



# **Universidad Nacional Autónoma de México**



## **Facultad de Ingeniería**

**Semestre 2025-1**

### **Sistemas Operativos**

**Proyecto: (Micro) sistema de archivos multihilos**

**Profesor: Gunnar Eyal Wolf Iszaevich**

**Alumno: León Pérez Aarón Rodrigo**

**Fecha de entrega: 05/Noviembre/2024**

## Descripción General del Proyecto

El objetivo del proyecto es desarrollar un sistema de archivos simulado, llamado FiUnamFS, que permita realizar operaciones básicas de gestión de archivos. Este sistema es un proyecto académico diseñado para comprender mejor los conceptos de sistemas de archivos y concurrencia. Las funcionalidades incluyen:

1. Listar archivos: Mostrar el contenido del sistema de archivos.
2. Copiar un archivo de FiUnamFS al sistema local.
3. Copiar un archivo del sistema local a FiUnamFS.
4. Eliminar archivos dentro de FiUnamFS.
5. Sincronización entre hilos para gestionar operaciones concurrentes.

Este sistema fue desarrollado en Python y utiliza una interfaz gráfica simple, permitiendo la interacción con el sistema de archivos sin la necesidad de una terminal.

## Funcionalidades Implementadas

### 1. Listado de Archivos

El programa permite listar todos los archivos almacenados en **FiUnamFS**. Para cada archivo, se muestra:

- Nombre del archivo
- Tamaño del archivo
- Fecha de creación
- Fecha de modificación

Esta información es desplegada en un área de texto dentro de la interfaz gráfica.

### 2. Copiar Archivo de FiUnamFS al Sistema Local

Permite seleccionar un archivo en **FiUnamFS** y copiarlo al sistema de archivos local del usuario. La operación:

- Solicita el nombre del archivo en **FiUnamFS**.

- Verifica su existencia y copia el contenido al sistema local en la ruta indicada por el usuario.
- Informa si la copia se realiza con éxito o si el archivo no se encuentra.

### 3. Copiar Archivo del Sistema Local a FiUnamFS

Permite cargar un archivo desde el sistema local y almacenarlo en **FiUnamFS**. La operación:

- Solicita el archivo del sistema local.
- Verifica el tamaño y nombre del archivo para cumplir con las restricciones de FiUnamFS.
- Asigna espacio en **FiUnamFS** para almacenar el archivo, generando la información correspondiente en el directorio del sistema de archivos.

### 4. Eliminar Archivos en FiUnamFS

Permite eliminar archivos en **FiUnamFS**. La operación:

- Solicita el nombre del archivo a eliminar.
- Verifica su existencia y marca la entrada en el directorio de **FiUnamFS** como disponible.
- Informa si la eliminación se realiza con éxito o si el archivo no se encuentra.

### 5. Sincronización entre Hilos

La aplicación usa hilos para gestionar operaciones concurrentes, asegurando que las operaciones en **FiUnamFS** no interfieran entre sí. Se emplea un **Lock** (candado) en Python para manejar el acceso seguro a la estructura de datos compartida entre hilos.

## Entorno y Dependencias

### 1. Lenguaje de Programación

El proyecto fue desarrollado en **Python 3.9**. Asegúrate de tener Python 3.6 o superior, ya que se utilizan algunas características específicas de versiones recientes.

## 2. Bibliotecas

- **tkinter**: Para la interfaz gráfica de usuario.
- **threading**: Para manejar la concurrencia y sincronización entre hilos.
- **struct**: Para manejar las operaciones de lectura y escritura de datos binarios.
- **os**: Para operaciones en el sistema de archivos local.
- **datetime**: Para manejar fechas en la creación y modificación de archivos.

## Estructura del Código

El proyecto consta de los siguientes módulos principales:

- **Clase FiUnamFS**: La clase principal que contiene los métodos para listar, copiar y eliminar archivos, así como cargar el superbloque del sistema.
- **Funciones de Interfaz Gráfica**: Contiene las funciones que interactúan directamente con la interfaz gráfica de usuario, como `listar_archivos`, `copiar_a_sistema`, `copiar_a_fiunamfs`, y `eliminar_archivo`.
- **Interfaz Gráfica (tkinter)**: Diseño de la ventana principal con botones para cada una de las funcionalidades y un área de texto para mostrar el contenido del directorio.

