Universidad de Guadalajara.

Centro Universitario de Ciencias Exactas e Ingenierías.

Ejercicio 5

Estatus

Computación tolerante a fallas NRC 179961

Nombre del alumno: Loredo Padilla Orlando Javier

Código: 217560328

Departamento: Departamento de ciencias computacionales

Nombre del maestro: López Franco Michel Emanuel

Carrera: Ingeniería en computación

Sección: D06

Ciclo: 23B

Fecha: 18/9/2023.

Introducción

Realizar un programa que sea capaz de revisar el estado de tu aplicación.

Programa

El programa de esta práctica hará una manipulación de los servicios y procesos en el equipo. Lo se hizo fue un programa que revise los estados de dos aplicaciones y verifique si están activadas o no, una funcionará como la principal y la segunda como la dependiente de la primera.

En esta parte del código primero enviaremos cual será el programa principal, de aquí se llamarán a dos funciones que verificarán todos los servicios en ejecución.

```
if __name__ == '__main__':
    check_arguments()
    targets = get_targets()
    while True:
        for target in targets:
            lock(target)
        print("#########################################################")
        time.sleep(25) #25 segundos para testear
        #time.sleep(300) #5 minutos para el programa
```

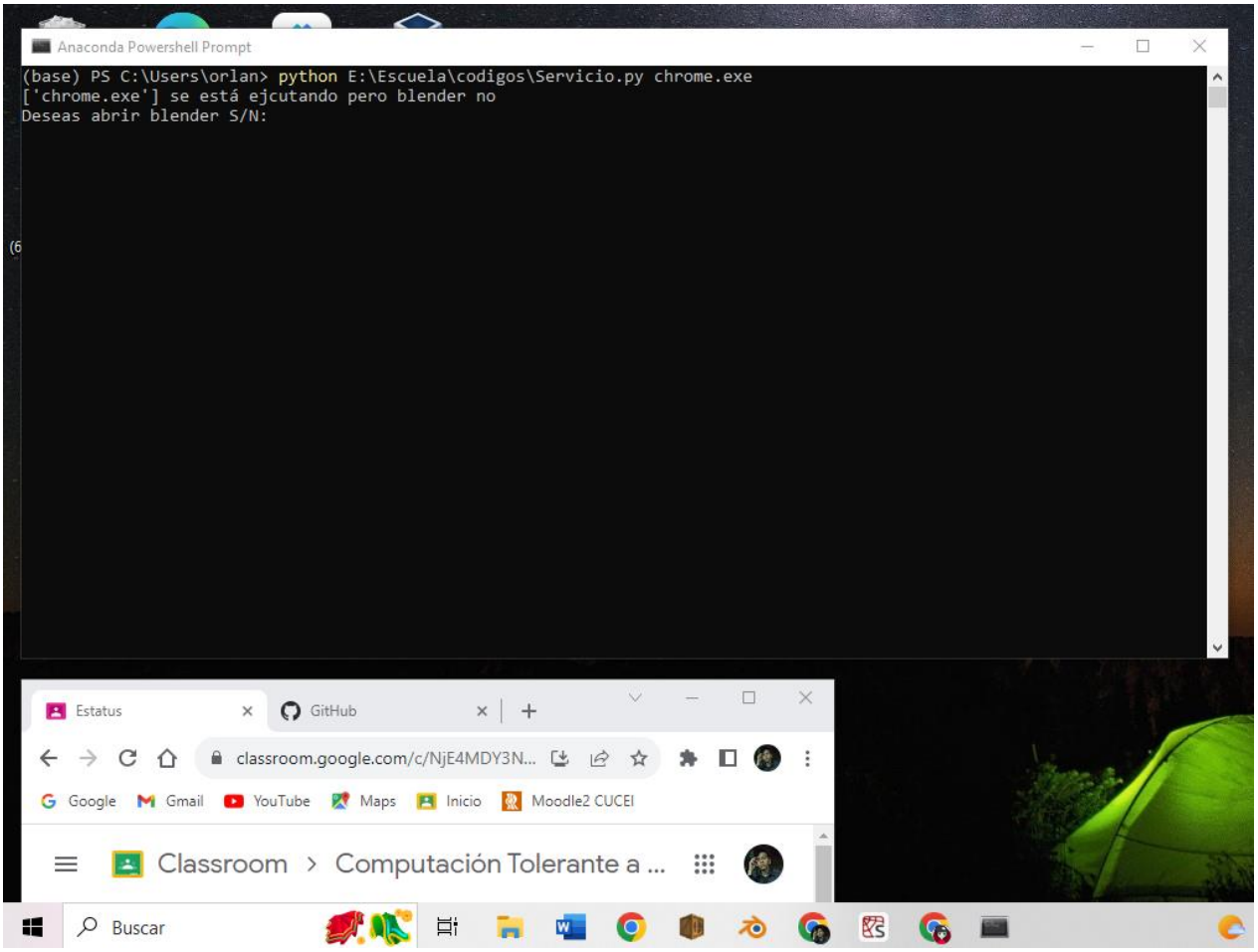
```
def check_arguments():
    if len(sys.argv) == 1:
        print('Este programa no funciona sin argumentos')
        sys.exit(0)

def get_targets():
    targets = sys.argv[1:]
    i = 0
    while i < len(targets):
        if not targets[i].endswith('.exe'):
            targets[i] = targets[i] + '.exe'
        i += 1
    return targets
```

