquierda para seleccionar la ubicación, el segundo para seleccionar la métrica, y el último para elegir el periodo desde el cual y hasta el cual se quieren ver los datos. Debajo de estos tres filtros se ha colocado un Gráfico de líneas (izquierda), que indica el máximo registrado de la métrica seleccionada, así como su mínimo y el valor promedio del parámetro seleccionado (solo muestra puntos en lugar de líneas porque se ha configurado para mostrar un único dato, de forma que si se selecciona un día muestre los datos

relativos a ese día). Al lado derecho se ha configurado un Gráfico de columnas agrupadas y líneas, donde el eje X representa el tiempo, y el eje Y de columnas son los valores máximos y mínimos (cada columna representa un día). La línea representa el promedio de los valores.

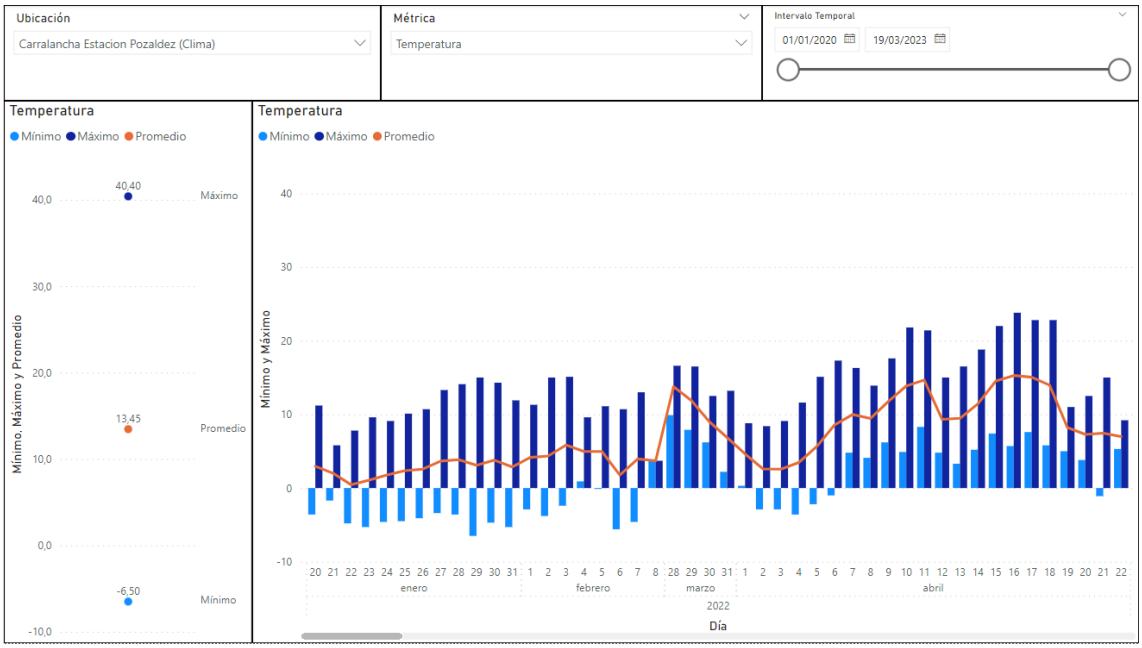


*Figura 30. Ventana de Cesens en Microsoft Power Bi.*

### 4.1.2. Interfaz para representar datos de la API Modpow

Todos los pasos para la realización de esta ventana o interfaz vienen indicados en el apartado anterior Cesens.

El esquema de tablas resultante de Modpow, después de la unión de las tablas que fuesen necesarias, es el mostrado en la Figura 31. La tabla BigModpow contiene los datos de todas las tablas de Modpow. Para llegar a estos datos, que están mezclados, se utilizan las tablas ubicacion_modpow y metricas_modpow. La primera se relaciona de una a varios, de forma que una ubicación dará acceso a varios datos de BigModpow. La relación con metricas_modpow es de varios a varios debido a que, por fallos en la API de Modpow y no solucionados por esta empresa, hay métricas con el mismo nombre pero en donde existe caracteres extra como espacios innecesarios o letras minúsculas donde debería haber mayúsculas.

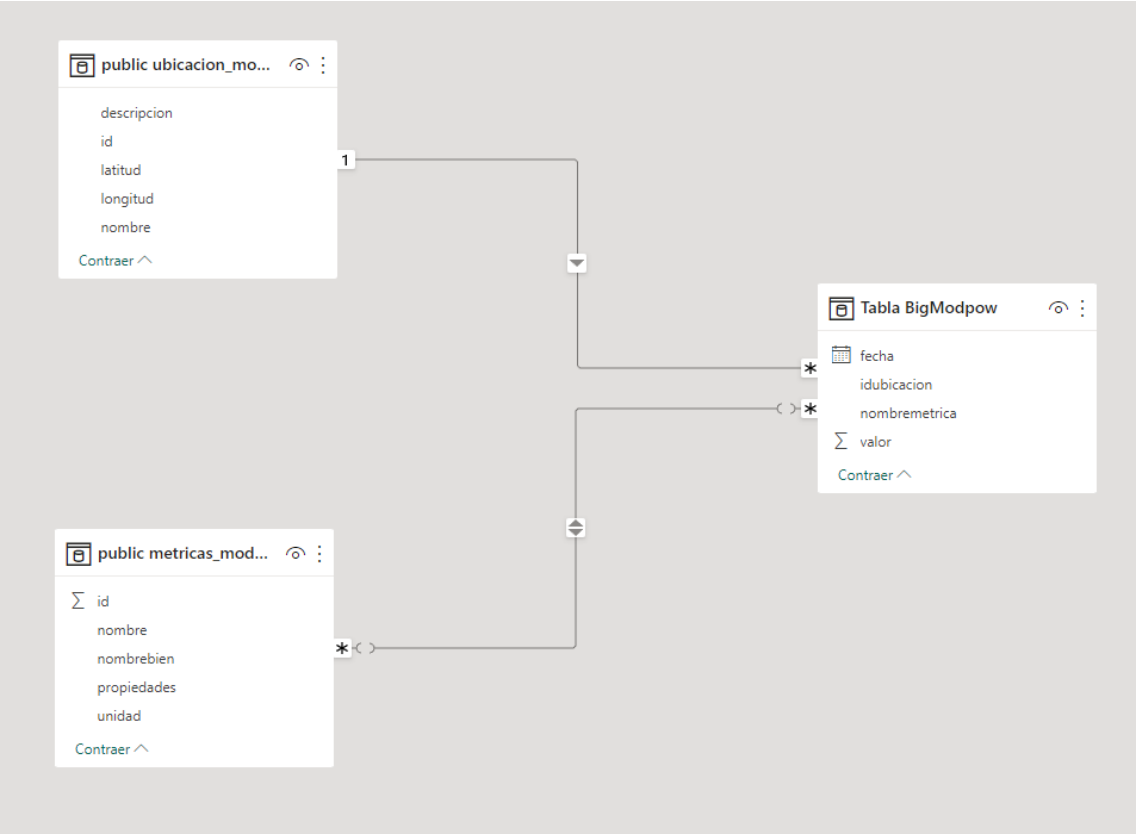


*Figura 31. Esquema de tablas de Modpow.*

El informe final, Figura 32, tiene el mismo esquema que el visto en el apartado anterior. Los tres filtros indican, de izquierda a derecha, la ubicación, la métrica y el intervalo de fecha que se quiera seleccionar. Debajo de estos se ha colocado un Gráfico de líneas (izquierda), que indica el máximo registrado de la métrica seleccionada, así como su mínimo y el valor promedio de esta (solo muestra puntos en lugar de líneas porque se ha configurado para mostrar un único dato, de forma que si se selecciona un día muestre los datos relativos a ese día). Al lado se ha elegido un Gráfico de columnas agrupadas y líneas, donde el eje X representa el tiempo, y el eje Y de columnas son los valores máximos y mínimos para cada día representado. La línea representa el promedio de los valores.

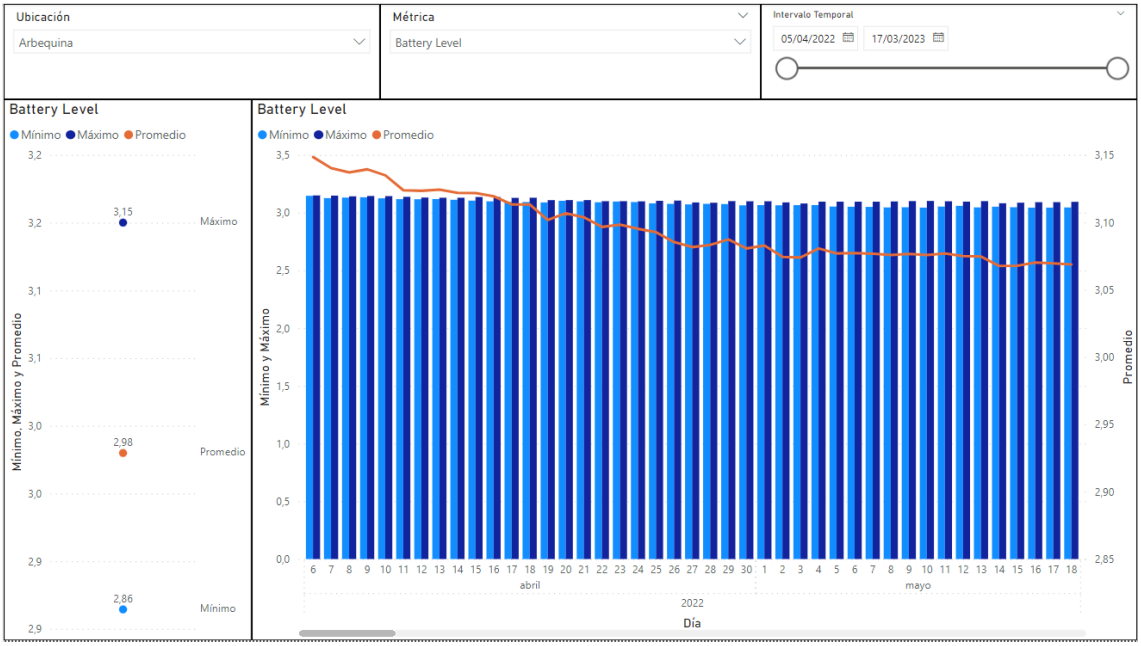


*Figura 32. Ventana de Modpow en Microsoft Power Bi.*

### 4.1.3. Interfaz para representar datos de Temperaturas

Todos los pasos para la realización de esta ventana vienen indicados en el apartado anterior Cesens.

El esquema de tablas resultante de Temperaturas, después de la unión de las tablas que fuesen necesarias, es el mostrado en la Figura 33. Cabe destacar que los datos de Temperaturas provienen también de Cesens, pero de estaciones públicas, como se ha dicho anteriormente. Así, BigTemperatura contiene los datos de las estaciones públicas de donde se han obtenido los datos. Para poder obtenerlos de forma separada, se hace uso de la relación de uno a varios desde la tabla de ubicaciones_temperaturas. También existe la relación de uno a varios con la tabla temperatura, que contiene unicamente los datos de temperatura de Carralancha, y cuya relación es usada para las fechas, de forma que solo se puedan seleccionar las que están en la tabla de la métrica de Carralancha.