## Estructura de Datos y Sincronización

### 1. Organización de los Archivos en FiUnamFS

**FiUnamFS** organiza los archivos mediante una estructura de directorio que utiliza 64 entradas de 64 bytes cada una. Cada entrada contiene:

- Tipo de archivo (1 byte)
- Nombre del archivo (15 bytes)
- Tamaño del archivo (4 bytes)
- Fecha de creación y modificación (13 bytes cada una)

El sistema de archivos es un **archivo binario** (fiunamfs.img) que emula un disco en el cual se realizan todas las operaciones.

## 2. Sincronización

Se utiliza un **Lock** de Python para sincronizar el acceso a los datos compartidos en la clase FiUnamFS. De esta forma, se asegura que dos operaciones de copia o eliminación no se realicen simultáneamente, evitando corrupción de datos.





## Instrucciones de Uso

1. **Verificar archivo .img:** Antes de cualquier operación, verifica que el archivo de imagen de **FiUnamFS** (fiunamfs.img) sea válido. Haz clic en “Verificar archivo .img” para que el sistema valide el superbloque del archivo.
2. **Listar Archivos:** Haz clic en “Listar Archivos” para ver el contenido del sistema de archivos. Se mostrará la lista de archivos con su tamaño, fecha de creación y fecha de modificación.
3. **Copiar a Sistema:** Haz clic en “Copiar a Sistema” y proporciona el nombre del archivo a copiar y la carpeta de destino en el sistema local.
4. **Copiar a FiUnamFS:** Haz clic en “Copiar a FiUnamFS” y selecciona un archivo de tu sistema local para agregarlo a **FiUnamFS**. El archivo debe cumplir con los límites de nombre y tamaño.
5. **Eliminar Archivo:** Haz clic en “Eliminar Archivo” e ingresa el nombre del archivo que deseas eliminar de **FiUnamFS**.
6. **Cerrar Programa:** Usa el botón de “Salir” para cerrar la aplicación.

## Ejemplo de Uso

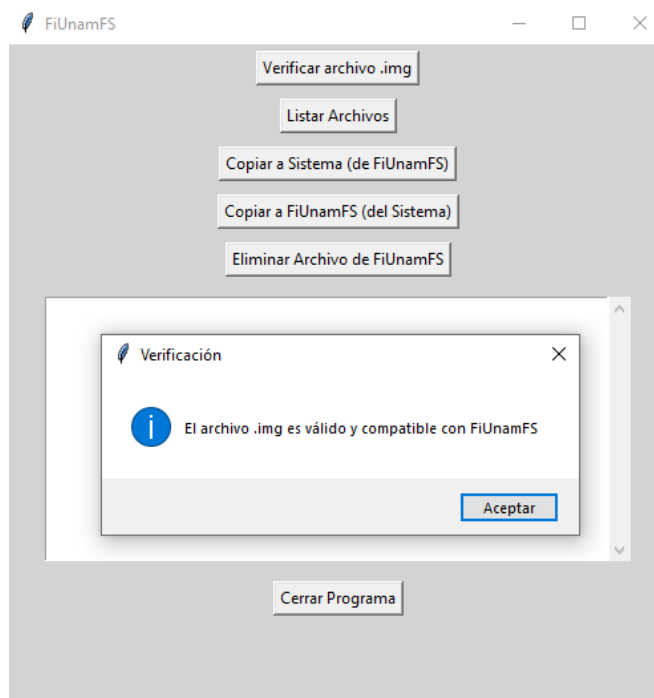
### Ejemplo 0: Documentos

Antes de nada, debes de tener el archivo **“fiunamfs.img”** y el archivo **“Proyecto2.py”** en la misma carpeta, se recomienda crear una carpeta aparte

