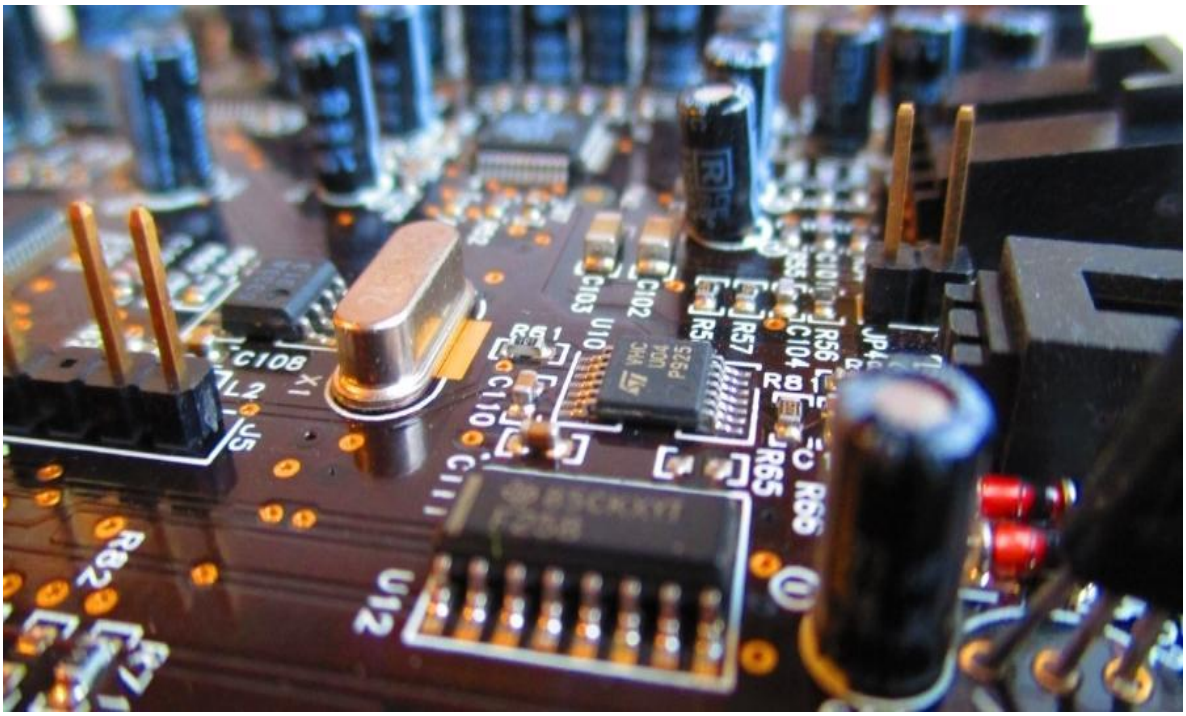
FACULTAD DE INGENIERÍA

CIUDAD UNIVERSITARIA



SISTEMAS OPERATIVOS

MAESTRO: GUNNAR EYAL WOLF ISZAEVICH

ALUMNO: YUKIOAYAX CANEK GABRIEL HERNÁNDEZ

BLOQUE: 136

GRUPO: 8

SEMESTRE 2026-1

Sistema de Archivos

Un sistema de archivos es el mecanismo que utiliza un sistema operativo para organizar, almacenar y recuperar datos en un dispositivo de almacenamiento de manera estructurada, sin este componente la información sería un conjunto de bytes sin sentido ni orden.
El objetivo de este proyecto es la implementación del gestor para el sistema de archivos FiUnamFS, para ello se trabajó manipulando directamente los bytes crudos de un archivo que simula ser un disquete de 1.44 MB (o siendo más específicos, 1.40625MB) interpretando su estructura interna (Superbloque y Directorio) sin depender de controladores del sistema operativo.
El principal reto fue la correcta lectura de datos binarios, ya que fue necesario realizar la conversión manual entre el formato Little Endian (del disco) y Big Endian (de Java), finalmente se desarrolló una Interfaz Gráfica de Usuario (GUI) que implementa hilos de ejecución (Threads) para gestionar las operaciones de lectura, escritura y eliminación de archivos de forma concurrente y eficiente.
Como se mencionó en la introducción, el desarrollo del proyecto se realizó en el lenguaje Java haciendo uso del JDK 21 (manteniendo compatibilidad con versiones anteriores, como Java 8), esta elección se debió principalmente a que tengo un mejor dominio de este lenguaje frente a otras opciones, lo cual facilitó el desarrollo, sumado a las ventajas que ofrece su API nativa para la manipulación de archivos binarios y la facilidad de que la Máquina Virtual de Java (JVM) permite ejecutar el programa en diferentes sistemas operativos.
Para la codificación, limpieza y compilación del código se utilizó Visual Studio Code, aprovechando sus herramientas de gestión de proyectos.
Es importante destacar que el proyecto se basa únicamente en la Biblioteca Estándar de Java sin la necesidad de instalar librerías externas o configuraciones complejas, se utilizaron los paquetes java.io para leer y escribir en el disco virtual, java.nio para solucionar el problema del formato Little Endian y javax.swing para crear la interfaz gráfica.
Para resolver el problema de manera ordenada y cumplir con los requerimientos de mantenibilidad y legibilidad, decidí no escribir todo el código en un solo archivo gigante, en su lugar dividí el proyecto en cuatro clases principales, cada una con una responsabilidad específica.

- **GestorDisco.java**: Esta es la clase más importante, se encarga de toda la lógica de bajo nivel, aquí utilizo la clase RandomAccessFile para abrir el archivo fiunamfs.img y mover el puntero de lectura/escritura (seek) a los bytes exactos que necesito, es la única clase que toca el disco directamente.
