MANUAL DE USUARIO — SIMULACIÓN DEL SISTEMA DE ARCHIVOS FAT

Lenguaje: Python
Fecha de entrega: 15 de octubre de 2025
Autor: André Velasco - Carne: 1546124
Curso: Manejo e Implementación de Archivos

Índex

Contenido

Introducción

El presente manual de usuario tiene como objetivo guiar paso a paso el uso del programa “Sistema FAT Simulado”, desarrollado en el lenguaje de Python. Su objetivo principal es demostrar el funcionamiento lógico de una estructura FAT sin depender de un entorno de sistema real.

Este sistema permite gestionar archivos mediante una estructura inspirada en el sistema de archivos FAT también conocido como File Allocation Table, simulando las operaciones básicas que realiza un sistema operativo para controlar la creación de archivos, modificación, eliminación y recuperación de archivos.

El programa ha sido diseñado para ejecutar operaciones desde consola y replicar la lógica de asignación de bloques, almacenamiento en archivos JSON, y manejo de permisos de usuarios.

Requisitos del sistema

Antes de ejecutar el programa, asegúrese de cumplir con los siguientes requisitos:

Requisitos mínimos

- Sistema operativo: Windows 10 / 11, macOS o Linux
- Lenguaje: Python 3.10 o superior / programa de Python implementado en Visual Studio Code
- Librerías necesarias:
    - os → Manejo de archivos y directorios
    - json → Serialización de datos en formato JSON
    - datetime → Registro de fechas y horas de los eventos

Aunque con las librerías ya vendrán implementadas en caso de usar Python en Visual

Archivos generados por el sistema

Durante la ejecución, el programa crea y utiliza las siguientes carpetas:

- fat_storage/

Contiene el archivo principal fat.json (Tabla FAT)

- blocks/

Contiene los bloques de datos segmentados en archivos JSON.

Estructura del sistema FAT

El sistema simula el funcionamiento de la tabla FAT, que sirve como índice de control y seguimiento de archivos.

Cada archivo creado por el usuario se compone de dos partes principales:

1. Tabla FAT (fat.json)

Contiene los metadatos del archivo.

2. Bloques físicos (blocks/" nombre_archivo" .json)

Contienen los datos reales del archivo divididos en fragmentos de máximo 20 caracteres.

Ejemplo de estructura FAT (fat.json):


```
{
  "files": {
    "documento": {
      "ruta_inicial": "blocks/documento_block0.json",
      "papelera": false,
      "size": 45,
      "created": "2025-10-09 20:13",
      "modified": null,
      "deleted": null,
```

```
    "owner": "admin",
    "permisos": {
      "admin": ["lectura", "escritura"]
    }
   }
  }
}
```


Ejemplo del bloque generado por fat.json:


```
{
  "datos": "Hola esto es un ejem",
  "siguiente": "blocks/documento_block1.json",
  "eof": false
}
```


Cada bloque guarda 20 caracteres del contenido, cuando alcanza el límite pasaría al siguiente bloque, y un valor booleano “eof” que indica el final del archivo.

Funcionamiento general

Al ejecutar el programa, se crea automáticamente la estructura básica del sistema de las carpetas y archivos. Luego se presenta un menú principal en consola con distintas operaciones disponibles.

--- Menú ---

1. Crear archivo

2. Listar archivos

3. Mostrar papelera

4. Abrir archivo

5. Modificar archivo

6. Eliminar (mover a papelera)

7. Recuperar desde papelera

8. Asignar permisos (owner/creador)

9. Salir

Descripción de cada funcionalidad

- Crear archivo

Permite crear un nuevo archivo con contenido definido por el usuario.

Durante este proceso, el sistema divide el texto ingresado en fragmentos de 20 caracteres y los guarda en la carpeta blocks/.

Pasos:

1. El usuario ingresa el nombre del archivo.
2. Escribe el contenido línea por línea.
3. Finaliza con la palabra clave FIN.
4. El sistema genera los bloques y los registra en fat.json.

Ejemplo:

Nombre del archivo: notas

Contenido (FIN para terminar):

Proyecto FAT en Python

Listo para entrega

FIN

Archivo 'notas' creado.

- Listar archivos

Muestra todos los archivos activos (no eliminados) con su tamaño y propietario.

Ejemplo de salida:

=== Archivos activos ===

Prueba | Tamaño: 38 | Owner: admin

documento | Tamaño: 25 | Owner: admin

Esta lista excluye los archivos que han sido movidos a la papelera.