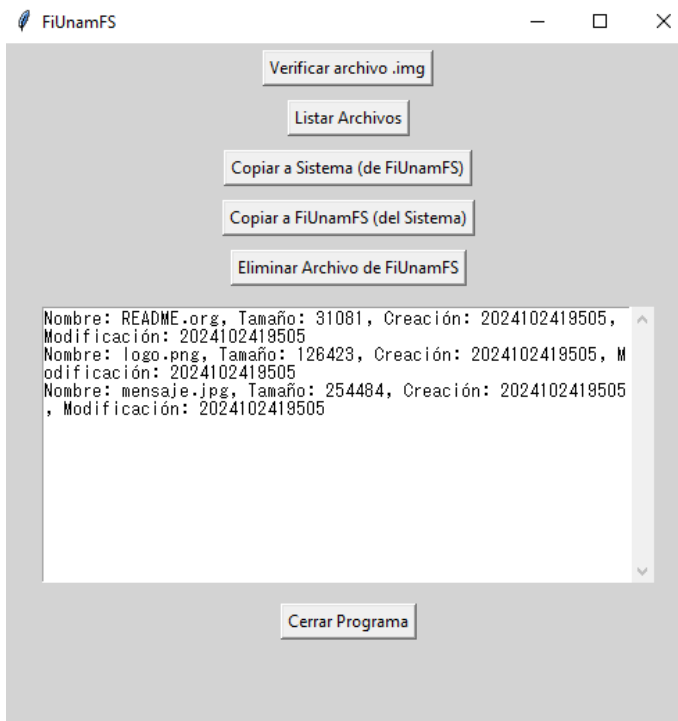
Nombre	Fecha de modificación	Tipo	Tamaño
 Fiptueba.py	03/11/2024 05:03 p. m.	Python File	7 KB
 fiunamfs.img	03/11/2024 05:04 p. m.	Archivo de image...	1,440 KB
 O1.txt	03/11/2024 03:40 p. m.	Documento de tex...	1 KB
 tempCodeRunnerFile.py	03/11/2024 04:26 p. m.	Python File	1 KB

### Ejemplo 1: Listar Archivos

1. Haz clic en "Verificar archivo .img" para asegurarte de que el sistema reconoce el archivo fiunamfs.img.

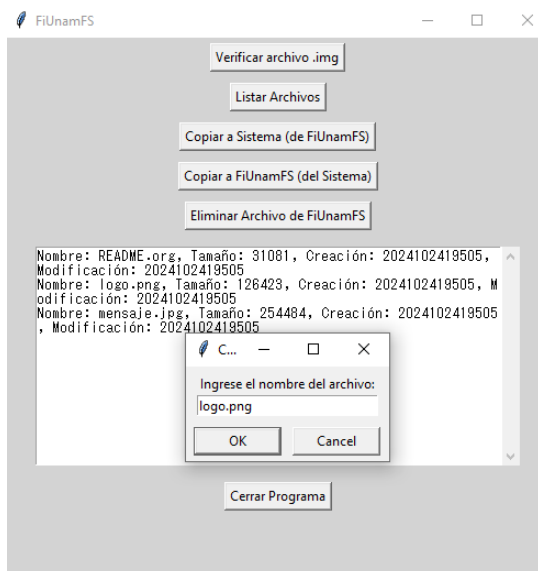


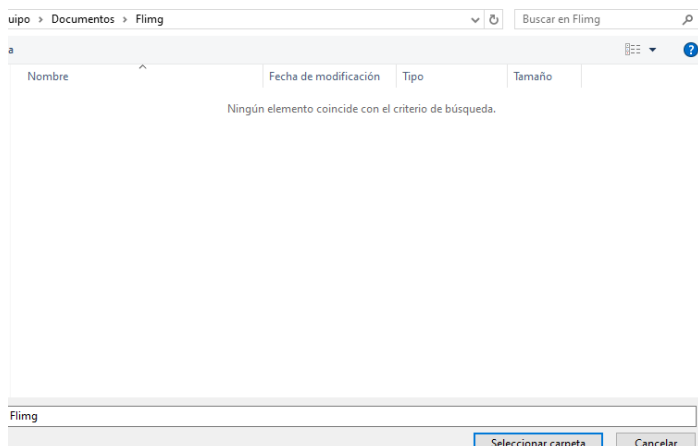
2. Haz clic en "Listar Archivos".
3. Los archivos disponibles en **FiUnamFS** aparecerán en el área de texto.