- **Entrada.java**: El directorio del sistema de archivos está compuesto por bloques de 64 bytes, leerlos a ojo es difícil, así que creé esta clase para que funcione como un molde, toma esos 64 bytes crudos y los convierte en un objeto fácil de usar con atributos como nombre, tamano y clusterInicial, también incluye una función vital para limpiar los nombres de archivo, eliminando los caracteres basura o espacios vacíos.

- **InterfazGrafica.java**: Para cumplir con el requerimiento de una interfaz de usuario amigable, desarrollé una ventana utilizando la biblioteca Swing, en lugar de pedir comandos de texto, el usuario tiene botones claros para "Listar", "Sacar", "Meter" y "Eliminar".
- **Main.java**: Es el punto de entrada del programa, su único trabajo es arrancar la interfaz gráfica de manera segura.

Durante el desarrollo me enfrenté a tres problemas principales que definieron la lógica del programa:

- **El problema de los números invertidos (Little Endian)**: Uno de los primeros obstáculos fue que Java lee los números enteros en formato Big Endian (el byte más significativo primero), pero la especificación de FiUnamFS usa Little Endian, al principio esto hacía que el tamaño del clúster o de los archivos me diera números gigantes o negativos, la solución fue implementar el uso de ByteBuffer con la orden ByteOrder.LITTLE_ENDIAN, esto voltea los bytes automáticamente antes de convertirlos a un entero (int), asegurando que los números sean correctos.
- **Escritura y Asignación de Espacio**: Para la función de importar archivos ("Meter"), tenía que decidir dónde guardar los datos sin borrar lo que ya existía, la solución fue implementar una lógica de asignación contigua, antes de guardar el programa recorre todo el directorio, calcula dónde termina el último archivo guardado (basándose en su clúster inicial y su tamaño) y coloca el nuevo archivo justo en el siguiente clúster libre, también se añadió una validación para ignorar los espacios que tienen ceros (archivos fantasma) y evitar corrupciones.
- **Manejo de Concurrencia (Hilos)**: La rúbrica pide el uso de hilos para la sincronización, si el programa intentara leer todo el disco en el mismo hilo que dibuja la ventana, la interfaz se congelaría y parecería que el programa se trabó, la solución fue aplicar un modelo de "Hilo Principal vs. Hilo de Trabajo", el hilo principal de Java (EDT) se encarga solo de detectar los clics en los botones, cuando el usuario pide algo pesado (como listar o copiar), esa tarea se lanza en un nuevo Hilo (Thread) independiente, de esta forma, la interfaz sigue respondiendo mientras el disco trabaja en segundo plano.

