1. Servicio de systemd

Escribo un fichero de extensión *service* que ejecute mi programa escrito en *Python* tras el encendido y apagado del sistema. Más concretamente se ejecutará cuando haya red disponible. Esto lo hago empleando un target concreto, que espera hasta que la red esté conectada: *network-online.target*[1]. Debo comprobar que su servicio asociado esté correctamente iniciado, compruebo tanto el servicio de red de networkd como el gestor de redes que uso en mi computador que es *Networkmanager*:

```
[danih@~]$ systemctl is-enabled NetworkManager-wait-online.service systemd-networkd-wait-onl
ine.service
enabled
enabled
```

Figura 1: Correcto inicio de los servicios asociados a los target.

El fichero lo guardaré en */etc/systemd/system/Discord.service* y contendrá la siguiente información:

```
[Unit]
Description=Servicio de envío de mensaje de Discord en arranque
After=network-online.target
Wants=network-online.target

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/usr/bin/python3 /mnt/botDiscord.py 0
ExecStop=/usr/bin/python3 /mnt/botDiscord.py 1

[Install]
WantedBy=multi-user.target
```

Figura 2: /etc/systemd/system/Discord.service

- En unit indico la espera la conexión del sistema a red usando los campos *After* y *Wants*.
- Empleo el campo ExecStart para ejecutar el programa de inicio, que es el mismo ejecutable de *python* con un cero como parámetro.
- Empleo el campo ExecStop para ejecutar el programa a ejecutar antes del cierre del sistema, que es el mismo ejecutable de *python* con un uno como parámetro.
- En el campo WantedBy indico el target al que quiero asociar el servicio inicializado.

2. Bot de Discord

2.1. Programa en python

El programa que se encargará del envío del mensaje está escrito en *python*, la estructura de comunicación con Discord la he encontrado en una web[2]. La complejidad del programa radica en el envío de mensajes a un grupo de Discord, la parte redactada por mi es muy simple:
Dependiendo del primer parámetro pasado en la ejecución del programa, guardo una cadena que notifique el arranque o apagado del sistema. Luego concateno a esta cadena los datos horarios y hago el envío.

[1] `https://www.freedesktop.org/wiki/Software/systemd/NetworkTarget/`
[2] `https://aneshkesavan.com/how-to-send-messages-to-discord-server-with-python/`

```
dia = datetime.datetime.now()
if sys.argv[1] == '0':
    estado = "El sistema se ha arrancado: "
else:
    estado = "El sistema se ha apagado: "
mensaje = estado + str(dia)
```

Figura 3: Sección de código que determina el mensaje.

El envío del mensaje lo hago empleando la librería *requests*. Ejecutando el método "*post*", con la url y la estructura con el mensaje como parámetros, envío el mensaje al *Webhook* de *Discord*. El código de "*try*"posterior se utiliza solo para el depurado, dado que lo estamos ejecutando con systemd podríamos cortarlo. Pero sirvió de gran ayuda durante las pruebas iniciales.

```
url = "https://discord.com/api/webhooks/948910007146143784/KyCOCSsFdsFdzEWvp43dOK_qooHI5UDi358fwFQCPSyJM2F6_4903A5f-JRgT7heX3DJ"
```

Figura 4: Guardado de la URL en una variable de *Python*.

```
result = requests.post(url, json = data)

try:
    result.raise_for_status()
except requests.exceptions.HTTPError as err:
    print(err)
else:
    print("Mensaje entregado correctamente, cod {}.".format(result.status_code))
```

Figura 5: Envío del mensaje mediante la librería requests y su método post.

Adjunto el código del programa completo a continuación, aunque lo incluiré comprimido en la entrega.

```python
import requests
import sys
import datetime

url = "https://discord.com/api/webhooks/948910007146143784/KyCOCSsFdsFdzEWvp43dOK_qooHI5
UDi358fwFQCPSyJM2F6_4903A5f-JRgT7heX3DJ"
dia = datetime.datetime.now()
if sys.argv[1] == '0':
    estado = "El sistema se ha arrancado: "
else:
    estado = "El sistema se ha apagado: "
mensaje = estado + str(dia)
data = {
    "content" : str(mensaje),

    "username" : "BotSystemdAS"
}


result = requests.post(url, json = data)

try:
    result.raise_for_status()
except requests.exceptions.HTTPError as err:
    print(err)
else:
    print("Mensaje entregado correctamente, cod {}.".format(result.status_code))
```

Figura 6: Programa escrito en Python.

2.2. Configuración del canal del servidor

Antes de poder ejecutar este programa, necesito una dirección a la que enviar la información. Esta dirección la consigo desde Discord, añadiendo a un canal de texto cualquiera un *Webhook* y copiando la dirección de este. Para ello accedo a las opciones de configuración del servidor de discord.

