Informe Técnico

Esteban Giraldo Llano,

Mariana Gutiérrez Jaramillo,

Ana Sofia Rodríguez Orozco

1. Descripción de la Arquitectura

El sistema está dividido en varias partes para facilitar su desarrollo y mantenimiento.

- En la carpeta core tenemos las funciones para buscar todos los archivos dentro de las carpetas que el usuario selecciona y para comprimir esos archivos en un archivo ZIP.
- En encryption está el código para encriptar y desencriptar el archivo de backup usando AES-256.
- En storage están los métodos para copiar el archivo a un disco externo, subirlo a Google Drive y dividirlo en partes para guardarlo en USB.
- En restore tenemos las funciones para restaurar el backup, es decir, descomprimir y desencriptar los archivos.
- Finalmente, en el archivo principal main.py se juntan todas estas partes para que el usuario pueda hacer el respaldo y restauración con una interfaz simple.

El proceso que sigue el programa es:

1. El usuario selecciona carpetas
2. Se obtienen todos los archivos
3. Se comprimen en un ZIP
4. Si quiere, se encripta
5. Luego puede copiar el archivo
6. Subirlo a la nube o dividirlo para USB
7. También puede restaurar cuando necesite

2. Uso de Dask para Paralelismo

Para que el programa sea rápido usamos Dask, que permite hacer varias cosas al mismo tiempo aprovechando los núcleos del procesador.

Paralelizamos al menos dos partes importantes:

- La compresión de archivos, donde varios archivos se comprimen simultáneamente.
- La encriptación, que se hace como una tarea paralela para no bloquear el programa.

Con Dask pudimos mejorar el rendimiento cuando hay muchos archivos grandes y acelerar las operaciones sin complicar mucho el código.

3. Algoritmo de Compresión y Biblioteca Usada

Para comprimir los archivos usamos el formato ZIP, porque es muy conocido y fácil de usar.

Utilizamos la librería estándar de Python llamada zipfile, que permite agregar muchos archivos en un solo archivo ZIP manteniendo las carpetas originales. Además, esta biblioteca es confiable y no requiere instalar nada extra.

Después de crear el archivo ZIP, si el usuario quiere, lo encriptamos para mayor seguridad.

4. Justificación del Lenguaje y Herramientas

Elegimos Python porque es un lenguaje muy versátil y con muchas librerías que nos facilitaron implementar todas las funciones requeridas sin tener que preocuparnos por detalles de bajo nivel.

Usamos Dask para paralelismo porque nos permitió manejar tareas concurrentes de forma sencilla y organizada, facilitando que el programa aproveche varios núcleos del procesador para acelerar la compresión, encriptación y transferencia, sin tener que gestionar hilos o procesos manualmente.

Para la compresión utilizamos la librería estándar zipfile, que es robusta y compatible con el formato ZIP ampliamente usado. Para la encriptación, usamos la biblioteca cryptography, que ofrece un método seguro y fácil de integrar (AES-256).

También empleamos Tkinter para la interfaz, ya que es simple y suficiente para la selección de carpetas y archivos, manteniendo la aplicación ligera y práctica.

Finalmente, la integración con Google Drive se hizo mediante su API, para cubrir la opción de almacenamiento en la nube solicitada.

Esta combinación nos permitió desarrollar un sistema funcional, eficiente y bien organizado, aprovechando las ventajas del lenguaje y sus herramientas, enfocándonos en la lógica del problema y sin complicar demasiado la implementación.