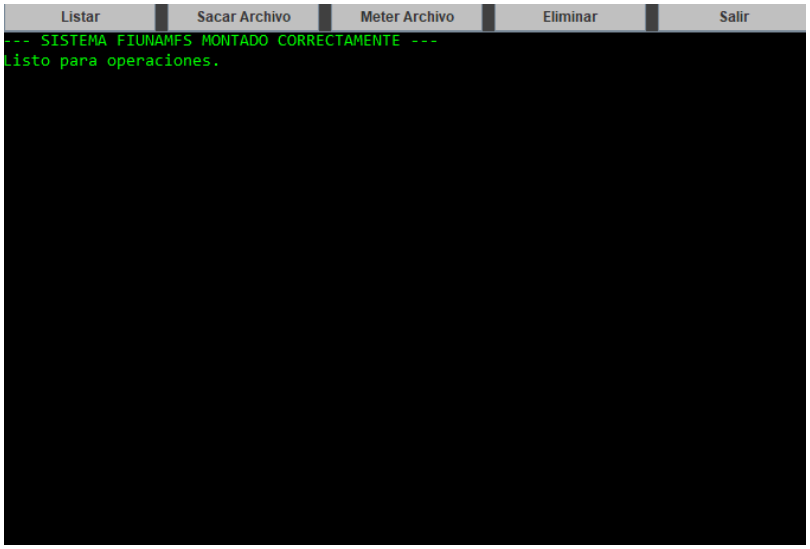
El desarrollo fue paso a paso y usé la terminal para gestionar el control de versiones, avancé por partes, primero la lectura del disco, luego la escritura y al final la interfaz gráfica.

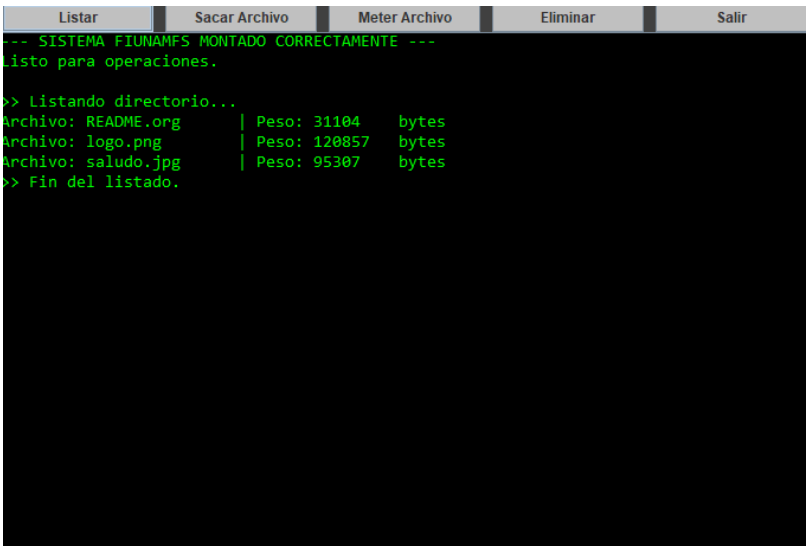
Con el sistema ya funcionando, agregué el archivo .gitignore para evitar subir los binarios compilados, manteniendo el repositorio limpio.

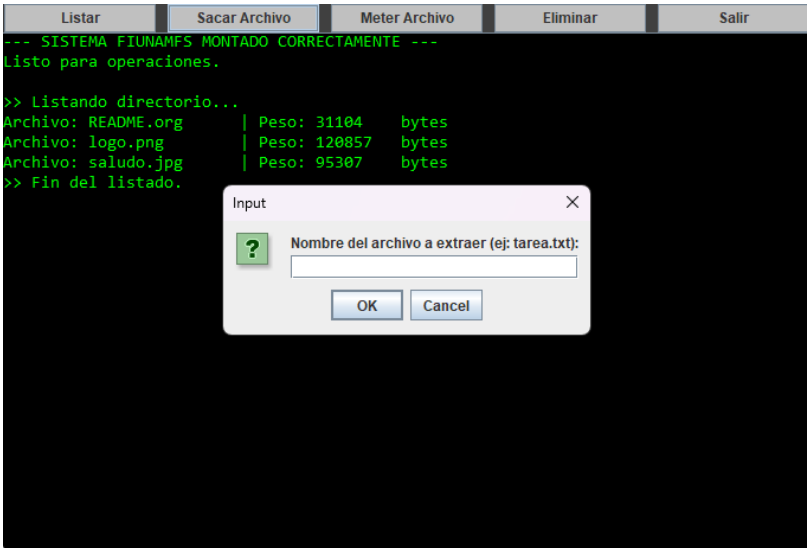
Para la ejecución del proyecto se debe abrir una terminal situándose directamente en la carpeta que contiene los archivos y la imagen del disco, acto seguido es necesario compilar la totalidad de las clases ejecutando la instrucción javac *.java para generar los binarios correspondientes y finalmente lanzar la aplicación mediante el comando java Main lo cual abrirá la interfaz gráfica permitiendo la manipulación inmediata del sistema de archivos.