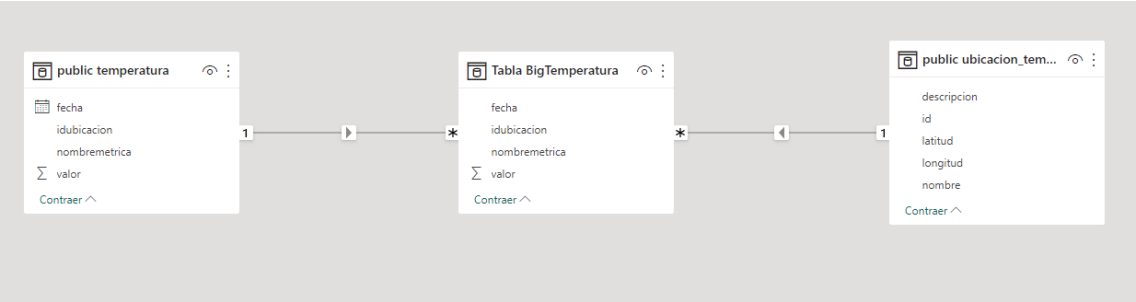


*Figura 33. Esquema de tablas de Temperaturas.*

El informe final, Figura 34, tiene el mismo esquema que el visto en el apartado anterior, pero con dos filtros. El filtro de la izquierda indica el intervalo de fechas a seleccionar y el de la derecha la ubicación que se quiera visualizar. Debajo de estos se ha colocado un Gráfico de líneas, que indica el valor promedio de la temperatura de Carralancha y el valor promedio de la estación pública seleccionada. Al lado se ha elegido un Gráfico de columnas agrupadas, donde el eje X representa el tiempo, y el eje Y de representa el valor promedio de la temperatura de Carralancha y de la estación pública seleccionada.

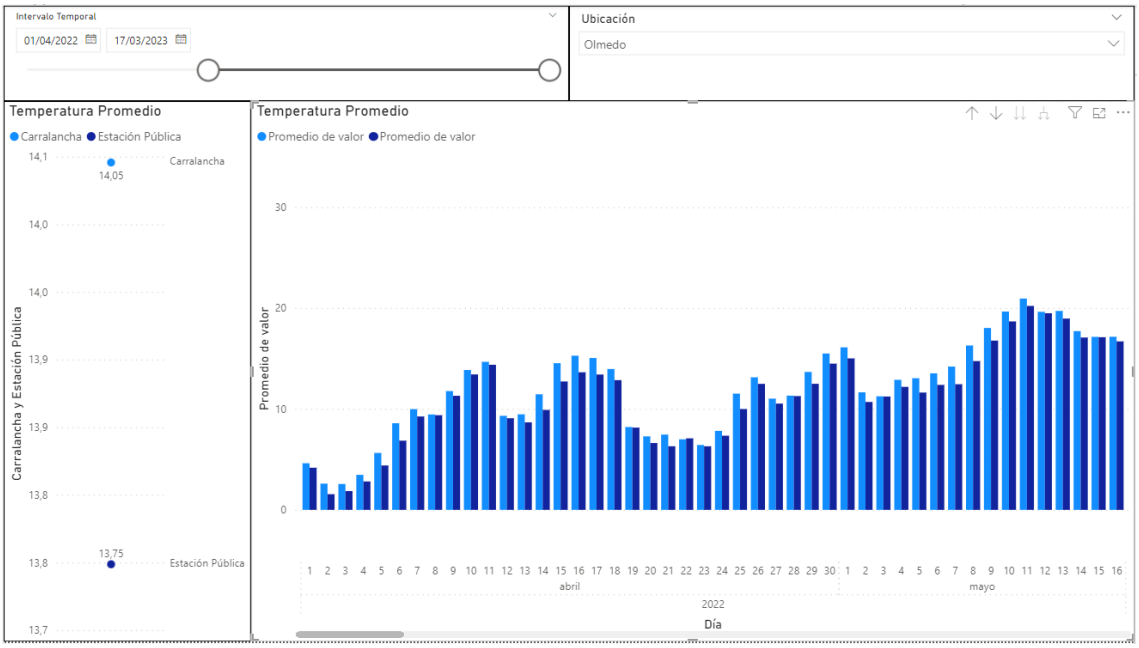


*Figura 34. Ventana de Temperaturas en Microsoft Power Bi.*

# Capítulo 5. Datos e imágenes de satélites

En este apartado del proyecto se ha investigado cómo el uso de los satélites de Sentinel podría ayudar a la hora de automatizar los riegos, y, además, añadir información para la futura implementación de la Inteligencia Artificial. En primer lugar, se comentará cómo funciona Sentinel, qué satélites usa y cómo funcionan. Seguidamente, se detallarán qué satélites se han usado para este proyecto y por qué.

Posteriormente, se mostrarán las webs que han ayudado en la realización de este apartado, así como el código Python desarrollado, y los resultados obtenidos.

## 5.1. Introducción a Sentinel

Sentinel es un proyecto multi-satélite desarrollado por la ESA (*European Space Agency*) en el marco del Programa Copérnico. Este programa consta de siete misiones diferentes, cada una con un propósito propio. Las misiones son las siguientes:

- Sentinel-1: comprende una constelación de dos satélites en órbita polar, que operan día y noche, promocionando imágenes de radar tanto terrestres como marinas.
- Sentinel-2: comprende una constelación de dos satélites de órbita polar ubicados en la misma órbita síncrona del Sol, con una fase de 180º entre ellos, proporcionando imágenes ópticas terrestres de alta resolución para servicios terrestres.
- Sentinel-3: su principal cometido es medir la topografía de la superficie marina, su temperatura, la temperatura de la superficie terrestre, y el color de la superficie del océano y la tierra con alta precisión para respaldar los sistemas de pronóstico oceánico y monitoreo ambiental y climático.

- Sentinel-4: su principal objetivo es monitorear los gases y aerosoles clave de la calidad del aire en Europa en apoyo del CAMS (Copernicus Atmosphere Monitoring Service) a alta resolución espacial y con un tiempo de revisión rápido.
- Sentinel-5: consiste en un solo instrumento, un espectrómetro UV-VIS-NIR-SWIR, proporcionando datos para la vigilancia de la composición atmosférica.
- Sentinel-5P, el precursor de Sentinel-5 (de ahí viene la P), también proporciona datos de la atmósfera terrestre con una alta resolución espacio-temporal, para usarse en calidad del aire, radiación UV y ozono, y monitorización climática.
- Sentinel-6: es una misión colaborativa del Programa Copérnico, implementada y cofundada por la EC, la ESA, la EUMETSAT (European Organisation for the Exploitation of Meteorological Satellites) y los EEUU, a través de la NASA y la NOAA (National Oceanic and Atmospheric Administration).

## 5.2. Aplicación de Sentinel al caso de estudio

Para el desarrollo de este proyecto no se han usado los siete satélites. Solamente se han usado dos para obtener los datos NDVI (*Normalised Difference Vegetation Index*), siglas en inglés de Índice de Vegetación de Diferencia Normalizada, en concreto, el Sentinel-1 y el Sentinel-2. Los datos del NDVI se obtienen mediante un código que combina ambos satélites. Para los datos atmosféricos se ha utilizado el Sentinel-5P.

Sentinel-1 cuenta con imágenes de banda C que operan en cuatro modos de imágenes exclusivos con diferente resolución (de hasta 5 m) y cobertura (de hasta 400 Km). Por su parte, Sentinel-2 posee un sensor MSI (*MultiSpectral Instrument*) de trece bandas, de las cuales cuatro son visibles, seis son de infrarrojo cercano, y tres son de onda corta. Su resolución llega hasta los 10 m de píxel, subiendo hasta los 60 m de píxel, dependiendo de la longitud de onda utilizada. Finalmente, Sentinel-5P tiene una resolución espacial de hasta 5.5 Km x 3.5 Km de píxel, lo que quiere decir que tiene una resolución de 5.5 Km en la dirección de vuelo del satélite, y de 3.5 Km en la dirección perpendicular en nadir. Su sensor es un TROPOMI (*Tropospheric Monitoring Instrument*), un espectrómetro que puede medir longitudes de onda del rango ultravioleta, visible, cercano al infrarrojo y de onda corta.

El objetivo a la hora de usar esta herramienta ha sido obtener datos sobre los terrenos de Carralancha y Valdecovo, específicamente datos sobre la composición atmosférica de gases y NDVI.

Dentro de cada misión de Sentinel, a la hora de acceder a los datos proporcionados por su API, existen diferentes formas de obtener estos datos. La opción elegida para obtener los datos de NDVI de Sentinel-1 ha sido la S1GRD (*Ground Range Detected*), con bandas VV y VH, y para Sentinel-2 ha sido S2L2A, con las bandas B04, B08 y SCL. Con Sentinel-5P, en función de lo que se quiera medir, se usa una configuración u otra. Las diferentes configuraciones permiten medir monóxido de carbono (CO), dióxido de nitrógeno (NO2), formaldehído (HCHO), ozono (O3), dióxido de sulfuro (SO2), metano (CH4), AER AI 340 y 380, AER AI 354 y 388, altura o presión de la base de las nubes, fracción de nube radiométrica efectiva, espesor óptico de las nubes, y altura y presión máxima de las nubes [8].

## 5.3. Problemática con datos de Sentinel

Para elaborar esta parte del proyecto, se ha tenido que hacer frente a un problema grave, la resolución de los satélites. Para entrar en contexto, las parcelas agrícolas usadas tienen una superficie total de 45.71 ha, o lo que es lo mismo, 0.4571 $Km^2$ en Carralancha, y una superficie total de 72.16 ha, o lo que es lo mismo, 0.7216 $Km^2$ en Valdecovo (todos los datos proporcionados por BMSL, empresa que trabaja con Grupo Matarromera).

Para los satélites de Sentinel-1 y Sentinel-2, el hecho de que la parcela sea de estas dimensiones no supone ningún problema, ya que la resolución de estos llega hasta los 10 $m^2$ de píxel. Anteriormente en el documento también se dio esta expresión de m de píxel, que vendría a ser que la resolución en estos satélites es de hasta 10 $m^2$. De esta forma no representa ningún problema el tamaño de la parcela con la resolución de la imagen obtenida.

El problema reside en Sentinel-5P. Como se dijo anteriormente, la resolución es de 5.5 Km x 3.5 Km, o lo que es lo mismo, un píxel de 5.5 Km de largo y 3.5 Km de ancho. Esta resolución es significativamente mayor al tamaño de la parcela, causando que la imagen obtenida para los datos atmosféricos sea monocromática, debido a que las

parcelas de Carralancha y Valdecovo son significativamente menores que el píxel obtenido.

Como se puede apreciar en la Figura 35, no se obtiene una serie de píxeles de diferente gama cromática como en la Figura 36, sino que se obtiene un único píxel debido al tamaño de la parcela y a la resolución del satélite tal y como se explicó con anterioridad. Debido a este inconveniente, se decidió no seguir adelante con el uso e integración en el proyecto de Sentinel-5P. De este modo, de ahora en adelante, solo se hablará en este documento de Sentinel-1 y Sentinel-2.