- Mostrar papelera

Muestra los archivos que se encuentran en estado de eliminación (papelera = True), junto con su fecha de eliminación.

Ejemplo:

=== Papelera ===

Documento | Eliminado: 2025-10-10 14:22

- Abrir archivo

Permite visualizar el contenido completo del archivo, uniendo los bloques de 20 caracteres.

Validaciones:

- Solo puede acceder el usuario con permiso de lectura.
- Si el archivo está en la papelera o no existe, el sistema mostrará un mensaje de error.

Ejemplo:

Archivo a abrir: notas

Contenido del archivo:

Proyecto FAT falta registro de otros usuarios.

- Modificar archivo

Actualiza el contenido de un archivo existente.

Funcionamiento interno:

1. Verifica permisos de escritura.
2. Elimina físicamente los bloques anteriores.
3. Crea nuevos bloques con el texto actualizado.
4. Actualiza la fecha de modificación en la FAT.

Ejemplo:

Archivo a modificar: notas

Nuevo contenido (FIN para terminar):

Proyecto FAT actualizado para entrega final

FIN

Archivo modificado.

- Eliminar archivo (mover a papelera)

No borra físicamente los bloques, sino que actualiza el valor de la papelera a “True” en el FAT y agrega la fecha de eliminación.

Ejemplo:

Archivo a eliminar: notas

Archivo movido a la papelera.

- Recuperar desde papelera

Permite restaurar un archivo previamente eliminado, cambiando papelera a False y limpiando la fecha de eliminación.

Ejemplo:

Archivo a recuperar: notas

Archivo recuperado.

- Asignar permisos

Solo el usuario propietario (owner) del archivo puede asignar permisos a otros usuarios. Los permisos disponibles son:

- lectura
- escritura

Ejemplo:

Archivo: notas

Usuario a asignar: juan

Permiso (lectura/escritura): lectura

Permiso 'lectura' asignado a 'juan'.

- Salir

Finaliza el programa limpiamente y guarda todos los cambios realizados en las respectivas carpetas y en loa archivos que se han generado.

Detalles a conocer sobre el sistema

Lógica de funcionamiento FAT

El sistema se basa en el modelo lógico del sistema de archivos FAT.

Cada archivo tiene una entrada en la tabla FAT que actúa como índice, apuntando al primer bloque del archivo.

Cada bloque, a su vez, apunta al siguiente mediante el campo siguiente, simulando la asignación encadenada de bloques.

Gestión de permisos

La FAT también incluye un sistema de control de acceso:

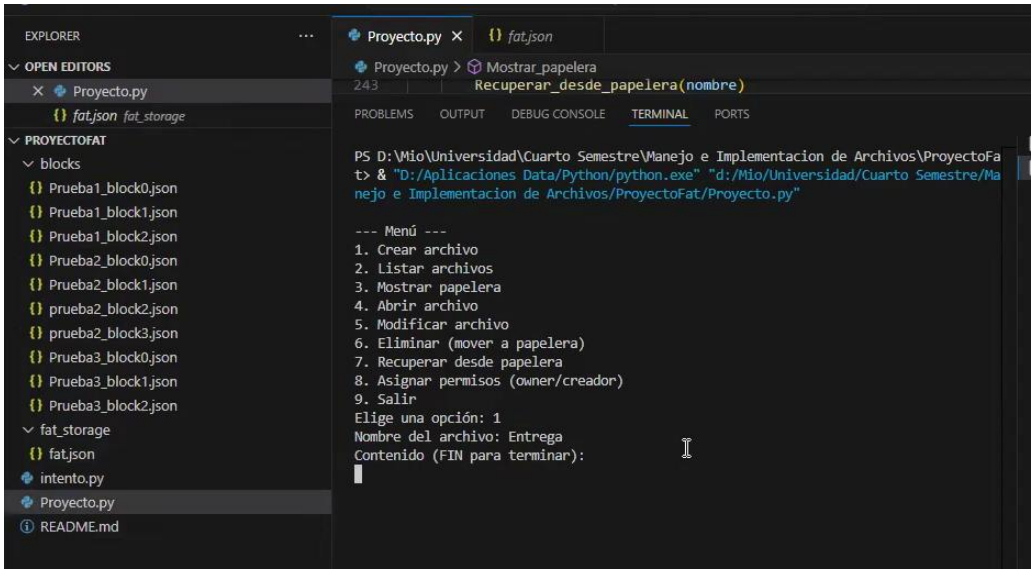
- El usuario propietario (owner) tiene todos los permisos por defecto.
- Puede otorgar permisos de lectura o escritura a otros usuarios.
- Si un usuario no tiene permisos, no podrá abrir ni modificar archivos.