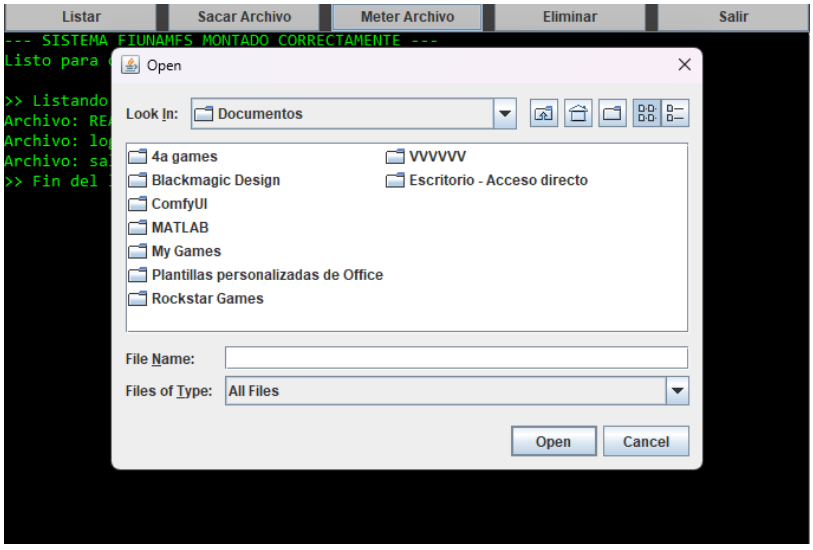
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\GAB> cd 'D:\Escritorio\Proyecto 2\'
PS D:\Escritorio\Proyecto 2> ls


    Directorio: D:\Escritorio\Proyecto 2


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----      20/11/2025  09:29 a. m.           1749 Entrada.java
-a----      20/11/2025  08:10 p. m.        1474560 fiunamfs.img
-a----      20/11/2025  09:37 a. m.           6742 GestorDisco.java
-a----      20/11/2025  09:40 p. m.             10 holamundo.txt
-a----      20/11/2025  08:21 a. m.           6438 InterfazGrafica.java
-a----      20/11/2025  08:17 a. m.            555 Main.java