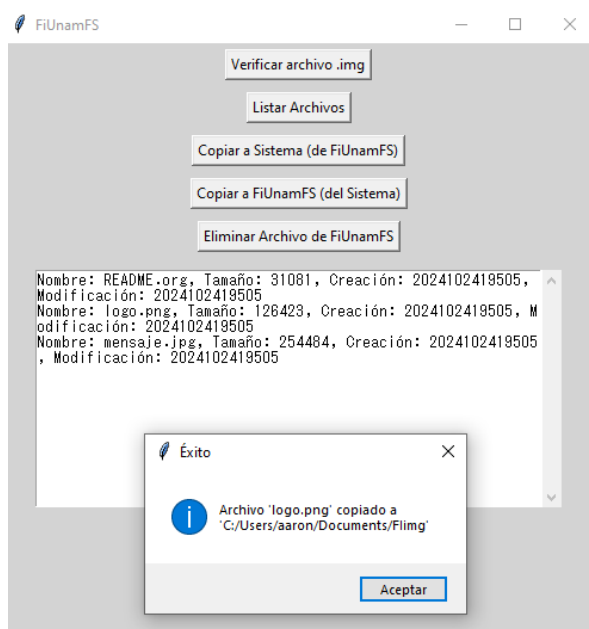
## Ejemplo 2: Copiar un archivo desde FiUnamFS al sistema local

1. Selecciona "Copiar a Sistema".
2. Ingresa el nombre del archivo y elige la carpeta de destino.





3. El sistema copiará el archivo al destino y mostrará un mensaje de éxito.







Nombre	Fecha de modificación	Tipo	Tamaño
Fiptueba.py	03/11/2024 05:03 p. m.	Python File	7 KB
fiunamfs.img	03/11/2024 05:04 p. m.	Archivo de image...	1,440 KB
logo.png	03/11/2024 07:55 p. m.	Archivo PNG	124 KB
O1.txt	03/11/2024 03:40 p. m.	Documento de te...	1 KB
tempCodeRunnerFile.py	03/11/2024 04:26 p. m.	Python File	1 KB

Con esto podemos observar todos los archivos que se encontraban en fiunamfs.img



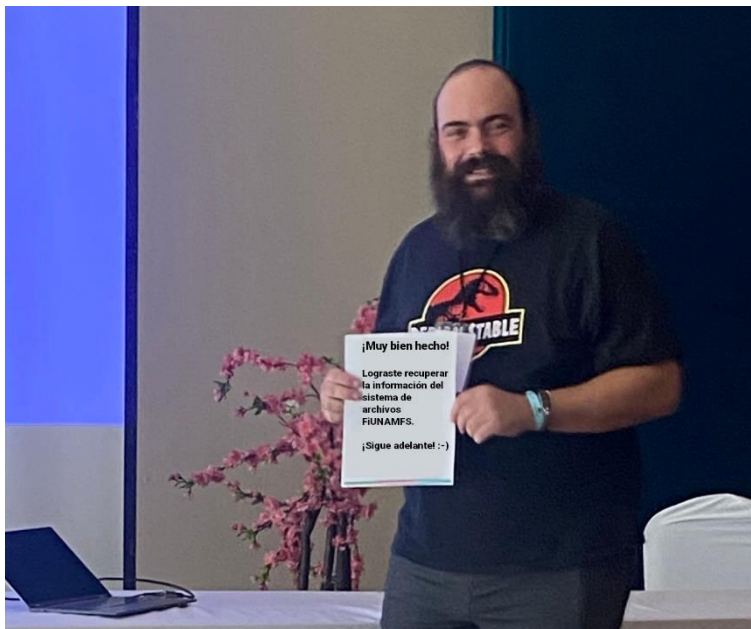
README.org

 mensaje.jpg	03/11/2024 07:57 p. m.	Archivo JPG	249 KB
 O1.txt	03/11/2024 03:40 p. m.	Documento de te...	1 KB
 README.org	03/11/2024 07:58 p. m.	Archivo ORG	31 KB
 tempCodeRunnerFile.py	03/11/2024 04:26 p. m.	Python File	1 KB