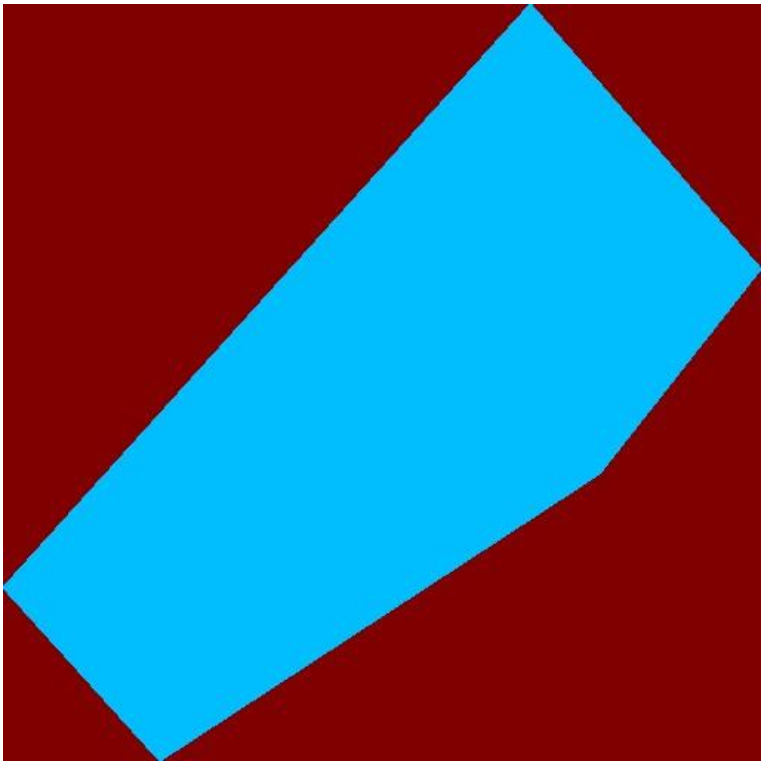

*Figura 35. Ejemplo de imagen obtenida con Sentinel-5P en Carralancha.*

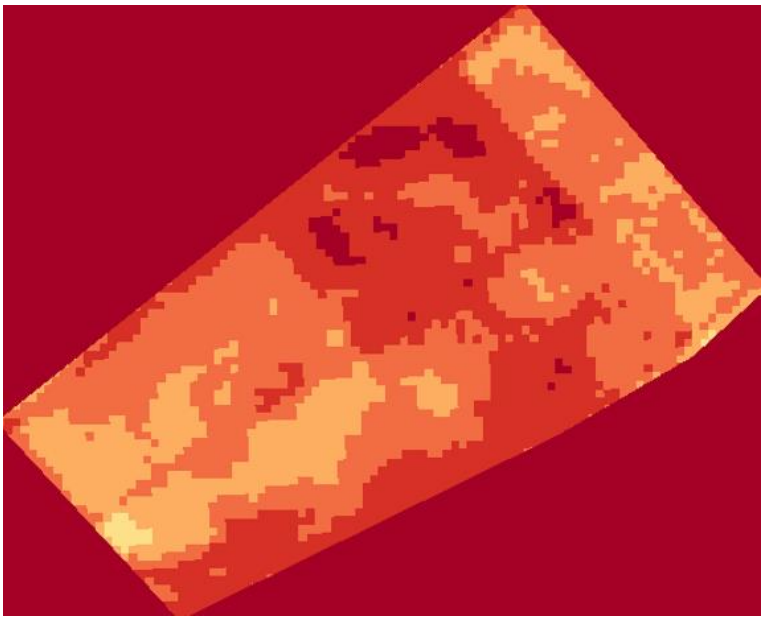

*Figura 36. Ejemplo de imagen obtenida con Sentinel-1 y Sentinel-2 en Carralancha.*

Otro problema, referido en esta ocasión a Sentinel-1 y Sentinel-2, ha sido la presencia de nubes en las imágenes obtenidas. Cuando hay una nube, esta aparece con un píxel de color verde oscuro (tal y como se observa en la Figura 37), muy diferente al color verde que se podría obtener en la escala del NDVI. El problema reside en que, si aparece una nube, esta dificulta la tarea de obtener los datos del NDVI, ya que bloquea la vista sobre parte de la parcela, de modo que los datos estarían incompletos. Como se puede apreciar en la Figura 37, las nubes tapan parte de la zona baja de la parcela, impidiendo obtener los datos al completo. Al contrario que con el problema que reside con Sentinel-5P, sí se podría evitar este problema, haciendo necesaria una herramienta que analice las imágenes que han sido obtenidas, y descartando aquellas donde se encuentre un píxel o más con este tono verde. Importante a destacar que solo sería ese tono de verde, ya que, como se aprecia en la imagen inferior, hay verdes que sí son válidos ya que forman parte de la escala cromática del NDVI, tal y como se puede observar en la Figura 38.

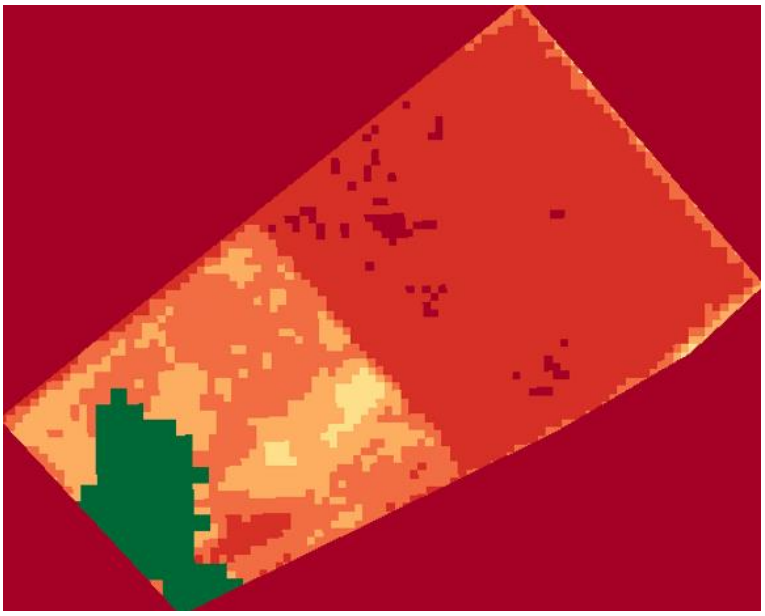

*Figura 37. Ejemplo de imagen con nubes en la parte inferior.*

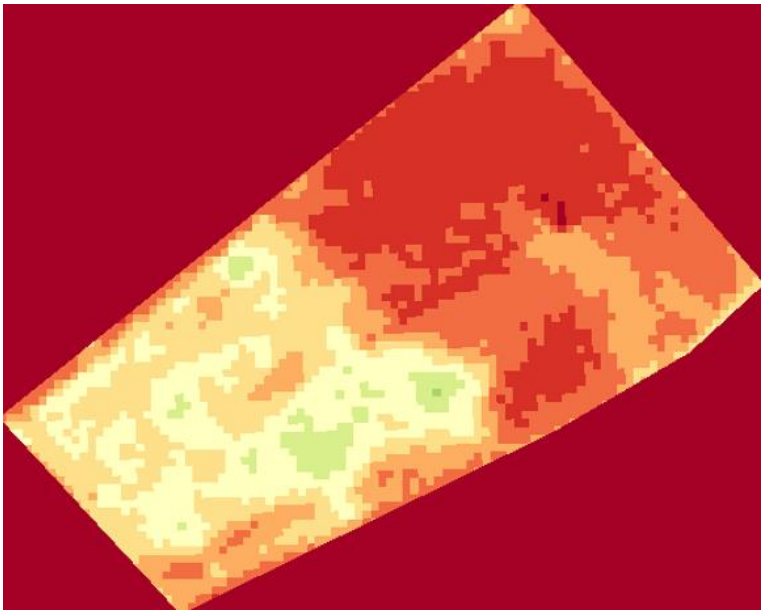

*Figura 38. Ejemplo de imagen con tonos de verde válidos.*

Como se observa en la Figura 39, las nubes corresponderían al nivel más alto de la escala cromática. En esta misma escala reside otro problema referente al problema de las nubes. Como ya se ha explicado, las nubes aparecen con un color verde oscuro, haciendo que las imágenes con este color deban ser eliminadas. Pero este color también representa un índice de vegetación elevado, que indicaría que las plantas están muy sanas, ya que a mayor NDVI, mejor salud tendrá la vegetación. Una posible solución para solventar este

problema sería analizar únicamente las imágenes de Sentinel-2 a partir de un código que nos devolvería imágenes satélite sin procesar, es decir, tal y como las veríamos en Google Maps, pero que destaca las nubes de color rojo, tal y como se aprecia en la imagen de la Figura 40. A partir de estas imágenes, se podría descartar las fechas en las que aparecen nubes, y solo obtener las imágenes de los satélites combinados de Sentinel-1 y Sentinel-2 en las que no hay nubes.

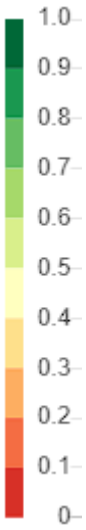


*Figura 39. Escala cromática NDVI usada por Sentinel.*

*Figura 40. Ejemplo de imagen satélite de Sentinel-2 con las nubes destacadas en rojo.*

## 5.4. Resultados

A partir del código, mostrado en el anexo, del fichero Sentinel.py, se han obtenido imágenes de los satélites Sentinel-1 y Sentinel-2 que recogen los datos sobre el NDVI en la parcela de Carralancha.

Las imágenes obtenidas varían entre archivos nítidos donde se distingue con claridad tanto la parcela de cultivo como el rango cromático para poder obtener los datos con detalle, a imágenes contaminadas con píxeles verdes que hacen inviable la tarea de obtener dato alguno, ya que tapan de forma parcial o incluso de forma completa parte del terreno a estudiar. Algunas de las imágenes obtenidas se han mostrado anteriormente en el documento.

A partir de un algoritmo que analizase las imágenes, se podría obtener el NDVI de la parcela en los diferentes puntos de esta, pudiendo estudiar así la evolución de la vegetación a lo largo del tiempo. Junto con los datos recogidos a partir de las APIs, se

estudiará su evolución con más detalle, viendo cómo afectan las lluvias al NDVI, así como las horas de frío y de calor, e incluso diferentes plagas que pudiera sufrir .

# Capítulo 6. Conclusiones y líneas futuras

## 6.1. Conclusiones

En este Trabajo de Final de Grado, en primer lugar, se han expuesto las motivaciones a la realización de este, se ha realizado una breve introducción y se han descrito los recursos utilizados para su elaboración con un detalle mayor. La motivación de este proyecto viene de la necesidad de Grupo Matarromera de elaborar una base de datos para poder crear un sistema automatizado de riego y fertilización.

Seguidamente, se ha explicado el código utilizado durante el proyecto, gracias al cual se han podido obtener todos los datos necesarios para poder construir la base de datos sobre la que se cimienta el trabajo.

A continuación, se ha descrito, como en una guía de usuario, la construcción pormenorizada del software de visualización de datos, gracias al cual podemos ver en detalle los datos extraídos de las APIs.

Finalmente, se ha mostrado como se obtienen los datos de Sentinel, los problemas que presenta, así como sus posibles soluciones, con el fin de poder ver el NDVI sobre las parcelas de trabajo.

## 6.2. Líneas Futuras

Tras la finalización de este trabajo, se indicarán a continuación algunas de las líneas futuras a seguir para, no solo mejorar el proyecto presente, si no para llevar más allá el desarrollo del *business analytics.*

Una de las ideas que se pueden desarrollar es la inclusión de más APIs a la base de datos, mejorando así la cantidad de datos a los que se pueden acceder y añadiendo más información de utilidad. No solo la inclusión de más APIs sería beneficiosa, añadir más

ubicaciones ayudaría a monitorizar más parcelas de cultivo, que podría suponer un ahorro de tiempo y mejora de eficiencia.

La línea futura que quizás sea más atractiva, y que más ayude a Grupo Matarromera, sería la implementación de una IA. Esta IA podría aprender gracias a todos los datos recogidos desde las APIs, y podría desarrollar modelos predictivos que indicasen a los encargados del cuidado de los cultivos cuándo regar o cuándo hacer uso de fertilizantes.

Además, se puede contar con las imágenes satélite de Sentinel, que pueden aportar datos valiosos actualizados sobre el estado del cultivo, lo cual junto con los datos de la tierra y los datos meteorológicos pueden proporcionar una mejor forma de automatización.

La inclusión de una inteligencia artificial capaz de automatizar sistemas supondría un gran avance para la bodega, que la podría situar por delante de muchos competidores a nivel tecnológico, proporcionándoles un ahorro económico considerable posiblemente, y más importante, mejorando la calidad de los productos más si cabe.