PS D:\Escritorio\Proyecto 2> javac *.java
PS D:\Escritorio\Proyecto 2> java Main
```

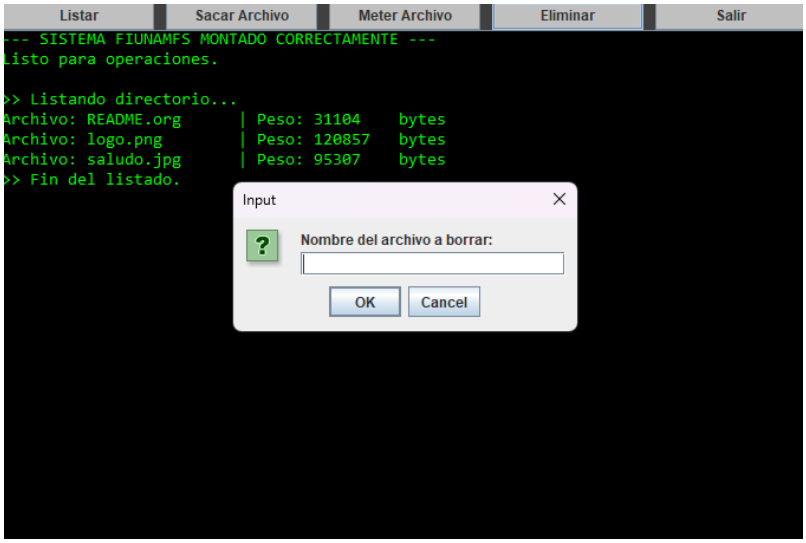
Una vez realizado lo anterior aparecerá una pantalla como la siguiente en la cual se podrán realizar las acciones solicitadas.






Listar
Sacar Archivo
Meter Archivo
Eliminar
Salir
--- SISTEMA FIUNAMFS MONTADO CORRECTAMENTE ---
Listo para operaciones.
>> Listando directorio...
Archivo: README.org | Peso: 31104 bytes
Archivo: logo.png | Peso: 120857 bytes
Archivo: saludo.jpg | Peso: 95307 bytes
>> Fin del listado.
Input
Nombre del archivo a extraer (ej: tarea.txt):
OK
Cancel


Listar
Sacar Archivo
Meter Archivo
Eliminar
Salir
--- SISTEMA FIUNAMFS MONTADO CORRECTAMENTE ---
Open
Look In:
Documentos
4a games
Blackmagic Design
ComfyUI
MATLAB
My Games
Plantillas personalizadas de Office
Rockstar Games
VVVVVV
Escritorio - Acceso directo
File Name:
Files of Type:
All Files
Open
Cancel


Listar
Sacar Archivo
Meter Archivo
Eliminar
Salir
--- SISTEMA FIUNAMFS MONTADO CORRECTAMENTE ---
Listo para operaciones.
>> Listando directorio...
Archivo: README.org | Peso: 31104 bytes
Archivo: logo.png | Peso: 120857 bytes
Archivo: saludo.jpg | Peso: 95307 bytes
>> Fin del listado.
Input
Nombre del archivo a borrar:
OK
Cancel

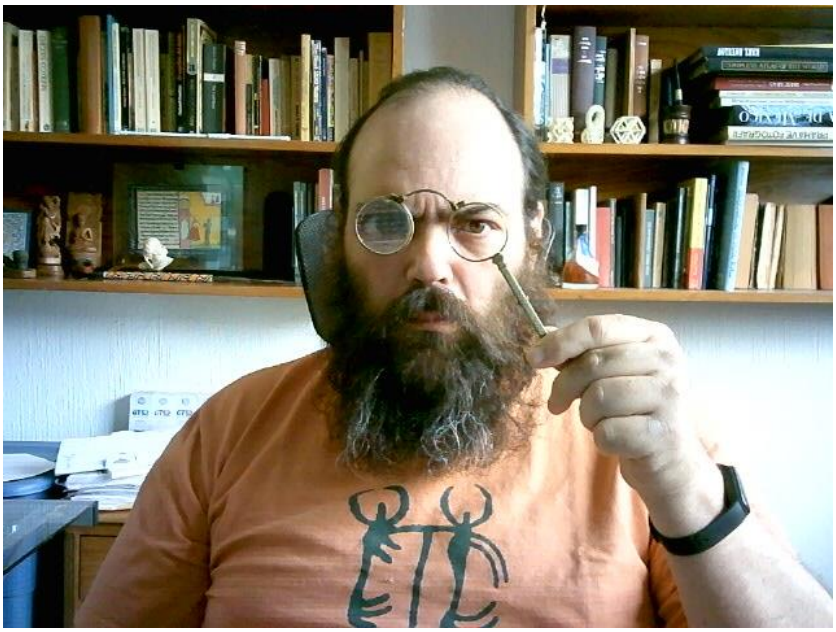
Para extraer o eliminar un archivo del sistema FiUnamFS hacia la computadora es necesario ingresar el nombre completo del archivo incluyendo su extensión (es decir, el sufijo que indica el tipo de archivo como .txt, .png o .c), el sistema requiere la coincidencia exacta del nombre para localizar la entrada en el directorio y proceder con la copia de los datos o la eliminación.
Tras realizar la lectura del directorio, es posible observar los siguientes archivos almacenados en el disco:

README.org