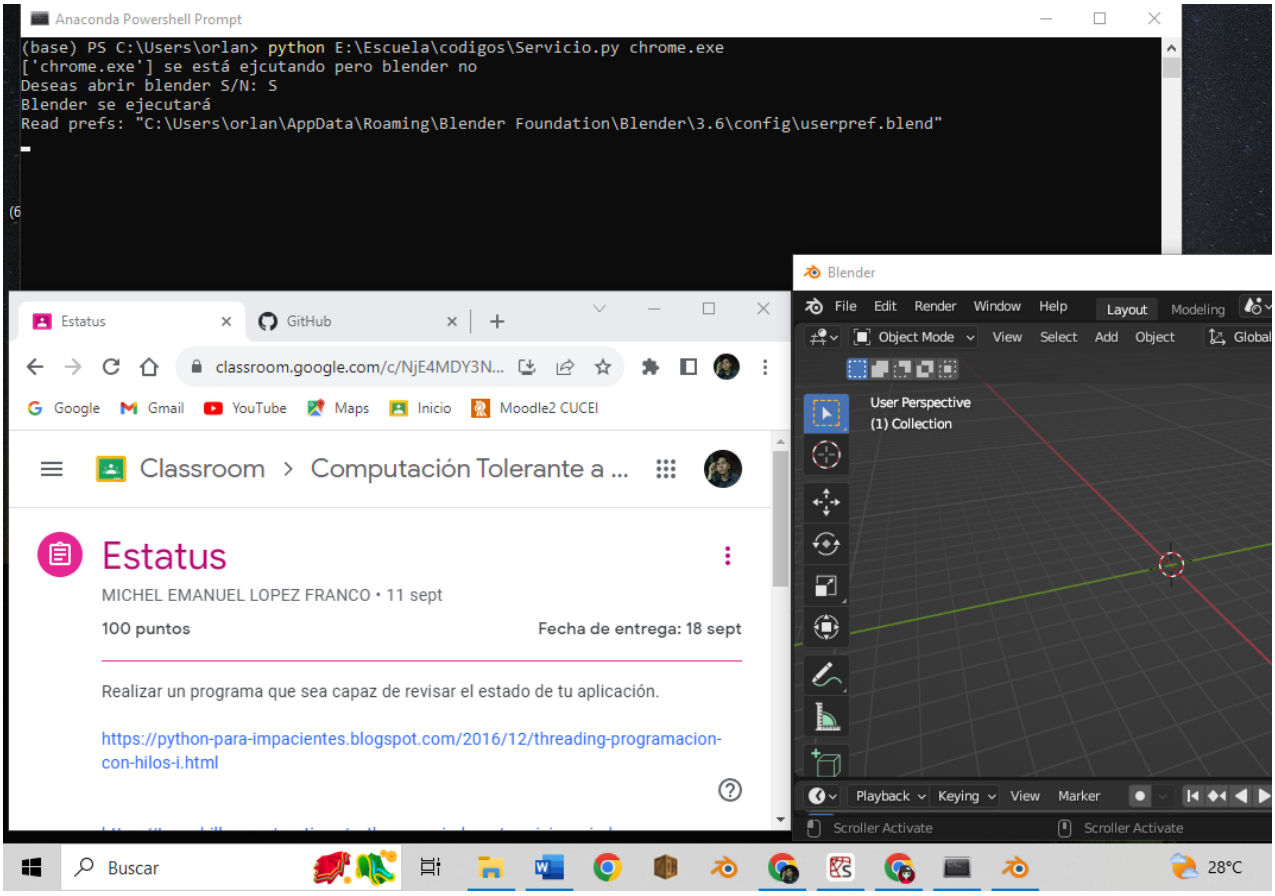
Una vez que recibamos la respuesta de las funciones llamaremos a otra función que hará todo el trabajo, aquí revisaremos si nuestra función principal está activa, en caso de estarlo revisará el estado de la segunda función, si se da la situación de que no se está ejecutando nos enviará la notificación y nos preguntará si queremos ejecutar, si respondemos que, si el programa se abrirá inmediatamente, de lo contrario seguirá ejecutándose el programa y preguntará cada cierto tiempo.

```
def lock(target):
    j = 0
    h = 0
    respuesta = ' '
    for proc in psutil.process_iter():
        #Caso donde el programa solicitado se esta ejecutando
        if proc.name().lower() == target.lower():
            j += 1
            for proc in psutil.process_iter():
                  #Caso donde blender se esta ejecutando
                  if proc.name().lower() == 'blender.exe':
                      h += 1
                      if h == 1:
                          print(targets, "y blender se estan ejecutando")
                          respuesta = input('Deseas cerrar blender S/N: ')
                          #El programa se cerrará
                          if respuesta == "S":
                              proc.kill()
                              print("Blender se cerró")
                          #El programa se ejecutará
                          if respuesta == "N":
                              print("Blender seguirá activado")
            #Caso donde nuestro programa se esta ejecutando pero blender no
            if h == 0:
                h += 2
                print(targets, "se está ejcutando pero blender no")
                respuesta = input('Deseas abrir blender S/N: ')
                #El programa se ejecutará
                if respuesta == "S":
                    print("Blender se ejecutará")
                    subprocess.run(r"E:\Orlando\Programas\Blender\blender.exe")
                #El programa no se ejecutará
                if respuesta == "N":
                    print("Blender seguirá desactivado")
    #Caso donde el programa ni blender se estan ejecutando
    if j == 0:
        print(targets, "ni blender se estan ejecutando")
```