logo.png










mensaje.jpg



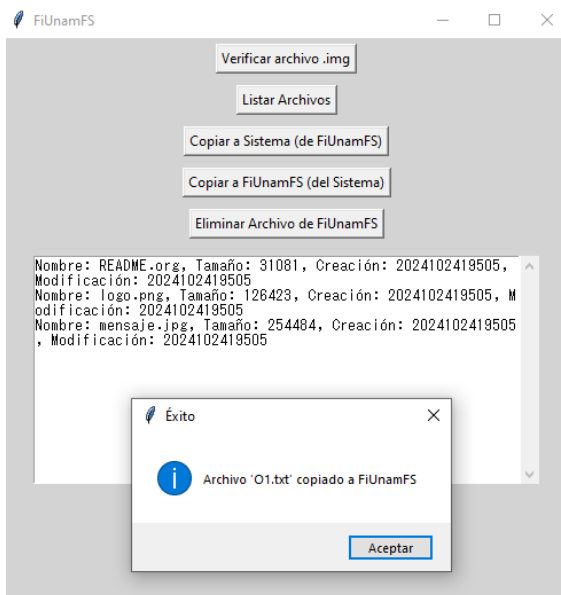
Gracias por el mensaje profesor :-)

### Ejemplo 3: Copiar un archivo desde el sistema local a FiUnamFS

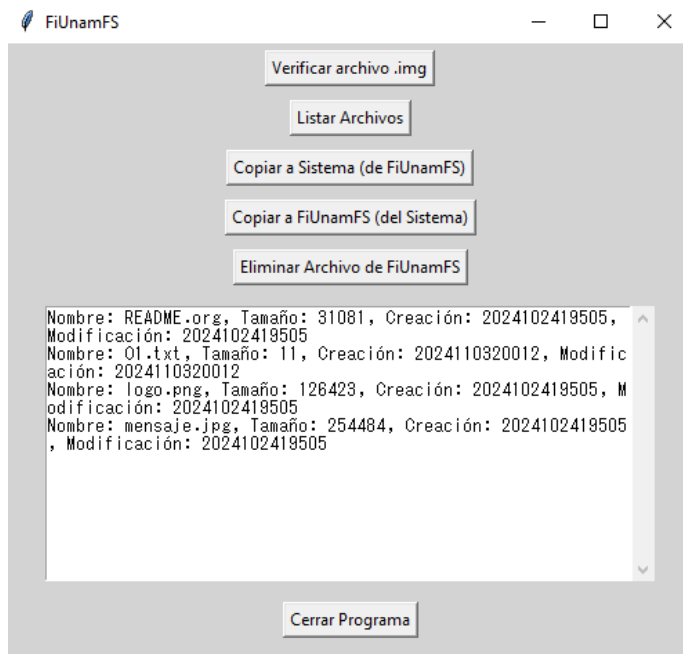
1. Selecciona "Copiar a FiUnamFS".
2. Selecciona el archivo que quieres copiar, en mi caso el documento de texto O1.txt

Nombre	Fecha de modificación	Tipo	Tamaño
 Fiptueba.py	03/11/2024 05:03 p. m.	Python File	7 KB
 fiunamfs.img	03/11/2024 05:04 p. m.	Archivo de image...	1,440 KB
 logo.png	03/11/2024 07:55 p. m.	Archivo PNG	124 KB
 mensaje.jpg	03/11/2024 07:57 p. m.	Archivo JPG	249 KB
 O1.txt	03/11/2024 03:40 p. m.	Documento de te...	1 KB
 README.org	03/11/2024 07:58 p. m.	Archivo ORG	31 KB
 tempCodeRunnerFile.py	03/11/2024 04:26 p. m.	Python File	1 KB