```
#+title: Proyecto: (Micro) sistema de archivos multihilos

#+BEGIN_SRC yaml
Planteamiento: 2025.10.30
Entrega: 2025.11.20
#+END_SRC

# ¡Las [[./calificaciones.org][calificaciones y comentarios]] ya están disponibles!

** Descripción del proyecto

Trabajaremos este proyecto conjunto para las últimas dos unidades:

- Sistemas de archivos :: Resulta natural que el proyecto sea implementar un
  sistema de archivos 😍 Para esto, lo harán trabajando sobre una
  /especificación/ y sobre un /caso de referencia/.
- Administración de procesos :: Para esta unidad toca dividir la lógica de un
  proceso en sus componentes /concurrentes/, buscando que se comuniquen los
  cambios de estado empleando /mecanismos de sincronización/.

** ¿Qué tengo que hacer?

Lo que ustedes deben desarrollar es un programa que pueda obtener, crear y
modificar información en el micro-sistema-de-archivos que desarrollé para la
Facultad de Ingeniería, =FiUnamFS=.

Siguiendo la especificación que aparece en la siguiente sección, tienen que
desarrollar un programa que pueda:

1. Listar los contenidos del directorio
2. Copiar uno de los archivos de dentro del =FiUnamFS= hacia tu sistema
3. Copiar un archivo de tu computadora hacia tu =FiUnamFS=
4. Eliminar un archivo del =FiUnamFS=
5. El programa que desarrollen debe contar, por lo menos, dos hilos de
   ejecución, operando /concurrentemente/, y que se /comuniquen su estado/
   mediante mecanismos de sincronización.

*** Sistema muestra

Para verificar su implementación, pueden [[./fiunamfs.img][descargar un sistema de archivos
ejemplo]] que cumple con el planteamiento. Verifiquen que pueden realizar
todas las tareas que les solicité con éste. Indíquenme — ¿Cuáles son los
contenidos del /«disco»/?

** Especificación de =FiUnamFS=

- El sistema de archivos cabe en un /diskette/ tradicional. Claro, no espero que
  tengan acceso al hardware, por lo que lo simularemos representándolo en un
  archivo de longitud fija, de 1440 Kilobytes
- Por simplicidad, en todas las estructuras de FIUnamFS, las cadenas de texto
  deben ser ASCII 8-bit (/no/ requerimos que sean Unicode UTF-8).
```

logo.png

saludo.jpg



La realización de este proyecto fue importante para comprender cómo opera internamente un sistema de archivos, trabajar directamente con los bytes del disco y resolver el problema del formato Little Endian fue el mayor reto, pero permitió entender cómo se estructuran los datos sin depender de herramientas ya hechas.
Se cumplieron todos los objetivos, el programa es funcional, permite leer, guardar y eliminar archivos y gestiona el espacio disponible correctamente, además, la implementación de hilos (threads) para la interfaz gráfica fue indispensable para que la aplicación responda rápido y no se congele durante las operaciones de lectura o escritura.
Personalmente, me quedo con la experiencia de haber organizado el código en módulos claros, separando la lógica del disco de la parte visual, esto no solo facilitó el desarrollo, sino que demostró que una buena estructura es esencial para que un proyecto de este tipo funcione bien.

Bibliografía

Astorgano, M. C. & Monés, A. M. (2023). Introducción al diseño de interfaces gráficas de usuario con JSwing. Repositorio Documental Universidad de Valladolid. https://uvadoc.uva.es/bitstream/handle/10324/59910/Guiones_IPC_versi%C3%B3n_final.pdf

Tanenbaum, A. S. (2003). *Sistemas operativos modernos (2.ª ed.).* Pearson Educación. https://books.google.com.mx/books?id=g88A4rxPH3wC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Eck, D. J. (mayo, 2022). *Introduction to Programming Using Java (9.ª ed.).* Hobart and William Smith Colleges. https://math.hws.edu/eck/cs124/downloads/javanotes9-linked.pdf

S.A. (s.f.). *JDK 25 Documentation*. ORACLE. Recuperado el 20 de octubre de 2025 de https://docs.oracle.com/en/java/javase/25/