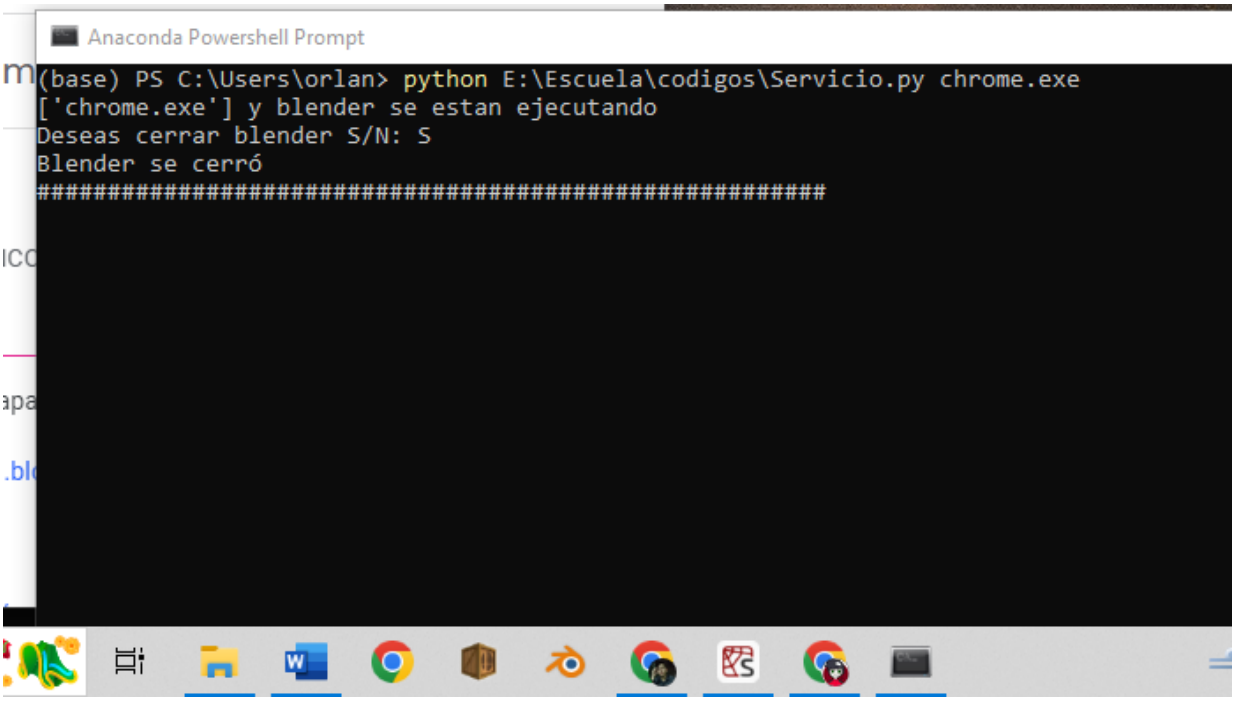
Si el programa se estaba ejecutando antes de ejecutar el código nos preguntará si queremos cerrar el programa. Aquí se muestra la demostración del programa en ejecución.



El programa no esta abierto por lo cual diremos que sí. Acto seguido se abrirá.



Y ahora mostraremos cuando el programa está abierto desde antes. El programa al ciclarse luego nos permitirá abrir de nuevo el segundo programa.



Conclusión

Para un usuario los servicios son cosas con las que normalmente nunca interactuaremos, por lo cual los pasaremos por alto la mayor parte del tiempo. Los servicios se tratan de todo programa que se estará ejecutando sin que nosotros estemos involucrados en su activación, se encargan de tareas en las cuales no se necesita de la participación del usuario.

Lo que nosotros hicimos en esta práctica fue simular el comportamiento de estos para entenderlos mejor, aunque el programa haya sido sencillo se pudo apreciar un poco mejor de su funcionamiento y si se sigue trabajando en estos verdaderamente se podría crear un programa que se trate de un servicio que realice alguna función importante para ayudar al equipo en la ejecución de tareas y reducir nuestro trabajo.

Referencias

- Tecnobillo (2022). Crear Servicios para Windows con Python en Tecnobillo. Recuperado el 12/9/2023 en: https://tecnobillo.com/sections/python-en-windows/servicios-windows-python/servicios-windows-python.html