

# Automatización Informes Grafana

## Requerimientos

Python 3.x

- apt install python3

Gestor de paquetes Python (pip)

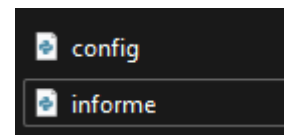
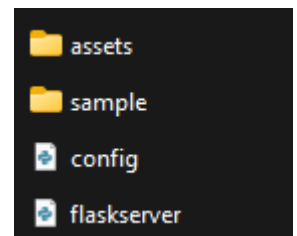
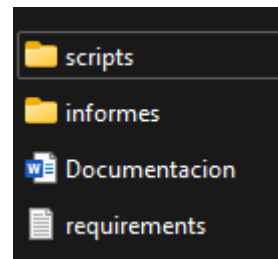
- apt install python3-pip

tkinter

- apt-get install python3-tk
- pip3 install -r requirements.txt /// pip install -r requirements.txt

## Organización de carpetas

- Scripts: están los archivos y carpetas encargados de la creación de los informes.
- Informes: aquí están todos los informes creados a partir del script.
- Requirements.txt: requerimientos para poder ejecutar el script. Se pueden instalar con el comando "*pip install -r requirements.txt*".
- Assets: aquí puedes poner las imágenes que quieras usar para tus informes.
- Sample: en esta carpeta está en script que crea el informe y el archivo de configuración para ese script. Puedes clonar esta carpeta las veces que quieras según el número de informes diferentes que quieras generar
- config.py (Flask): este archivo contiene la configuración para la API de Flask
- flaskserver.py: API para realizar peticiones y obtener los informes a partir de una petición HTTP.
- Config.py (informe): contiene la configuración para crear los informes. Abrá uno para cada carpeta de informes que se tenga.
- Informe.py: script que genera el excel con los datos de Grafana.



## Configuración Script

Para asegurar el funcionamiento correcto se recomienda solamente modificar los archivos `config.py`

Se recomienda en todo momento usar rutas absolutas para asegurar el correcto funcionamiento del script

Se deben modificar las constantes del archivo config según la configuración del Grafana al que se quiere hacer referencia y los dashboards de los cuales se quieran recoger datos.

```
ACTIVAR_SELECCION_RANGO_DE_FECHAS = False
DEBUG_FINAL = False
IMG = "[root]/Automatic-Grafana-Data-Report/scripts/assets/[sample].png"
DATA_DIR = "[root]/Automatic-Grafana-Data-Report/scripts/[sample]/data/"
# Informacion de autenticacion y URL base de Grafana
GRAFANA_URL = "http://[IP]:[PORT]"
API_KEY = ""

# Indica el rango de fechas desde hoy para el que se quieren recoger datos de Grafana
DAYS = 7

# Informacion CT
TITULO = ""
INFORMES_DIR = "[root]/Automatic-Grafana-Data-Report/informes/"+TITULO+"/"
```

ACTIVAR\_SELECCION\_RANGO\_DE\_FECHAS: Al estar en true cuando se ejecute el script manualmente se podrá seleccionar en una ventana el rango de fechas para los datos del informe.

DEBUG\_FINAL: Al estar en true se mostraran mensajes de debug en la consola.

IMG: ruta a la imagen que se quiera usar para el informe. En caso de que no se quiera usar imagen se debe dejar vacío ("").

DATA\_DIR: directorio donde se quieran crear los archivos de los datos recogidos en crudo. Se recomienda que se cree en la misma carpeta del script para evitar errores al tener mas informes diferentes.

GRAFANA\_URL: url del grafana que se quiera usar. Se debe indicar IP (o domino si quiere usar https) y puerto.

API\_KEY: clave API para poder establecer la conexión con la API de Grafana.


DAYS: Indica el rango de días desde hoy para el que se quieren recoger datos de Grafana.

TITULO: nombre que se quiera dar al informe.

INFORMES\_DIR: directorio que va a contener los excel que se generen con el script.

