A blue logo with a globe and text

Description automatically generated

**Materia:**

Programación computacional III

**Docente:**

William Alexis Montes Girón

**Actividad:**

Exposición sobre Watchdog

**Grupo de clases:**

Administración de base de datos II

Grupo A2.

**Alumno:**

Saul Antonio Amaya Umanzor.

Yerovi Josué Martínez Gómez.

Diego Steven Montoya Castro.

Cristian Alexis Velásquez Hernández

**Facultad:**

Ciencias y tecnología.

**Fecha de entrega:**

20/10/2025

**Año:**

2025

Contenido

[Introducción: 3](#_Toc211881624)

[¿En que consiste la librería y para que se usa? 4](#_Toc211881625)

[¿En que consiste la librería Watchdog? 4](#_Toc211881626)

[¿Para que se usa Watchdog? 4](#_Toc211881627)

[Funciones más relevantes y usadas de watchdog 4](#_Toc211881628)

[Ignorar archivos temporales de Word y otros .tmp 4](#_Toc211881629)

[Detectas archivos movidos o renombrados 5](#_Toc211881630)

[Detectar modificaciones a archivos 5](#_Toc211881631)

[Detectar creación de archivos 5](#_Toc211881632)

[Detectar eliminaciones de archivos 5](#_Toc211881633)

[Casos de uso de la librería en la vida real 6](#_Toc211881634)

[¿Qué es WIM? 6](#_Toc211881635)

[¿Cómo se relaciona WMI con watchdog? 7](#_Toc211881636)

[Ejemplo práctico 7](#_Toc211881637)

[Bibliografía 8](#_Toc211881638)

# Introducción:

Watchdog es una librería de Python diseñada para monitorear eventos en el sistema de archivos en tiempo real, facilitando la automatización de tareas ante cambios como creación, modificación o eliminación de archivos.

Es una herramienta especialmente útil cuando se necesita reaccionar automáticamente ante cambios en directorios o archivos, como en procesos de sincronización, actualización de registros, compilación automática o vigilancia de logs.

# ¿En que consiste la librería y para que se usa?

## ¿En que consiste la librería Watchdog?

Watchdog funciona mediante observadores(observers) y manejadores de eventos(evento handlers) siendo los observadores los procesos encargados de vigilar rutas especificas del sistema de archivo y los manejadores de eventos son clases que definen que hacer cuando ocurre un evento, como imprimir un mesaje, ejecutar una función o iniciar un proceso

## ¿Para que se usa Watchdog?

Watchdog se utiliza en distintos contextos donde se requiere de automatización o vigilancia de archivos. En este caso podemos mencionar los siguientes usos:

* Automatización de tareas mediante eventos: Por ejemplo podemos recompilar un código cuando se modifica un archivo y generar un registro de la modificación
* Monitoreo de logs: puede vigilar archivos de logs y si este presenta algún error u un tipo especifico de log se puede realizar una acción como por ejemplo enviar una alerta si una transacción fallo
* Sincronización y respaldo de información: en este caso podemos usar watchdog para guardar copias de seguridad a la hora de modificar un archivo

Funciones más relevantes y usadas de watchdog:

Este código son funciones de la librería las cuales normalmente se usan en una mismo función para que todas estén operando al mismo tiemp

## Ignorar archivos temporales de Word y otros .tmp

if '~WR' in event.src\_path or event.src\_path.endswith('.tmp'):

return

super().dispatch(event)

esta detecta los archivos temporales de word u otro tipo y los ignora, así vuelve más fácil el detectar los cambios que nosotros queremos monitorear

## Detectas archivos movidos o renombrados

def on\_moved(self, event):

print(f"renombrado de {event.src\_path} a {event.dest\_path}")

este detecta cuando un archivo es movido o renombrado y si detecta que eso ocurre genera un print

## Detectar modificaciones a archivos

def on\_modified(self, event):

print(f" Modificado: {event.src\_path}")

este detecta cuando un archivo es modificado y si detecta que eso ocurre genera un print

## Detectar creación de archivos

def on\_created(self, event):

print(f" Creado: {event.src\_path}")

este detecta la creación de nuevos archivos y si detecta que eso ocurre genera un print

## Detectar eliminaciones de archivos

def on\_deleted(self, event):

print(f" Eliminado: {event.src\_path}")

este detecta la eliminación de un archivo y si detecta que eso ocurrió genera un print

Casos de uso de la librería en la vida real:

* Se puede usar para automatizar copias de seguridad como si por ejemplo borramos algo, podamos crear una copia del archivo por si es necesaria luego u incluso podemos aplicarlo si modificamos un archivo y gracias a watchdog guardamos una copia del archivo antes de la modificación y la modificación sale mal, podemos volver a tener el archivo antes de la modificación.
* Monitorear actividad sospechosa en los archivos de nuestra computadora, podemos analizar si alguna aplicación realiza cambios en carpetas importantes y hacer que si se detecta actividad sospechosa genere algún bloqueo

==============================Importante==============================

Por fines de demostración se uso una segunda librería llamada WMI (Windows Management Instrumentation).

# ¿Qué es WIM?

Es una librería la cual permite acceder a la información del sistema operativo Windows como por ejemplo procesos, servicios, discos, usuarios, etc.

Ejemplo del uso e WMI para Verificar si el antivirus está activo y encenderlo si está apagado:

import wmi

c = wmi.WMI()

for s in c.Win32\_Service(Name="WinDefend"):

if s.State != "Running":

s.StartService()

## ¿Cómo se relaciona WMI con watchdog?

Combinar WMI y watchdog permite crear sistemas de monitoreo híbrido. Por ejemplo, puedes usar WMI para verificar si un proceso crítico está corriendo (como el antivirus), y watchdog para detectar si los logs de ese proceso están siendo modificados o eliminados. Esto permite correlacionar eventos del sistema con cambios en archivos, útil para auditoría, detección de incidentes, y automatización.

Ejemplo práctico: Verificar si el antivirus está activo y detectar si se borra su log:

import wmi

from watchdog.observers import Observer

from watchdog.events import FileSystemEventHandler

# Verificar estado del antivirus

c = wmi.WMI()

for s in c.Win32\_Service(Name="WinDefend"):

if s.State != "Running":

print("Antivirus desactivado")

# Detectar eliminación de log

class LogHandler(FileSystemEventHandler):

def on\_deleted(self, event):

print(f"Log eliminado: {event.src\_path}")

observer = Observer()

observer.schedule(LogHandler(), path="C:\\ProgramData\\Microsoft\\Windows Defender\\Support", recursive=False)

observer.start()

WMI es una librería muy interesante que se complementa bien con watchdog.

Bibliografía:

Diaz, R. (2020). *You tube*. Obtenido de Como usar watchdog: https://youtu.be/jyE-MJRwsqo?si=DXxEAWfZTedLY4Px

Mallón, X. (2024). *keepcoding*. Obtenido de https://keepcoding.io/blog/que-es-wmi-y-como-usarlo/

python. (2024). *Python watchdog*. Obtenido de https://python-watchdog.readthedocs.io/en/stable/index.html