3. Aparecerá un mensaje diciendo que la acción tuvo éxito

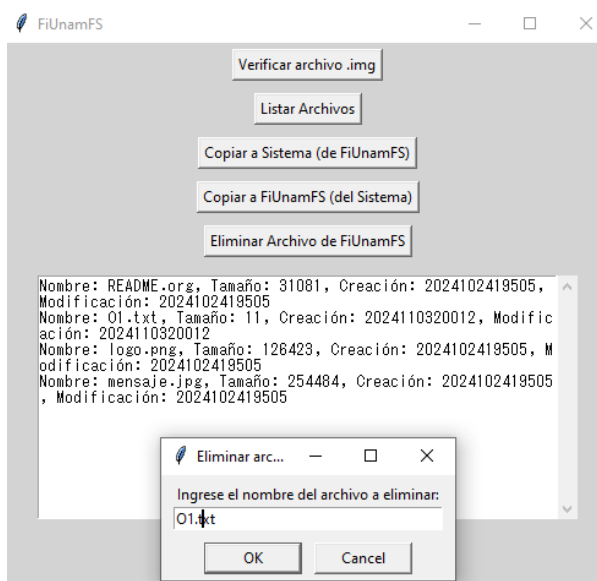


4. Presionamos el botón de “Listar Archivos” para comprobar que esta el archivo que copiamos, en este caso si aparece listado el archivo “O1.txt” con toda su información

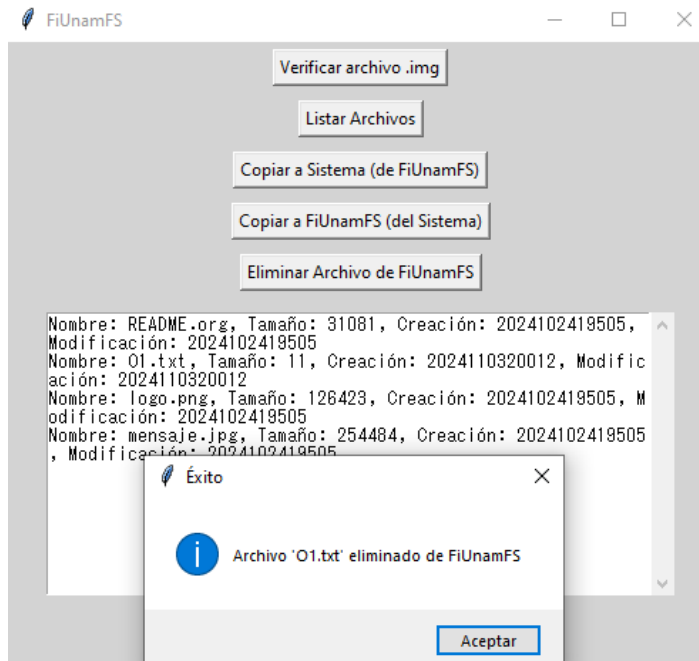


#### Ejemplo 4: Eliminar un archivo

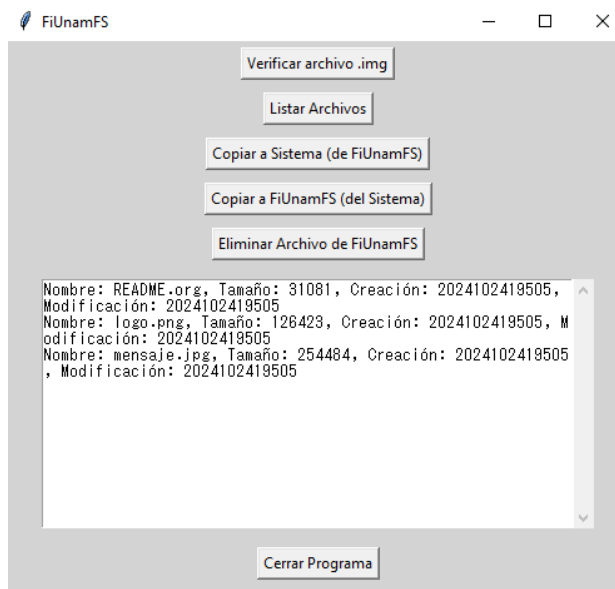
1. Selecciona "Eliminar Archivo de FiUnamFS".
2. Escribimos el nombre del archivo que queremos borrar de FiUnamFS