# Capítulo 7. Bibliografía


[1] «Página web de Microsoft Power Bi» [En línea]. Available: https://powerbi.microsoft.com/es-es/what-is-power-bi/

[2] «Página web de Sentinel» [En línea]. Available: https://sentinels.copernicus.eu/web/sentinel/home

[3] «Página web de Python» [En línea]. Available: https://www.python.org/

[4] «Página web de Visual Studio Code» [En línea]. Available: https://code.visualstudio.com/

[5] «Página web de Cesens» [En línea]. Available: https://www.cesens.es/

[6] «Página web de Modpow» [En línea]. Available: https://www.modpowagritech.com/

[7] «Página web de PostgreSQL» [En línea]. Available: https://www.postgresql.org/about/

[8] «Página web de Sentinel Hub, Sentinel 5P» [En línea]. Available: https://docs.sentinel-hub.com/api/latest/data/sentinel-5p-l2/

# Capítulo 8. Anexos

## 8.1. Código main.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamadas a todos los archivos para la ejecución del código
from imports import *
from loggin_cesens import *
from loggin_modpow import *
from ubicaciones_cesens import *
from ubicaciones_modpow import *
from metricas_cesens import *
from metricas_modpow import *
from datos_cesens import *
from datos_modpow import *
from database import *
from metubiscesens import *
from temperaturas import *
#Si la carpeta ya existe, no se crea de nuevo
carpeta_cesens='Datos Cesens'
if os.path.isdir(carpeta_cesens)==False:
    os.mkdir(carpeta_cesens)
#Si la carpeta ya existe, no se crea de nuevo
carpeta_modpow='Datos Modpow'
if os.path.isdir(carpeta_modpow)==False:
    os.mkdir(carpeta_modpow)
#Se realiza la conexión con la base de datos
conexion=database()
#Se obtiene la cabecera para obtener las URLs de Cesens
cabeza_cesens=loggearse_cesens()
#Se obtiene la cabecera para obtener las URLs de Modpow
cabeza_modpow=loggearse_modpow()
#Se obtienen las ubicaciones de Cesens
ubicacion_cesens(cabeza_cesens,conexion)
#Se obtienen las ubicaciones de Modpow
ubicacion_modpow(cabeza_modpow,conexion)
#Se obtienen las metricas de Cesens
metrica_cesens(cabeza_cesens,conexion)
#Se obtienen las metricas de Modpow
metrica_modpow(cabeza_modpow,conexion)
#Se obtienen las métricas asociadas a cada ubicación de Cesens
metubis(cabeza_cesens,conexion)
#Se obtienen los datos de las estaciones climáticas más cercanas
temperaturas(cabeza_cesens,conexion)
#Se obtienen los datos de Cesens
dato_cesens(cabeza_cesens,carpeta_cesens,conexion)
#Se obtienen los datos de Modpow
dato_modpow(cabeza_modpow,carpeta_modpow,conexion)
```

## 8.2. Código imports.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Lamadas a las librerias para la ejecución del código
import json as js
from urllib import response as rp
from wsgiref.util import request_uri as rq
import requests as rqs
import win32com.client as win32
import os as os
import pandas as pd
import shutil as sh
import csv as filecsv
import datetime as dt
import time as tm
import psycopg2 as psql
#Abreviatura de las librerias para llamar a sus funciones
__all__ = [
    'js',
    'rp',
    'rq',
    'rqs',
    'win32',
    'os',
    'pd',
    'sh',
    'filecsv',
    'dt',
    'tm',
    'psql'
]
```

## 8.3. Código loggin_cesens.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener la cabecera de Cesens
def loggearse_cesens():
    #Cuerpo que se envía junto coon el requests.post para iniciar
sesión con las credenciales
    log = {
        "nombre": "correo@email.com",
        "clave": "contraseña"
    }
```

```
    #Inicio de sesión
    login=rqs.post("https://app.cesens.com/api/usuarios/login",
data=js.dumps(log))
    #Para poder trabajar con .json
    def jprint(obj):
        text = js.dumps(obj, sort_keys=True, indent=4)
        print(text)
    #Si el inicio de sesión ha ido bien, se crea la cabecera de
autorización
    if login.status_code==200:
        head='Token' + ' ' + login.json()['auth']
        #Crea la cabecera
        cabeza_cesens = {
            "Authentication": head
        }
    #Si no inicia sesión bien se notifica
    else:
        jprint('Algo estás haciendo mal')
    #Se retorna la cabecera
    return cabeza_cesens
```

## 8.4. Código loggin_modpow.py

```
#######################################
##                                   ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ    ##
##                                   ##
#######################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener la cabecera de Modpow
def loggearse_modpow():
    #Cuerpo que se envía junto coon el requests.post para iniciar
sesión con las credenciales
    log = {
        "login":"correo@email.com",
        "password":"contraseña",
        "device":{"vendor":"API","platform":"API","type":"API"}
    }
    #Inicio de sesión
    login=rqs.post("https://apiv2.wireless-
monitoring.net/ioecrops/extra/login", data=js.dumps(log))
    #Para poder trabajar con .json
    def jprint(obj):
        text = js.dumps(obj, sort_keys=True, indent=4)
        print(text)
    #Si el inicio de sesión ha ido bien, se crea la cabecera de
autorización
    if login.status_code==200:
        head=login.json()['token']
        #Crea la cabecera
        cabeza_modpow = {
            "Authorization": head
        }
```

```
    #Si no inicia sesión bien se notifica
    else:
        jprint('Algo estás haciendo mal')
    #Retorna la cabecera
    return cabeza_modpow
```

## 8.5. Código database.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerías necesarias
from imports import *
#Función con la cual se hace conexión con la base de datos de
PostgreSQL
def database():
    #Intento de conexión
    try:
        connection=psql.connect(
            #Credenciales de la base de datos
            host=host,
            user=user,
            password=contraseña,
            database=nombredatabase
        )
        #Notificación de conexión correcta
        print("Conexión exitosa")
        connection.autocommit=True
        return(connection)
    #Notificación en caso de fallo de conexión
    except Exception as ex:
        print(ex)
```

## 8.6. Código ubicaciones_cesens.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener las ubicaciones de Cesens
def ubicacion_cesens(cabeza_cesens,conexion):
    #Si el fichero existe, no hay necesidad de ejecutar el código
    if os.path.isfile('Ubicaciones_Cesens.csv')==False:
        #Cadenas para obtener unicamente las ubicaciones de
Carralancha y Valdecovo
```

```
        nombre1="Carralancha"
        nombre2="Valdecovo"
        #Array principal donde se guardarán los datos
        filas=[]
        #Cabecera del .csv
        header = ('id','nombre','descripcion', 'latitud', 'longitud')
        #Obtención de las ubicaciones
        ubi=rqs.get("https://app.cesens.com/api/ubicaciones/",
headers=cabeza_cesens)
        data1=js.dumps(ubi.json())
        info=js.loads(data1)
        #Iteración en los datos para despues guardarlos en un vector
que se usará para cargar los datos en el .csv
        for registro in info:
            if(nombre1 in registro['nombre'] or nombre2 in
registro['nombre']):
                id=registro['id']
                nombre=registro['nombre']
                descripcion=registro['descripcion']
                latitud=registro['latitud']
                longitud=registro['longitud']
                filas.append([id, nombre, descripcion, latitud,
longitud])
        #Ordenación el vector
        filas.sort()
        #Se guardan en un archivo los datos ordenados
        with open('Ubicaciones_Cesens.json','w') as json_file:
            js.dump(filas, json_file, indent=4)
        #Se cambia el formato de .json a .csv
        read_file = pd.read_json
(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Cesens.json')
        read_file.to_csv
(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Cesens.csv', index =
None, header=header,sep=";",decimal='.',float_format='%.5f')
        #Se borra el archivo .json
        os.remove('Ubicaciones_Cesens.json')
        #Conexión con base de datos
        cursor=conexion.cursor()
        #Sentencia SQL para crear las tablas indicadas si no existían
antes
        tubicacion="CREATE TABLE IF NOT EXISTS ubicacion_cesens(id INT
PRIMARY KEY,nombre VARCHAR(50),descripcion VARCHAR(50),latitud
FLOAT,longitud FLOAT)"
        cursor.execute(tubicacion)
        try:
            #Se borra el contenido de la tabla para no repetir filas
            borrar="DELETE FROM ubicacion_cesens"
            cursor.execute(borrar)
            #Se copia el contenido del .csv en la tabla SQL
            with
open('C:/Users/Jaime/Desktop/Database/Ubicaciones_Cesens.csv','r',enco
ding="utf8") as f:
                next(f)
                cursor.copy_from(f,'ubicacion_cesens',sep=';')
            #Desconexión de base de datos
            cursor.close()
        #Si la tabla ya existía no se hace nada
        except:
            print("Tabla ya existente")
```

## 8.7. Código ubicaciones_modpow.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener las ubicaciones de Modpow
def ubicacion_modpow(cabeza_modpow,conexion):
    #Si el fichero existe, no hay necesidad de ejecutar el código
    if os.path.isfile('Ubicaciones_Modpow.csv')==False:
        #Obtención las ubicaciones
        ubi=rqs.get("https://apiv2.wireless-
monitoring.net/ioecrops/devices",headers=cabeza_modpow)
        data1=js.dumps(ubi.json())
        info=js.loads(data1)
        #Array principal donde se guardarán los datos
        filas=[]
        #Cabecera del .csv
        header=('id','nombre','descripcion','latitud','longitud')
        #Iteración en las diferentes ubicaciones
        for item in info['devices'] :
            newurl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']
            ubi2=rqs.get(newurl,headers=cabeza_modpow)
            data2=js.dumps(ubi2.json())
            info2=js.loads(data2)
            #Se guardan los datos en el array
            for item2 in info2['devices']:
                id=item2['id']
                nombre=item2['deployment']['label']
                descripcion=item2['label']
                latitud=item2['location']['latitude']
                longitud=item2['location']['longitude']

filas.append([int(id),nombre,descripcion,latitud,longitud])
        #Ordenación el vector
        filas.sort()
        #Se guardan en un archivo los datos ordenados
        with open('Ubicaciones_Modpow.json','w') as json_file:
            js.dump(filas, json_file, indent=4)
        #Se cambia el formato de .json a .csv
        read_file = pd.read_json
(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Modpow.json')
        read_file.to_csv
(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Modpow.csv', index =
None, header=header,sep=";",decimal='.',float_format='%.5f')
        #Se borra el archivo .json
        os.remove('Ubicaciones_Modpow.json')
        #Conexión con base de datos
        cursor=conexion.cursor()
        #Sentencia SQL para crear las tablas indicadas si no existían
antes
        tubicacion="CREATE TABLE IF NOT EXISTS ubicacion_modpow(id INT
PRIMARY KEY,nombre VARCHAR(50),descripcion VARCHAR(50),latitud
FLOAT,longitud FLOAT)"
```

```
        cursor.execute(tubicacion)
        try:
            #Se borra el contenido de la tabla para no repetir filas
            borrar="DELETE FROM ubicacion_modpow"
            cursor.execute(borrar)
            #Se copia el contenido del .csv en la tabla SQL
            with
open('C:/Users/jaime/Desktop/Database/Ubicaciones_Modpow.csv','r',enco
ding="utf8") as f:
                next(f)
                cursor.copy_from(f,'ubicacion_modpow',sep=';')
            #Desconexión de base de datos
            cursor.close()
        #Si la tabla ya existía no se hace nada
        except:
            print("Tabla ya existente")
```