Los UIDS se corresponden a los dashboards donde se realizarán las consultas, debe están en el orden en que se desea que aparezcan en el informe. Estos se pueden encontrar fácilmente en la URL del navegador cuando te encuentras dentro de un dashboard.

```
UIDS = ("a52d2222-2b74-4f18-97a1-7c0ab04a1d6d", "fd901658-067b-492c-960e-87de2a2481fb",
        "a42945c7-7da4-4464-9f48-a6ecd4012845", "bdmgjotoo1pmob", "bdmgjfcsmbk00e", "fdmgmgi9ilh4wd")
```



El diccionario DASHBOARDS contiene la información de los dashboards y sus paneles de donde se recogerán los datos para mostrar en el informe. Deben estar en orden y concordar con el

orden de los UUIDS. Las claves del diccionario son un array con el título del dashboard y el texto que se quiere mostrar en su separador del Excel. El valor del diccionario es otro diccionario donde la clave es el id del panel correspondiente de donde se sacará información, y el valor es un array de 7 posiciones donde cada posición corresponde a una característica o opción de la información que saldrá de ese panel en el informe.

Las opciones son:

- Posición 1: título del panel.
- Posición 2: tipo de gráfico. “L” = Gráfico de líneas // “B” = Gráfico de barras
- Posición 3: tamaño del gráfico. “P” = Pequeño // “M” = Mediano // “G” = Grande
- Posición 4: indica si los datos recogidos son binarios (0, 1) o no. “True” // “False”
- Posición 5: indica si se desea que en el gráfico aparezca la leyenda o no. “True” // “False”
- Posición 6: indica si se desea información extra de los datos recogidos y si es así que tipo de información. “” = Sin información extra // “INFO” = Añade el texto deseado si no se ha recogido ningún dato // “MAXMIN” = Muestra los valores máximos y mínimos de la primera serie recogida del panel // “TABLE” = Muestra una tabla con los valores máximos y mínimos de todas las series recogidas del panel.
- Posición 7: texto adicional que se muestra de una forma u otra dependiendo de la opción escogida en la posición 6.

```
# CLAVE:[NOMBRE, TIPO(LINEAS, BARRAS), TAMAÑO(PEQUEÑO, MEDIANO, GRANDE), BINARIO(True, False), LEYENDA(True, False), EXTRA(MAXMIN, INFO, TABLA), EXTRA_INFO(mensaje, titulo tablas), EXTRA_INFO(unidad)]
DASHBOARDS = [{"Estado", "Datos de entorno del CT:": {17: ["Historial_Estados", "L", "G", True, True, "", ""],
4: ["Sensor_Puerta", "B", "M", True, False, "INFO", "No se ha detectado APERTURA de la Puerta"],
5: ["Sensor_Presencia", "B", "M", True, False, "INFO", "No se ha detectado PRESENCIA"],
1: ["Humedad_Interior", "L", "P", False, False, "MAXMIN", "Humedad", "%"],
2: ["Temperatura_Interior", "L", "P", False, False, "MAXMIN", "Temperatura", "Cº"],
3: ["Temperaturas", "L", "G", False, True, "TABLA", "Temperatura (Cº)"]},
("General", "Datos generales del CT:"): {5: ["Voltaje_ACUREV1_General", "L", "G", False, True, "TABLA", "Voltaje (V)"],
4: ["Intensidad_ACUREV1_General", "L", "G", False, True, "TABLA", "Intensidad (A)"],
6: ["Potencia_ACUREV1_General", "L", "G", False, True, "TABLA", "Potencia (kW)"],
7: ["Potencia_Reactiva_ACUREV1_General", "L", "G", False, True, "TABLA", "P. Reactiva (kVar)"]},
("Linea_1", "Datos Línea 1:"): {4: ["Intensidad_Linea_1", "L", "G", False, True, "TABLA", "Intensidad (A)"],
2: ["Potencia_Linea_1", "L", "G", False, True, "TABLA", "Potencia (kW)"],
3: ["Potencia_Reactiva_Linea_1", "L", "G", False, True, "TABLA", "P. Reactiva (kVar)"],
26: ["Potencia_Aparente_Linea_1", "L", "G", False, True, "TABLA", "P. Aparente (kVA)"]},
("Linea_2", "Datos Línea 2:"): {4: ["Intensidad_Linea_2", "L", "G", False, True, "TABLA", "Intensidad (A)"],
2: ["Potencia_Linea_2", "L", "G", False, True, "TABLA", "Potencia (kW)"],
3: ["Potencia_Reactiva_Linea_2", "L", "G", False, True, "TABLA", "P. Reactiva (kVar)"],
26: ["Potencia_Aparente_Linea_2", "L", "G", False, True, "TABLA", "P. Aparente (kVA)"]},
("Linea_3", "Datos Línea 3:"): {4: ["Intensidad_Linea_3", "L", "G", False, True, "TABLA", "Intensidad (A)"],
2: ["Potencia_Linea_3", "L", "G", False, True, "TABLA", "Potencia (kW)"],
3: ["Potencia_Reactiva_Linea_3", "L", "G", False, True, "TABLA", "P. Reactiva (kVar)"],
26: ["Potencia_Aparente_Linea_3", "L", "G", False, True, "TABLA", "P. Aparente (kVA)"]},
("Linea_4", "Datos Línea 4:"): {4: ["Intensidad_Linea_4", "L", "G", False, True, "TABLA", "Intensidad (A)"],
2: ["Potencia_Linea_4", "L", "G", False, True, "TABLA", "Potencia (kW)"],
3: ["Potencia_Reactiva_Linea_4", "L", "G", False, True, "TABLA", "P. Reactiva (kVar)"],
26: ["Potencia_Aparente_Linea_4", "L", "G", False, True, "TABLA", "P. Aparente (kVA)"]}]}
```