3. Si escribimos correctamente el nombre del archivo, saldrá una ventana diciendo que tuvimos éxito



4. Verificamos que el archivo haya sido borrado con éxito presionando otra vez el botón de "Listar archivos"



5. Al igual que todos los otros puntos, podemos cerrar el programa haciendo click en el botón "Cerrar programa"

## Limitaciones y Posibles Mejoras

- **Limitaciones:**

- El sistema de archivos no maneja subdirectorios.
- La longitud máxima de los nombres de archivo es de 15 caracteres.
- La sincronización entre hilos está limitada a operaciones sencillas y no cubre situaciones complejas de concurrencia.

- **Posibles Mejoras:**

- Implementar soporte para subdirectorios.
- Mejorar la interfaz gráfica con más detalles y opciones avanzadas de configuración.
- Añadir un módulo de registro para guardar las operaciones realizadas en un log.

## Conclusión:

Desarrollar el proyecto **FiUnamFS** fue una experiencia enriquecedora que nos permitió no solo profundizar en el manejo de sistemas de archivos y concurrencia, sino también enfrentar de manera directa los desafíos que surgen al implementar conceptos complejos en código funcional. Desde el inicio, fue evidente que lograr un sistema de archivos simulado con una interfaz gráfica amigable requeriría de múltiples iteraciones y ajustes.

Durante el proceso, hubo numerosos momentos en los que el código no funcionaba como esperábamos, desde problemas con la sincronización de hilos que provocaban fallas de concurrencia, hasta dificultades para interpretar correctamente los formatos binarios del sistema de archivos.

El desarrollo de la interfaz gráfica también presentó sus propios obstáculos. En varias ocasiones, implementamos componentes visuales que, aunque cumplían con las funcionalidades básicas, no ofrecían la claridad o el diseño deseado. La necesidad de realizar ajustes constantes en la interfaz y en los mensajes de error fue un recordatorio constante de la importancia de la experiencia del usuario en cualquier proyecto de software.

Al final, cada error y cada frustración nos enseñaron algo nuevo, llevándonos a refinar no solo el código, sino también nuestra capacidad de resolución de problemas y adaptación. **FiUnamFS** ahora no solo cumple con los requisitos planteados —listar, copiar y eliminar archivos con sincronización entre hilos—, sino que también se ha convertido en un proyecto que refleja mi esfuerzo, perseverancia y compromiso con la calidad del trabajo.

## Referencias

- Python Software Foundation. (2024). *tkinter — Python interface to Tcl/Tk*. Recuperado de <https://docs.python.org/es/3/library/tkinter.html>
- Adictos al Trabajo. (2020). *Interfaces gráficas en Python con Tkinter*. Recuperado de <https://adictosaltrabajo.com/2020/06/30/interfaces-graficas-en-python-con-tkinter/>
- PythonGuía. (2024). *Tkinter Tutoriales en Python*. Recuperado de <https://pythonguia.com/tkinter/>
- Iborra, N. (2024). *Desarrollo de interfaces con Tkinter - Programación en Python*. Recuperado de <https://nachoiborraies.github.io/python/13.html>
- Código Pitón. (n.d.). *Cómo Usar Hilos (Threads) en Python*. Retrieved from <https://www.codigopiton.com/como-usar-hilos-o-threads-en-python/>
- Ninoc, J. (n.d.). *Crear y administrar tareas concurrentes utilizando hilos en Python*. Retrieved from <https://www.jesusninoc.com/08/07/crear-y-administrar-tareas-concurrentes-utilizando-hilos-en-python/>