## 8.8. Código metricas_cesens.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener las métricas de Cesens
def metrica_cesens(cabeza_cesens,conexion):
    #Si el fichero existe, no hay necesidad de ejecutar el código
    if os.path.isfile('Metricas_Cesens.csv')==False:
        #Cadena para identificar posibles errores o métricas vacias
        error1="msg"
        error2=[]
        error3="[]"
        #Array principal donde se guardarán los datos
        filas=[]
        #Cabecera a usar en el .csv
        header = ('id' ,'nombre','acronimo', 'unidad','metbien')
        #Iteración para obtener todas las métricas
        for x in range (1,353):
            #Obtención de las ubicaciones
            url="https://app.cesens.com/api/metricas/"+repr(x)
            ubi=rqs.get(url, headers=cabeza_cesens)
            data1=js.dumps(ubi.json())
            info=js.loads(data1)
            #Condición para evitar errores y cargar los datos en el
array
            if(error1 not in info and info!=error2 and info!=error3):
                id=info['id']
                nombre=info['nombre'].replace(" ","")
                acronimo=info['acronimo']
                unidad=info['unidad']
                metbien=info['nombre']
                filas.append([id, nombre.lower(), acronimo,
unidad,metbien])
        #Se ordena el array
```

```
        filas.sort()
        #Se guardan en un archivo los datos ordenados
        with open('Metricas_Cesens.json','w') as json_file:
            js.dump(filas, json_file, indent=4)
        #Se cambia el formato de .json a .csv
        read_file = pd.read_json
(r'C:/Users/jaime/Desktop/Database/Metricas_Cesens.json')
        read_file.to_csv
(r'C:/Users/jaime/Desktop/Database/Metricas_Cesens.csv', index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
        #Se borra el archivo .json
        os.remove('Metricas_Cesens.json')
        #Conexión con base de datos
        cursor=conexion.cursor()
        #Sentencia SQL para crear las tablas indicadas si no existían
antes
        tubicacion="CREATE TABLE IF NOT EXISTS metricas_cesens(id INT
,nombre VARCHAR(50) PRIMARY KEY,acronimo VARCHAR(50),valor
VARCHAR(50),metbien VARCHAR(100))"
        cursor.execute(tubicacion)
        try:
            #Se borra el contenido de la tabla para no repetir filas
            borrar="DELETE FROM metricas_cesens"
            cursor.execute(borrar)
            #Se copia el contenido del .csv en la tabla SQL
            with
open('C:/Users/Jaime/Desktop/Database/Metricas_Cesens.csv','r',encodin
g="utf8") as f:
                next(f)
                cursor.copy_from(f,'metricas_cesens',sep=';')
            #Desconexión de base de datos
            cursor.close()
        #Si la tabla ya existía no se hace nada
        except:
            print("Tabla ya existente")
```

## 8.9. Código metricas_modpow.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener las métricas de Modpow
def metrica_modpow(cabeza_modpow,conexion):
    #Si el fichero existe, no hay necesidad de ejecutar el código
    if os.path.isfile('Metricas_Modpow.csv')==False:
        #Obtención de las ubicaciones de Modpow
        met=rqs.get("https://apiv2.wireless-
monitoring.net/ioecrops/devices",headers=cabeza_modpow)
        data1=js.dumps(met.json())
        info=js.loads(data1)
        #Definición de los arrays donde guardaremos los datos
        filas=[]
```

```
        filas2=[]
        #Cabecera a usar en el .csv
        header=('id','nombre','propiedades','unidad','nombrebien')
        #Iteración en las diferentes ubicaciones
        for item in info['devices']:
            newurl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"
            met2=rqs.get(newurl,headers=cabeza_modpow)
            data2=js.dumps(met2.json())
            info2=js.loads(data2)
            #Iteración para obtener las diferentes métricas
            for item2 in info2['datastreams']:
                newurl2=newurl+'/'+repr(item2['id'])
                met3=rqs.get(newurl2,headers=cabeza_modpow)
                data3=js.dumps(met3.json())
                info3=js.loads(data3)
                #Se guardan los datos en el array
                for item3 in info3['datastreams']:
                    id=repr(item3['id'])
                    nombre=item3['label'].replace(" ","")
                    propiedades=item3['property']
                    unidad=item3['units']
                    nombrebien=item3['label']

filas.append([int(id),nombre.lower(),propiedades,unidad,nombrebien])
        #Se ordena el array
        filas.sort()
        #Se eliminan las métricas repetidas
        for x in range (0,len(filas)):
            if filas[x] not in filas2:
                filas2.append(filas[x])
        #Se guardan en un archivo los datos ordenados
        with open('Metricas_Modpow.json','w') as json_file:
            js.dump(filas2, json_file, indent=4)
        #Se cambia el formato de .json a .csv
        read_file = pd.read_json
(r'C:/Users/jaime/Desktop/Database/Metricas_Modpow.json')
        read_file.to_csv
(r'C:/Users/jaime/Desktop/Database/Metricas_Modpow.csv', index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
        #Se borra el archivo .json
        os.remove('Metricas_Modpow.json')
        #Conexión con base de datos
        cursor=conexion.cursor()
        #Sentencia SQL para crear las tablas indicadas si no existían
antes
        tubicacion="CREATE TABLE IF NOT EXISTS metricas_modpow(id INT
,nombre VARCHAR(50),propiedades VARCHAR(50),unidad
VARCHAR(50),nombrebien VARCHAR(100) PRIMARY KEY)"
        cursor.execute(tubicacion)
        try:
            #Se borra el contenido de la tabla para no repetir filas
            borrar="DELETE FROM metricas_modpow"
            cursor.execute(borrar)
            #Se copia el contenido del .csv en la tabla SQL
            with
open('C:/Users/jaime/Desktop/Database/Metricas_Modpow.csv','r',encodin
g="utf8") as f:
                next(f)
                cursor.copy_from(f,'metricas_modpow',sep=';')
            #Desconexión de base de datos
```

```
            cursor.close()
        #Si la tabla ya existía no se hace nada
        except:
            print("Tabla ya existente")
```

## 8.10. Código metubiscesens.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para identificar qué métricas están ligadas a qué ubicaciones
def metubis(cabeza_cesens,conexion):
    #Si el fichero existe, no hay necesidad de ejecutar el código
    if os.path.isfile('metubis.csv')==False:
        ubi=rqs.get("https://app.cesens.com/api/ubicaciones/",
headers=cabeza_cesens)
        data1=js.dumps(ubi.json())
        info=js.loads(data1)
        #Cadenas esto para obtener únicamente las ubicaciones de
Carralancha y Valdecovo
        nombre1="Carralancha"
        nombre2="Valdecovo"
        #Cadena esto para identificar posibles errores o métricas
vacias
        error1="msg"
        error2=[]
        error3="[]"
        #Array principal donde guardaremos los datos
        filas=[]
        #Cabecera a usar en el .csv
        header = ('idubicacion','nombremetrica','metbien')
        #Iteración en las diferentes ubicaciones
        for x in range (1,353):
            #Excepciones de métricas erróneas
            if(x!=90 and x!=163):
                new_url2 = 'https://app.cesens.com/api/metricas/' +
repr(x)
                metricas=rqs.get(new_url2, headers=cabeza_cesens)
                data3=js.dumps(metricas.json())
                info3=js.loads(data3)
                if(error1 not in info3 and info3!=error2 and
info3!=error3):
                    for item in info:
                        #Si no es la ubicación deseada no sigue
                        if(nombre1 in item['nombre'] or nombre2 in
item['nombre']):
                            new_url1 =
'https://app.cesens.com/api/datos/' + repr(item['id']) + '/' + repr(x)
                            datos1=rqs.get(new_url1,
headers=cabeza_cesens)
                            data2=js.dumps(datos1.json())
                            info2=js.loads(data2)
```

```
                            #Si hay un error no sigue
                            if(error1 not in info2 and info2!=error2
and info2!=error3):
                                #Se almacenan los datos necesarios en
el array principal
                                ubi=repr(item['id'])
                                met=info3['nombre'].replace(" ","")
                                metbien=info3['nombre']

filas.append([ubi,met.lower(),metbien])
        #Si el array principal está vacío no sigue
        if len(filas)!=0:
        #Se crea un nuevo archivo donde cargar los datos
            with open('metubis.json','w') as json_file:
                js.dump(filas, json_file, indent=4)
            #Se cambia el formato de .json a .csv
            path_json=r'C:/Users/jaime/Desktop/Database/metubis.json'
            read_file = pd.read_json(path_json)
            path_csv=r'C:/Users/jaime/Desktop/Database/metubis.csv'
            read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
            #Se borra el archivo .json
            os.remove('metubis.json')
        #Conexión con base de datos
        cursor=conexion.cursor()
        #Sentencia SQL para crear las tablas indicadas si no existían
antes
        tubicacion="CREATE TABLE IF NOT EXISTS metubis(idubicacion INT
,nombremetrica VARCHAR(50), metbien VARCHAR(100),FOREIGN
KEY(idubicacion) REFERENCES ubicacion_cesens,FOREIGN
KEY(nombremetrica) REFERENCES metricas_cesens)"
        cursor.execute(tubicacion)
        #Se borra el contenido de la tabla para no repetir filas
        borrar="DELETE FROM metubis"
        cursor.execute(borrar)
        #Se copia el contenido del .csv en la tabla SQL
        with
open('C:/Users/Jaime/Desktop/Database/metubis.csv','r',encoding="utf8"
) as f:
            next(f)
            cursor.copy_from(f,'metubis',sep=';')
        #Desconexión de base de datos
        cursor.close()
```

## 8.11. Código temperaturas.py

```
#######################################
##                                   ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ    ##
##                                   ##
#######################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener las temperaturas de las estaciones públicas
cercanas a Carralancha
def temperaturas(cabeza_cesens,conexion):
```

```
    carpeta='Temperaturas'
    #Si la carpeta ya existe, no la se vuelve a crear
    if os.path.isdir(carpeta)==False:
        os.mkdir(carpeta)
    #Cabecera de los .csv
    header = ('idubicacion','nombremetrica','fecha','valor')
    headerubis = ('id','nombre','descripcion', 'latitud', 'longitud')
    #Arrays de datos y URLs de las ubicaciones
    filas=[]
    urlmedinacyl = "https://app.cesens.com/api/datos/37/1/20220401"
    urlubimedina="https://app.cesens.com/api/ubicaciones/37"
    filasmedina=[]
    urlolmedocyl = "https://app.cesens.com/api/datos/189/1/20220401"
    urlubiolmedo="https://app.cesens.com/api/ubicaciones/189"
    filasolmedo=[]
    urlruedaaemet = "https://app.cesens.com/api/datos/984/1/20220401"
    urlubirueda="https://app.cesens.com/api/ubicaciones/984"
    filasrueda=[]
    #Obtención de las ubicaciones
    ubimedina=rqs.get(urlubimedina,headers=cabeza_cesens)
    datomedina=js.dumps(ubimedina.json())
    infoubimedina=js.loads(datomedina)
    ubiolmedo=rqs.get(urlubiolmedo,headers=cabeza_cesens)
    datoolmedo=js.dumps(ubiolmedo.json())
    infoubiolmedo=js.loads(datoolmedo)
    ubirueda=rqs.get(urlubirueda,headers=cabeza_cesens)
    datorueda=js.dumps(ubirueda.json())
    infoubirueda=js.loads(datorueda)
    #Guardado de los datos en los arrays
    id=infoubimedina['id']
    nombre=infoubimedina['nombre']
    descripcion=infoubimedina['descripcion']
    latitud=infoubimedina['latitud']
    longitud=infoubimedina['longitud']
    filas.append([id, nombre, descripcion, latitud, longitud])
    id=infoubiolmedo['id']
    nombre=infoubiolmedo['nombre']
    descripcion=infoubiolmedo['descripcion']
    latitud=infoubiolmedo['latitud']
    longitud=infoubiolmedo['longitud']
    filas.append([id, nombre, descripcion, latitud, longitud])
    id=infoubirueda['id']
    nombre=infoubirueda['nombre']
    descripcion=infoubirueda['descripcion']
    latitud=infoubirueda['latitud']
    longitud=infoubirueda['longitud']
    filas.append([id, nombre, descripcion, latitud, longitud])
    #Ordenación el vector
    filas.sort()
    #Se guardan en un archivo los datos ordenados
    with open('Ubicaciones_Temperaturas.json','w') as json_file:
        js.dump(filas, json_file, indent=4)
    #Se cambia el formato de .json a .csv
    read_file = pd.read_json
(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Temperaturas.json')
    read_file.to_csv
(r'C:/Users/jaime/Desktop/Database/Ubicaciones_Temperaturas.csv',
index = None,
header=headerubis,sep=";",decimal='.',float_format='%.5f')
    #Se borra el archivo .json
    os.remove('Ubicaciones_Temperaturas.json')
```