## Ejecución del script en Linux

En Linux para poder ejecutar el script correctamente primero se debe asegurar que los requisitos se cumplen.

Desde consola debes estar en la carpeta donde se guarda el script.

➤ `cd [/ruta/completa/a/tu/script/]`

Luego hay que darle permisos de ejecución.

➤ `chmod +x informe.py`

Por si acaso debemos formatear el archivo a un formato reconocible en Linux.

- apt install dos2unix
- dos2unix informe.py

Ahora ya podremos ejecutar el archivo en Linux desde consola.

- ./informe.py

También podemos automatizar su ejecución con cron:

- crontab -e

Añade la siguiente línea al final del archivo:

```
// H M DIA_MES MES DIA_SEMANA /usr/bin/python3 [/ruta/completa/]informe.py
```

- 0 0 \* \* 0 /usr/bin/python3 [/ruta/completa/]informe.py

Esto automatiza la ejecución para que se ejecute a las 0 horas, 0 minutos, \* cualquier día del mes, \* cualquier mes, 0 domingo (0-6 representa domingo a sábado)

## Ejecutar script desde API

Primero se debe rellenar el archivo config.py.

```
INFORMES_DIR = "[root]/Automatic-Grafana-Data-Report/informes/"
# INFORMES_DICT = {"dashboard_id":"/absolute/route/to/script.py"}
INFORMES_DICT = {"[sample]":"/[root]/Automatic-Grafana-Data-Report/scripts/[sample]/informe.py"}
```

INFORMES\_DIR: al igual que en el archivo config del script aquí se debe especificar la ruta donde se generan los scripts. Es recomendable que todos los informes diferentes que tengas apunten a la misma ruta para poder llegar a todos desde la API.

INFORMES\_DICT: diccionario donde la clave es el nombre o id para identificar el script y indicarlo cuando se haga la petición, mientras que el valor es la ruta al script correspondiente.

Luego:

- apt update
- pip3 install flask /// pip install flask
- apt install nginx

Crea un archivo de servicio systemd:

- nano /etc/systemd/system/flask\_server.service

---

[Unit]

Description=Servidor Flask para ejecutar script

After=network.target

---

[Service]

---

User=root

WorkingDirectory=/ruta/a/scripts/

ExecStart=/usr/bin/python3 /ruta/a/scripts/flaskserver.py

Restart=always

RestartSec=10

Environment="PORT=5000"

[Install]

WantedBy=multi-user.target

---

Habilita e inicia el servicio:

- systemctl daemon-reload
- systemctl start myflaskapp.service
- systemctl enable myflaskapp.service
- systemctl status myflaskapp.service

Crea un archivo de configuración

- nano /etc/nginx/sites-available/flask\_proxy
- 

server {

# Otras configuraciones existentes...

location /ejecutar\_script {

proxy\_pass http://localhost:5000;

proxy\_set\_header Host \$host;

proxy\_set\_header X-Real-IP \$remote\_addr;

proxy\_set\_header X-Forwarded-For \$proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto \$scheme;

}

}

---

- systemctl restart nginx

Para ejecutar el script y que se descargue el Excel hay que abrir el link [http://\[IP donde hayas establecido el servicio\]:5000/ejecutar\\_script/\[id\]](http://[IP donde hayas establecido el servicio]:5000/ejecutar_script/[id])