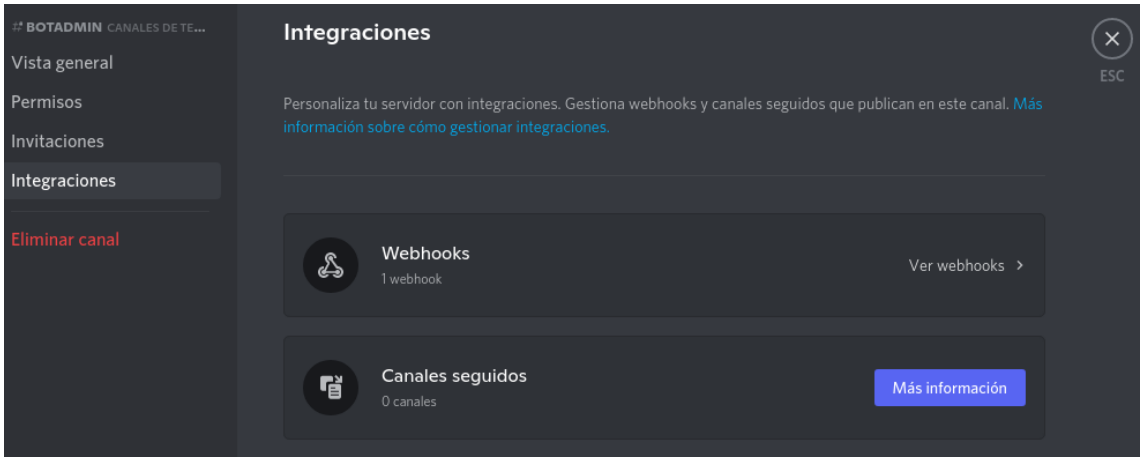

Figura 7: Ventana de configuración de mi servidor de Discord.

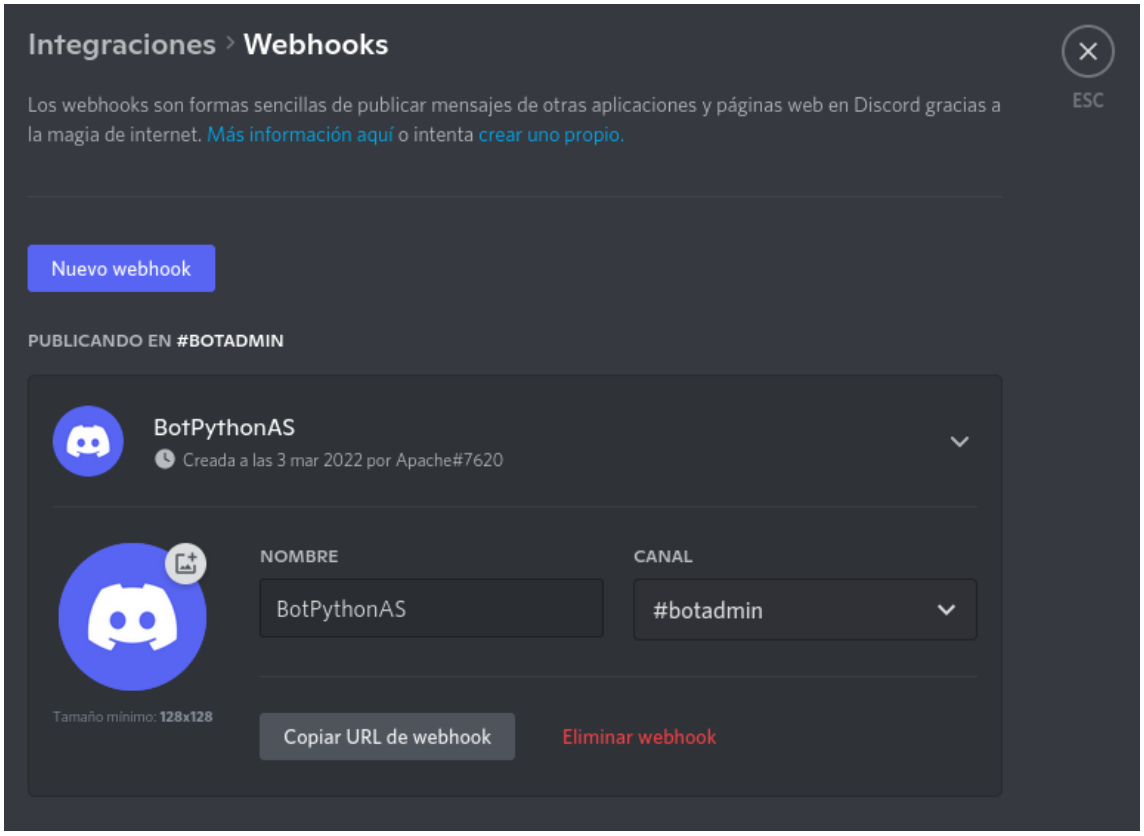

Figura 8: Ventana de configuración de mi servidor de Discord.

Con este enlace en mano ya podemos efectuar el envío de mensajes desde nuestro programa *python*. Dicho enlace identificará el punto de acceso abierto en el servidor.

3. Funcionamiento

Para que el script se ejecute automáticamente, deberemos iniciar el servicio de *systemd* que hemos escrito previamente. Para ello emplearemos la herramienta *systemctl*, ejecutada como administrador con la opción: enable. No deberemos preocuparnos por la ruta de nuestro fichero de servicio, puesto que desde un primer momento lo guardamos en una carpeta conocida por *systemd* y solo tendremos que ejecutar los siguiente:

```
[danih@BotDiscord]$ sudo systemctl enable Discord
[sudo] contraseña para danih:
Created symlink /etc/systemd/system/multi-user.target.wants/Discord.service → /etc/systemd/s
ystem/Discord.service.
```

De esta manera queda finalizada la configuración a realizar. Podemos comprobar su correcto funcionamiento apagando, encendiendo y volviendo a apagar el equipo. Adjunto captura del primer funcionamiento adecuado.