```
    #Se copia el archivo .csv a la carpeta indicada
    sh.copy('Ubicaciones_Temperaturas.csv', carpeta +
'/Ubicaciones_Temperaturas.csv')
    #Se borra el archivo .csv
    os.remove('Ubicaciones_Temperaturas.csv')
    #Conexión con base de datos
    cursor=conexion.cursor()
    #Sentencia SQL para crear las tablas indicadas si no existían
antes
    tubistemp="CREATE TABLE IF NOT EXISTS ubicacion_temperaturas(id
INT,nombre VARCHAR(50),descripcion VARCHAR(50),latitud FLOAT,longitud
FLOAT)"
    cursor.execute(tubistemp)
    try:
        #Se borra el contenido de la tabla para no repetir filas
        borrar="DELETE FROM ubicacion_temperaturas"
        cursor.execute(borrar)
        #Se copia el contenido del .csv en la tabla SQL
        with
open('C:/Users/jaime/Desktop/Database/Temperaturas/Ubicaciones_Tempera
turas.csv','r',encoding="utf8") as f:
            next(f)
            cursor.copy_from(f,'ubicacion_temperaturas',sep=';')
        #Desconexión de base de datos
        cursor.close()
    #Si la tabla ya existía no se hace nada
    except:
        print("Tabla ya existente")
    ubimedinacyl = rqs.get(urlmedinacyl, headers=cabeza_cesens)
    datamedinacyl = js.dumps(ubimedinacyl.json())
    infomedinacyl = js.loads(datamedinacyl)
    ubiolmedocyl = rqs.get(urlolmedocyl, headers=cabeza_cesens)
    dataolmedocyl = js.dumps(ubiolmedocyl.json())
    infoolmedocyl = js.loads(dataolmedocyl)
    ubiruedaaemet = rqs.get(urlruedaaemet, headers=cabeza_cesens)
    dataruedaaemet = js.dumps(ubiruedaaemet.json())
    inforuedaaemet = js.loads(dataruedaaemet)
    #Iteración para obtener los datos de las temperaturas
    for itemmedina in infomedinacyl:
        idubicacion=37
        nombremetrica='Temperatura'
        fecha_unix=itemmedina
        fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
        fecha=fecha_normal.strftime('%Y-%m-%d %H:%M:%S')
        valor=infomedinacyl[itemmedina]
        filasmedina.append([idubicacion,nombremetrica,fecha,valor])
    #Si el array está vacío no sigue
    if len(filasmedina)!=0:
        #Creación de un nuevo archivo donde cargar los datos
        with open('Temperatura_Medina_Inforiego.json','w') as
json_file:
            js.dump(filasmedina, json_file, indent=4)
        #Se cambia el formato de .json a .csv

path_json=r'C:/Users/jaime/Desktop/Database/Temperatura_Medina_Inforie
go.json'
        read_file = pd.read_json(path_json)

path_csv=r'C:/Users/jaime/Desktop/Database/Temperatura_Medina_Inforieg
o.csv'
```

```
        read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
        #Se borra el archivo .json
        os.remove('Temperatura_Medina_Inforiego.json')
        #Se copia el contenido del .csv en la tabla SQL
        sh.copy('Temperatura_Medina_Inforiego.csv', carpeta +
'/Temperatura_Medina_Inforiego.csv')
        #Se borra el archivo .csv
        os.remove('Temperatura_Medina_Inforiego.csv')
        #Conexión con base de datos
        cursor=conexion.cursor()
        #Sentencia SQL para crear las tablas indicadas si no existían
antes
        tmedina="CREATE TABLE IF NOT EXISTS
temperatura_medina(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float)"
        cursor.execute(tmedina)
        #Se borra el contenido de la tabla para no repetir filas
        borrar="DELETE FROM temperatura_medina"
        cursor.execute(borrar)
        #Se copia el contenido del .csv en la tabla SQL
        with
open('C:/Users/jaime/Desktop/Database/Temperaturas/Temperatura_Medina_
Inforiego.csv','r',encoding="utf8") as f:
            next(f)
            cursor.copy_from(f,'temperatura_medina',sep=';')
        #Desconexión de base de datos
        cursor.close()
    for itemolmedo in infoolmedocyl:
        idubicacion=189
        nombremetrica='Temperatura'
        fecha_unix=itemolmedo
        fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
        fecha=fecha_normal.strftime('%Y-%m-%d %H:%M:%S')
        valor=infoolmedocyl[itemolmedo]
        filasolmedo.append([idubicacion,nombremetrica,fecha,valor])
    if len(filasolmedo)!=0:
        with open('Temperatura_Olmedo_Inforiego.json','w') as
json_file:
            js.dump(filasolmedo, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/Temperatura_Olmedo_Inforie
go.json'
        read_file = pd.read_json(path_json)

path_csv=r'C:/Users/jaime/Desktop/Database/Temperatura_Olmedo_Inforieg
o.csv'
        read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
        os.remove('Temperatura_Olmedo_Inforiego.json')
        sh.copy('Temperatura_Olmedo_Inforiego.csv', carpeta +
'/Temperatura_Olmedo_Inforiego.csv')
        os.remove('Temperatura_Olmedo_Inforiego.csv')
        cursor=conexion.cursor()
        tolmedo="CREATE TABLE IF NOT EXISTS
temperatura_olmedo(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float)"
        cursor.execute(tolmedo)
        borrar="DELETE FROM temperatura_olmedo"
        cursor.execute(borrar)
```

```
        with
open('C:/Users/jaime/Desktop/Database/Temperaturas/Temperatura_Olmedo_
Inforiego.csv','r',encoding="utf8") as f:
            next(f)
            cursor.copy_from(f,'temperatura_olmedo',sep=';')
        cursor.close()
    for itemrueda in inforuedaaemet:
        idubicacion=984
        nombremetrica='Temperatura'
        fecha_unix=itemrueda
        fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
        fecha=fecha_normal.strftime('%Y-%m-%d %H:%M:%S')
        valor=inforuedaaemet[itemrueda]
        filasrueda.append([idubicacion,nombremetrica,fecha,valor])
    if len(filasrueda)!=0:
        with open('Temperatura_Rueda_AEMET.json','w') as json_file:
            js.dump(filasrueda, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/Temperatura_Rueda_AEMET.js
on'
        read_file = pd.read_json(path_json)

path_csv=r'C:/Users/jaime/Desktop/Database/Temperatura_Rueda_AEMET.csv
'
        read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
        os.remove('Temperatura_Rueda_AEMET.json')
        sh.copy('Temperatura_Rueda_AEMET.csv', carpeta +
'/Temperatura_Rueda_AEMET.csv')
        os.remove('Temperatura_Rueda_AEMET.csv')
        cursor=conexion.cursor()
        trueda="CREATE TABLE IF NOT EXISTS
temperatura_rueda(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float)"
        cursor.execute(trueda)
        borrar="DELETE FROM temperatura_rueda"
        cursor.execute(borrar)
        with
open('C:/Users/jaime/Desktop/Database/Temperaturas/Temperatura_Rueda_A
EMET.csv','r',encoding="utf8") as f:
            next(f)
            cursor.copy_from(f,'temperatura_rueda',sep=';')
        cursor.close()
```

## 8.12. Código datos_cesens.py

```
#######################################
##                                   ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ    ##
##                                   ##
#######################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener los datos de Cesens
def dato_cesens(cabeza_cesens,carpeta_cesens,conexion):
    #Se obtienen las ubicaciones de Cesens
```

```
    ubi=rqs.get("https://app.cesens.com/api/ubicaciones/",
headers=cabeza_cesens)
    data1=js.dumps(ubi.json())
    info=js.loads(data1)
    #Cadenas para obtener únicamente las ubicaciones de Carralancha y
Valdecovo
    nombre1="Carralancha"
    nombre2="Valdecovo"
    #Cadenas para identificar posibles errores o métricas vacias
    error1="msg"
    error2=[]
    error3="[]"
    #Array principal donde se guardarán los datos
    filas=[]
    #Cabecera a usar en el .csv
    header = ('idubicacion','nombremetrica','fecha','valor')
    #Bucle para iterar en las diferentes métricas
    for x in range (1,353):
        #Excepciones de métricas erróneas
        if(x!=90 and x!=163):
            #Se guardan los datos de las métricas
            new_url2 = 'https://app.cesens.com/api/metricas/' +
repr(x)
            metricas=rqs.get(new_url2, headers=cabeza_cesens)
            data3=js.dumps(metricas.json())
            info3=js.loads(data3)
            #Si hay un error no sigue
            if(error1 not in info3 and info3!=error2 and
info3!=error3):
                nombremetrica=info3['nombre'].replace(" ","")
                new_archivo = nombremetrica.lower()+ '.json'
                new_archivo_csv = nombremetrica.lower() + '.csv'
                for item in info:
                    #Si no es la ubicación deseada no sigue
                    if(nombre1 in item['nombre'] or nombre2 in
item['nombre']):
                        new_url1 = 'https://app.cesens.com/api/datos/'
+ repr(item['id']) + '/' + repr(x)
                        datos1=rqs.get(new_url1,
headers=cabeza_cesens)
                        data2=js.dumps(datos1.json())
                        info2=js.loads(data2)
                        #Si hay un error no sigue
                        if(error1 not in info2 and info2!=error2 and
info2!=error3):
                            for item4 in info2:
                                if(info2[item4] != None):
                                    #Se almacenan los datos necesarios
en el array principal
                                    idubi=repr(item['id'])
                                    nombremet=nombremetrica
                                    fecha_unix=item4

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                    fecha=fecha_normal.strftime('%Y-
%m-%d %H:%M:%S')
                                    valor=info2[item4]

filas.append([idubi,nombremet.lower(),fecha,valor])
                #Si el array principal está vacío no sigue
                if len(filas)!=0:
```

```
                    #Se crea un nuevo archivo donde cargar los datos
                    with open(new_archivo,'w') as json_file:
                        js.dump(filas, json_file, indent=4)
                    #Se cambia el formato de .json a .csv

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                    read_file = pd.read_json(path_json)

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv
                    read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                    #Se borra el archivo .json
                    os.remove(new_archivo)
                    #Se copia el archivo .csv a la carpeta indicada
                    sh.copy(new_archivo_csv, carpeta_cesens + '/' +
new_archivo_csv)
                    #Se borra el archivo de la ruta inicial
                    os.remove(new_archivo_csv)
                    #Liberación del array de datos
                    filas=[]
                    #Conexión con base de datos
                    cursor=conexion.cursor()
                    #Creación de tablas
                    nombretabla1=nombremet.lower()
                    nombretabla2=nombretabla1.replace("(","_")
                    nombretabla3=nombretabla2.replace(")","_")
                    nombretabla4=nombretabla3.replace("-","_")
                    #Sentencia SQL para crear las tablas indicadas si
no existían antes
                    tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla4+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_cesens,FOREIGN KEY(nombremetrica) REFERENCES
metricas_cesens)"
                    cursor.execute(tdato)
                    #Se borra el contenido de la tabla para no repetir
filas
                    borrar="DELETE FROM "+nombretabla4
                    cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_cesens+'/'+new_archivo
_csv
                    #Se copia el contenido del .csv en la tabla SQL
                    with open(path,'r',encoding="utf8") as f:
                        next(f)
                        cursor.copy_from(f,nombretabla4,sep=';')
                    #Desconexión de base de datos
                    cursor.close()
```

## 8.13. Código datos_modpow.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
```

```
#####################################
#Llamada para obtener todas las librerias necesarias
from imports import *
#Función para obtener los datos de Modpow
def dato_modpow(cabeza_modpow,carpeta_modpow,conexion):
    #Obtención de las ubicaciones de Modpow
    ubi=rqs.get("https://apiv2.wireless-
monitoring.net/ioecrops/devices",headers=cabeza_modpow)
    data1=js.dumps(ubi.json())
    info=js.loads(data1)
    #Obtención de la fecha actual
    fechaahora=dt.datetime.now()
    fechaactual=tm.mktime(fechaahora.timetuple())
    fechaactual=int(fechaactual)
    #Cabecera a usar en el .csv
    header = ('idubicacion','nombremetrica','fecha','valor')
    #Array donde se guardarán los datos
    filas=[]
    error="error"
    #Si el fichero existe, solo se recogerán los datos desde la fecha
contenida en este
    if os.path.isfile('UltimaEjecucion.txt')==True:
        #Apertura del fichero
        f=open('UltimaEjecucion.txt','r')
        #Lectura del fichero
        texto=f.read()
        #Cierre del fichero
        f.close()
        #Iteración de las distintas métricas
        for x in range (0,8):
            for item in info['devices']:
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                #Bucle para avanzar semana a semana
                while ts_unix<fechaactual:
                    #Cambio de formato de fechas
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    #Obtención de datos
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    #Si hay un error no sigue
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
```

```
                        #Iteración para almacenar los datos en el
array
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    #Actualización de las fechas
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            #Si el array principal está vacío no sigue
            if len(filas)!=0:
                #Creación de fichero temporal
                new_archivo=nombre+'x.json'
                #Guardado de datos en fichero temporal
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)
                    #Cambio de formato a .csv

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'
                new_archivo_csv2 = nombre+'x.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv2
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv2, carpeta_modpow + '/' +
new_archivo_csv2)
                os.remove(new_archivo_csv2)
                #Fusión del archivo temporal con el fichero principal
en el que están todos los datos
                df=pd.read_csv(carpeta_modpow + '/' +new_archivo_csv)
                df.head()
                csv_files=[carpeta_modpow + '/'
+new_archivo_csv,carpeta_modpow + '/' +new_archivo_csv2]
                files = [file for file in csv_files]

df1=pd.concat(map(pd.read_csv,files),ignore_index=True)
                df1.to_csv(new_archivo_csv,index=None)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(carpeta_modpow + '/' +new_archivo_csv2)
                os.remove(new_archivo_csv)
                #Reseteo de fechas
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                #Liberación del array de datos
                filas=[]
                #Conexión con base de datos
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
```

```
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                #Sentencia SQL para crear las tablas indicadas si no
existían antes
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                #Se copia el contenido del .csv en la tabla SQL
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                #Desconexión de base de datos
                cursor.close()
        for x in range (101,108):
            for item in info['devices']:
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
```

```
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'x.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'
                new_archivo_csv2 = nombre+'x.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv2
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv2, carpeta_modpow + '/' +
new_archivo_csv2)
                os.remove(new_archivo_csv2)
                df=pd.read_csv(carpeta_modpow + '/' +new_archivo_csv)
                df.head()
                csv_files=[carpeta_modpow + '/'
+new_archivo_csv,carpeta_modpow + '/' +new_archivo_csv2]
                files = [file for file in csv_files]

df1=pd.concat(map(pd.read_csv,files),ignore_index=True)
                df1.to_csv(new_archivo_csv,index=None)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(carpeta_modpow + '/' +new_archivo_csv2)
                os.remove(new_archivo_csv)
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (111,118):
            for item in info['devices']:
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
```

```
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'x.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'
                new_archivo_csv2 = nombre+'x.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv2
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv2, carpeta_modpow + '/' +
new_archivo_csv2)
                os.remove(new_archivo_csv2)
                df=pd.read_csv(carpeta_modpow + '/' +new_archivo_csv)
                df.head()
                csv_files=[carpeta_modpow + '/'
+new_archivo_csv,carpeta_modpow + '/' +new_archivo_csv2]
                files = [file for file in csv_files]

df1=pd.concat(map(pd.read_csv,files),ignore_index=True)
                df1.to_csv(new_archivo_csv,index=None)
```

```
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(carpeta_modpow + '/' +new_archivo_csv2)
                os.remove(new_archivo_csv)
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (121,128):
            for item in info['devices']:
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000
```

```
fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'x.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'
                new_archivo_csv2 = nombre+'x.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv2
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv2, carpeta_modpow + '/' +
new_archivo_csv2)
                os.remove(new_archivo_csv2)
                df=pd.read_csv(carpeta_modpow + '/' +new_archivo_csv)
                df.head()
                csv_files=[carpeta_modpow + '/'
+new_archivo_csv,carpeta_modpow + '/' +new_archivo_csv2]
                files = [file for file in csv_files]

df1=pd.concat(map(pd.read_csv,files),ignore_index=True)
                df1.to_csv(new_archivo_csv,index=None)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(carpeta_modpow + '/' +new_archivo_csv2)
                os.remove(new_archivo_csv)
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
```

```
        for x in range (201,205):
            for item in info['devices']:
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'x.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'
                new_archivo_csv2 = nombre+'x.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv2
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv2, carpeta_modpow + '/' +
new_archivo_csv2)
                os.remove(new_archivo_csv2)
                df=pd.read_csv(carpeta_modpow + '/' +new_archivo_csv)
                df.head()
```

```
                csv_files=[carpeta_modpow + '/'
+new_archivo_csv,carpeta_modpow + '/' +new_archivo_csv2]
                files = [file for file in csv_files]

df1=pd.concat(map(pd.read_csv,files),ignore_index=True)
                df1.to_csv(new_archivo_csv,index=None)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(carpeta_modpow + '/' +new_archivo_csv2)
                os.remove(new_archivo_csv)
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (211,215):
            for item in info['devices']:
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
```

```
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'x.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'
                new_archivo_csv2 = nombre+'x.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv2
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv2, carpeta_modpow + '/' +
new_archivo_csv2)
                os.remove(new_archivo_csv2)
                df=pd.read_csv(carpeta_modpow + '/' +new_archivo_csv)
                df.head()
                csv_files=[carpeta_modpow + '/'
+new_archivo_csv,carpeta_modpow + '/' +new_archivo_csv2]
                files = [file for file in csv_files]

df1=pd.concat(map(pd.read_csv,files),ignore_index=True)
                df1.to_csv(new_archivo_csv,index=None)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(carpeta_modpow + '/' +new_archivo_csv2)
                os.remove(new_archivo_csv)
                ts_unix=int(texto)
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)
```

```
path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
    #Si no existe se recogerán todos los datos
    else:
        for x in range (0,8):
            for item in info['devices']:
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv
```

```
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(new_archivo_csv)
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (101,108):
            for item in info['devices']:
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
```

```
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(new_archivo_csv)
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (111,118):
            for item in info['devices']:
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
```

```
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(new_archivo_csv)
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
```

```
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (121,128):
            for item in info['devices']:
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv
```

```
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(new_archivo_csv)
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (201,205):
            for item in info['devices']:
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
```

```
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(new_archivo_csv)
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
        for x in range (211,215):
            for item in info['devices']:
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                while ts_unix<fechaactual:
                    ts_normal=dt.datetime.fromtimestamp(ts_unix)
                    ts=ts_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
                    te_normal=dt.datetime.fromtimestamp(te_unix)
                    te=te_normal.strftime('%Y-%m-
%d'+'T'+'%H:%M:%S'+'.000Z')
```

```
                    url="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams"+'/'+repr(x
)+'?ts='+ts+'&te='+te
                    datos=rqs.get(url,headers=cabeza_modpow)
                    data2=js.dumps(datos.json())
                    info2=js.loads(data2)
                    if error not in info2:
                        meturl="https://apiv2.wireless-
monitoring.net/ioecrops/devices/"+item['id']+"/datastreams/"+repr(x)
                        metdatos=rqs.get(meturl,headers=cabeza_modpow)
                        data3=js.dumps(metdatos.json())
                        info3=js.loads(data3)
                        for itemxp in info3['datastreams']:
                            nombre=itemxp['label']
                            nombre=nombre.replace(" ","")
                            nombre=nombre.replace("/","-")
                            nombre=nombre.lower()
                        for item2 in info2['datastreams']:
                            for item3 in item2['values']:
                                id=item['id']
                                metrica=itemxp['label']
                                fecha_unix=int(item3)/1000

fecha_normal=dt.datetime.fromtimestamp(int(fecha_unix))
                                fecha=fecha_normal.strftime('%Y-%m-%d
%H:%M:%S')
                                valor=item2['values'][item3]
                                filas.append([id,metrica,fecha,valor])
                    ts_unix=te_unix+1
                    te_unix=te_unix+7*24*60*60
            if len(filas)!=0:
                new_archivo=nombre+'.json'
                with open(new_archivo,'w') as json_file:
                    js.dump(filas, json_file, indent=4)

path_json=r'C:/Users/jaime/Desktop/Database/'+new_archivo
                read_file = pd.read_json(path_json)
                new_archivo_csv = nombre+'.csv'

path_csv=r'C:/Users/jaime/Desktop/Database/'+new_archivo_csv
                read_file.to_csv (path_csv, index = None,
header=header,sep=";",decimal='.',float_format='%.5f')
                os.remove(new_archivo)
                sh.copy(new_archivo_csv, carpeta_modpow + '/' +
new_archivo_csv)
                os.remove(new_archivo_csv)
                ts_unix=1648764000
                te_unix=ts_unix+7*24*60*60
                filas=[]
                cursor=conexion.cursor()
                nombretabla1=nombre.lower()
                nombretabla2=nombretabla1.replace("(","_")
                nombretabla3=nombretabla2.replace(")","_")
                nombretabla4=nombretabla3.replace("-","_")
                nombretabla5='modpow_'+nombretabla4.replace(".","")
                tdato="CREATE TABLE IF NOT EXISTS
"+nombretabla5+"(idubicacion INT ,nombremetrica VARCHAR(50),fecha
TIMESTAMP,valor float, FOREIGN KEY(idubicacion) REFERENCES
ubicacion_modpow,FOREIGN KEY(nombremetrica) REFERENCES
metricas_modpow)"
                cursor.execute(tdato)
```

```
                borrar="DELETE FROM "+nombretabla5
                cursor.execute(borrar)

path='C:/Users/Jaime/Desktop/Database/'+carpeta_modpow+'/'+new_archivo
_csv
                with open(path,'r',encoding="utf8") as f:
                    next(f)
                    cursor.copy_from(f,nombretabla5,sep=';')
                cursor.close()
    f=open('UltimaEjecucion.txt','w')
    f.write(str(fechaactual))
    f.close()
```

## 8.14.Código database.bat

```
C:\Users\jaime\AppData\Local\Programs\Python\Python310\python.exe
C:\Users\jaime\Desktop\Database\main.py
```

## 8.15. Código sentinel.py

```
#######################################
##                                   ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ    ##
##                                   ##
#######################################
#Llamadas a todos los archivos para la ejecución del código
from sentinel_imports import *
from sentinel_NDVIcarralancha import *
from sentinel_NDVIvaldecovo import *
#Función para obtener las imagenes de Carralancha
NDVIcarralancha()
#Función para obtener las imagenes de Valdecovo
NDVIvaldecovo()
```

## 8.16. Código sentinel_imports.py

```
#######################################
##                                   ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ    ##
##                                   ##
#######################################
#Lamadas a las librerias para la ejecución del código
from sentinelhub import SHConfig
from sentinelhub import MimeType as mt
```

```
from sentinelhub import CRS as crs
from sentinelhub import BBox as bb
from sentinelhub import SentinelHubRequest
from sentinelhub import DataCollection as dc
from sentinelhub import Geometry as geo
from utils import plot_image as uti
import matplotlib.pyplot as plt
import datetime as dt
import os as os
import shutil as sh
#Abreviatura de las librerias para llamar a sus funciones
__all__ = [
    'SHConfig',
    'mt',
    'crs',
    'bb',
    'SentinelHubRequest',
    'dc',
    'geo',
    'uti',
    'plt',
    'dt',
    'os',
    'sh'
]
```

## 8.17. Código sentinel_NDVIcarralancha.py

```
########################################
##                                    ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ     ##
##                                    ##
########################################
#Llamada para obtener todas las librerias necesarias
from sentinel_imports import *
#Función para obtener las imagenes NDVI de Carralancha
def NDVIcarralancha():
    #Inicio de sesión en Sentinel
    config = SHConfig()
    config.sh_client_id= codigo secreto
config.sh_client_secret=codigo secreto
    #Si la carpeta ya existe, no se crea de nuevo
    carpeta_NDVI_carralancha='NDVI Carralancha'
    if os.path.isdir(carpeta_NDVI_carralancha)==False:
        os.mkdir(carpeta_NDVI_carralancha)
    #Fecha inicial desde la que obtener los datos
    fecha_inicio_unix = 1640991600
    fecha_actual_normal = dt.datetime.now().date().strftime('%Y-%m-
%d')
    fecha_actual_normal = fecha_actual_normal+' 00:00:00'
    fecha_actual_normal = dt.datetime.strptime(fecha_actual_normal,
'%Y-%m-%d %H:%M:%S')
    fecha_hoy_unix = int(dt.datetime.timestamp(fecha_actual_normal))
    #Bucle para obtener todas las imagenes
    while fecha_inicio_unix < fecha_hoy_unix:
        fecha_inicio = dt.datetime.fromtimestamp(fecha_inicio_unix)
```

```
        dia_normal = fecha_inicio.strftime('%Y-%m-%d')
        #Cuerpo de la función obtenido desde SentinelHub
        evalscript = """
        //VERSION=3
        //Author: Maxim Lamare

        function setup() {
            return {
            input: [
                {datasource: "S1GRD", bands:["VV", "VH"]},
                {datasource: "S2L2A", bands:["B08", "B04", "SCL"]}],
            output: [
                {id: "default", bands: 3, sampleType: SampleType.AUTO}
            ]
            }
        }

        function toDb(linear) {
            // Convert the linear backscatter to DB (Filgueiras et al.
(2019), eq. 3)
            return 10 * Math.LN10 * linear
        }

        function calc_s1_ndvi(sigmaVV, sigmaVH){
            // Convert sigma0 to Decibels
            let vh_Db = toDb(sigmaVH)
            let vv_Db = toDb(sigmaVV)

            // Calculate NRPB (Filgueiras et al. (2019), eq. 4)
            let NRPB = (vh_Db - vv_Db) / (vh_Db + vv_Db)

            // Calculate NDVI_nc with approach A3 (Filgueiras et al.
(2019), eq. 14)
            let NDVInc = 2.572 - 0.05047 * vh_Db + 0.176 * vv_Db +
3.422 * NRPB

            return NDVInc
        }

        function evaluatePixel(samples) {
            var s1 = samples.S1GRD[0]
            var s2 = samples.S2L2A[0]

            // Create an NDVI visualiser
            var viz=new ColorMapVisualizer([[0.0,0xa50026],
                                            [0.1,0xd73027],
[0.2,0xf46d43],
                                            [0.3,0xfdae61],
[0.4,0xfee08b],
                                            [0.5,0xffffbf],
[0.6,0xd9ef8b],
                                            [0.7,0xa6d96a],
[0.8,0x66bd63],
                                            [0.9,0x1a9850],
[0.9,0x1a9850], [1.0,0x000000]]);
            // Calculate S2 NDVI
            let ndvi = index(s2.B08, s2.B04)
            // Calculate S1 NDVI
            let s1_ndvi = calc_s1_ndvi(s1.VV, s1.VH)

            // Use the S2-L2A classification to identify clouds
```

```
            if ([7, 8, 9, 10].includes(s2.SCL)) {
            // If clouds are present use S1 NDVI
            return {
                default: viz.process(s1_ndvi)
            }
            } else {
            // Otherwise use s2 NDVI
            return {
                default: viz.process(ndvi)
            }
            }
        }
        """
        bbox = bb(bbox=[-4.818271, 41.370497, -4.806933, 41.377356],
crs=crs.WGS84)
        geometry = geo(geometry={"type":"Polygon","coordinates":[[[-
4.81054,41.377356],[-4.818271,41.372686],[-4.815587,41.370497],[-
4.810003,41.372558],[-4.808028,41.373427],[-4.806933,41.374184],[-
4.81054,41.377356]]]}, crs=crs.WGS84)
        request = SentinelHubRequest(
            evalscript=evalscript,
            input_data=[
                SentinelHubRequest.input_data(
                    data_collection=dc.SENTINEL1_IW,
                    identifier='S1GRD',
                    time_interval=(dia_normal, dia_normal),
                ),
                SentinelHubRequest.input_data(
                    data_collection=dc.SENTINEL2_L2A,
                    identifier='S2L2A',
                    time_interval=(dia_normal, dia_normal),
                ),
            ],
            responses=[
                SentinelHubRequest.output_response('default', mt.PNG),
            ],
            bbox=bbox,
            geometry=geometry,
            size=[512, 412.735],
            config=config
        )
        response = request.get_data()
        #Creación de las imágenes
        image = response[0]
        uti(image, factor=1/255, clip_range=(0, 1))
        png='NDVI Carralancha '+dia_normal+'.png'
        plt.savefig(png)
        plt.close('all')
        #Guardado en carpeta de las imágenes
        sh.copy(png, carpeta_NDVI_carralancha + '/' + png)
        os.remove(png)
        #Actualización de la fecha
        fecha_inicio_unix = fecha_inicio_unix + 5*24*60*60
```

## 8.18. Código sentinel_NDVIvaldecovo.py

```
#####################################
##                                 ##
##  AUTOR: JAIME ANTOLÍN MARTÍNEZ  ##
##                                 ##
#####################################
#Llamada para obtener todas las librerias necesarias
from sentinel_imports import *
#Función para obtener las imagenes NDVI de Valdecovo
def NDVIvaldecovo():
    #Inicio de sesión en Sentinel
    config = SHConfig()
    config.sh_client_id= codigo secreto
    config.sh_client_secret= codigo secreto
    #Si la carpeta ya existe, no se crea de nuevo
    carpeta_NDVI_valdecovo='NDVI Valdecovo'
    if os.path.isdir(carpeta_NDVI_valdecovo)==False:
        os.mkdir(carpeta_NDVI_valdecovo)
    #Fecha inicial desde la que obtener los datos
    fecha_inicio_unix = 1640991600
    fecha_actual_normal = dt.datetime.now().date().strftime('%Y-%m-
%d')
    fecha_actual_normal = fecha_actual_normal+' 00:00:00'
    fecha_actual_normal = dt.datetime.strptime(fecha_actual_normal,
'%Y-%m-%d %H:%M:%S')
    fecha_hoy_unix = int(dt.datetime.timestamp(fecha_actual_normal))
    #Bucle para obtener todas las imagenes
    while fecha_inicio_unix < fecha_hoy_unix:
        fecha_inicio = dt.datetime.fromtimestamp(fecha_inicio_unix)
        dia_normal = fecha_inicio.strftime('%Y-%m-%d')
        #Cuerpo de la función obtenido desde SentinelHub
        evalscript = """
        //VERSION=3
        //Author: Maxim Lamare

        function setup() {
            return {
            input: [
                {datasource: "S1GRD", bands:["VV", "VH"]},
                {datasource: "S2L2A", bands:["B08", "B04", "SCL"]}],
            output: [
                {id: "default", bands: 3, sampleType: SampleType.AUTO}
            ]
            }
        }

        function toDb(linear) {
            // Convert the linear backscatter to DB (Filgueiras et al.
(2019), eq. 3)
            return 10 * Math.LN10 * linear
        }

        function calc_s1_ndvi(sigmaVV, sigmaVH){
            // Convert sigma0 to Decibels
            let vh_Db = toDb(sigmaVH)
            let vv_Db = toDb(sigmaVV)

            // Calculate NRPB (Filgueiras et al. (2019), eq. 4)
            let NRPB = (vh_Db - vv_Db) / (vh_Db + vv_Db)

            // Calculate NDVI_nc with approach A3 (Filgueiras et al.
(2019), eq. 14)
```

```
            let NDVInc = 2.572 - 0.05047 * vh_Db + 0.176 * vv_Db +
3.422 * NRPB

            return NDVInc
        }

        function evaluatePixel(samples) {
            var s1 = samples.S1GRD[0]
            var s2 = samples.S2L2A[0]

            // Create an NDVI visualiser
            var viz=new ColorMapVisualizer([[0.0,0xa50026],
                                            [0.1,0xd73027],
[0.2,0xf46d43],
                                            [0.3,0xfdae61],
[0.4,0xfee08b],
                                            [0.5,0xffffbf],
[0.6,0xd9ef8b],
                                            [0.7,0xa6d96a],
[0.8,0x66bd63],
                                            [0.9,0x1a9850],
[1.0,0x000000]]);
            // Calculate S2 NDVI
            let ndvi = index(s2.B08, s2.B04)
            // Calculate S1 NDVI
            let s1_ndvi = calc_s1_ndvi(s1.VV, s1.VH)

            // Use the S2-L2A classification to identify clouds
            if ([7, 8, 9, 10].includes(s2.SCL)) {
            // If clouds are present use S1 NDVI
            return {
                default: viz.process(s1_ndvi)
            }
            } else {
            // Otherwise use s2 NDVI
            return {
                default: viz.process(ndvi)
            }
            }
        }
        """
        bbox = bb(bbox=[-4.806565, 41.35816, -4.796736, 41.372139],
crs=crs.WGS84)
        geometry = geo(geometry={"type":"Polygon","coordinates":[[[-
4.803131,41.370271],[-4.797766,41.372139],[-4.797294,41.371881],[-
4.796736,41.363507],[-4.797551,41.360383],[-4.797336,41.359127],[-
4.79901,41.358805],[-4.800942,41.35816],[-4.805921,41.362477],[-
4.806565,41.362638],[-4.804504,41.364699],[-4.802702,41.364409],[-
4.803131,41.370271]]]}, crs=crs.WGS84)
        request = SentinelHubRequest(
            evalscript=evalscript,
            input_data=[
                SentinelHubRequest.input_data(
                    data_collection=dc.SENTINEL1_IW,
                    identifier='S1GRD',
                    time_interval=(dia_normal, dia_normal),
                ),
                SentinelHubRequest.input_data(
                    data_collection=dc.SENTINEL2_L2A,
                    identifier='S2L2A',
                    time_interval=(dia_normal, dia_normal),
```

```
            ),
        ],
        responses=[
            SentinelHubRequest.output_response('default', mt.PNG),
        ],
        bbox=bbox,
        geometry=geometry,
        size=[512, 412.735],
        config=config
    )
    response = request.get_data()
    #Creación de las imágenes
    image = response[0]
    uti(image, factor=1/255, clip_range=(0, 1))
    png='NDVI Valdecovo '+dia_normal+'.png'
    plt.savefig(png)
    plt.close('all')
    #Guardado en carpeta de las imágenes
    sh.copy(png, carpeta_NDVI_valdecovo + '/' + png)
    os.remove(png)
    #Actualización de la fecha
    fecha_inicio_unix = fecha_inicio_unix + 5*24*60*60
```