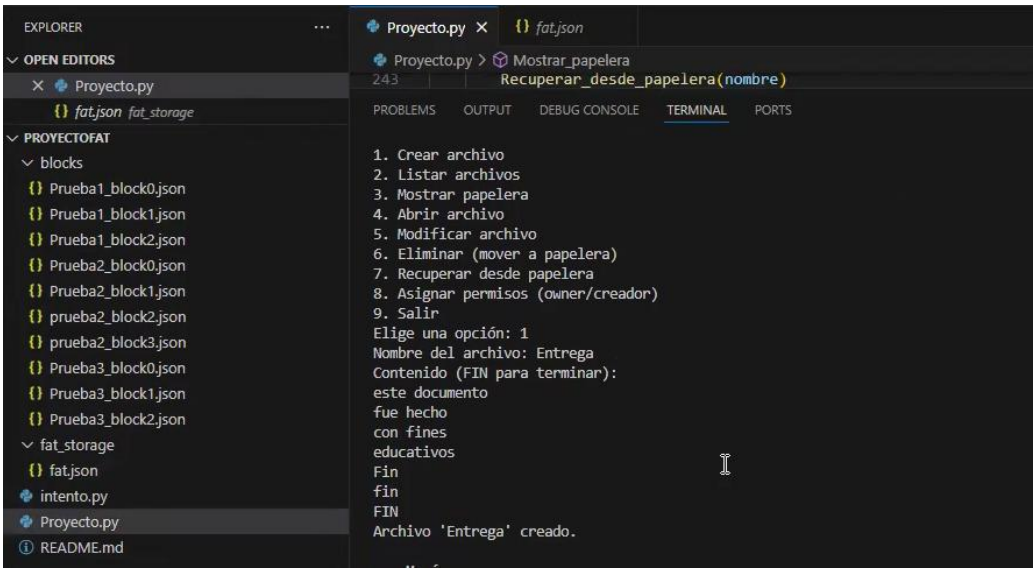
Esto replica la lógica de seguridad básica presente en sistemas operativos reales.

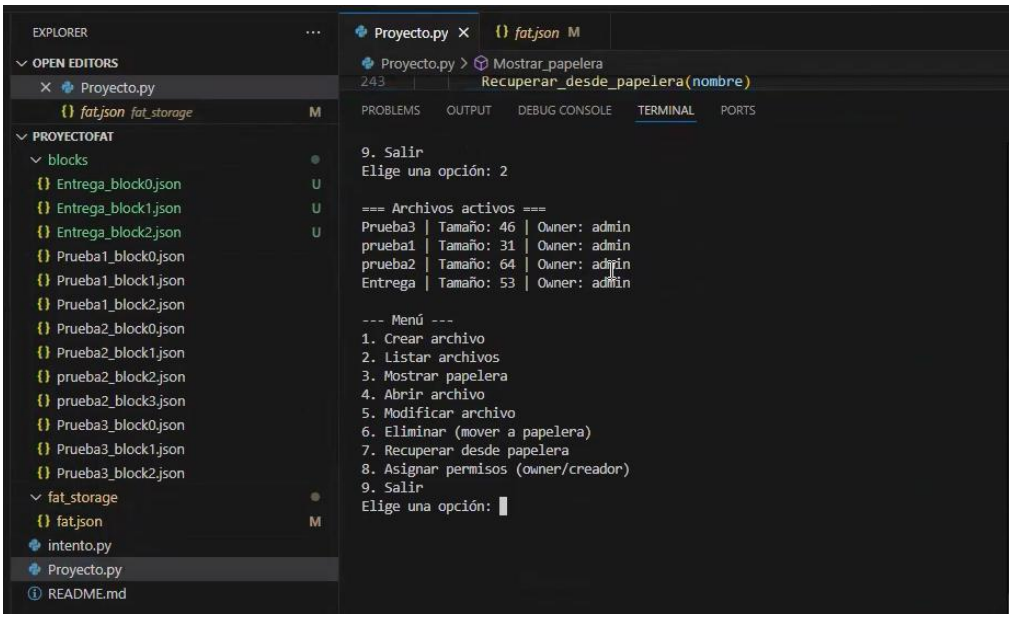
Buenas prácticas y recomendaciones

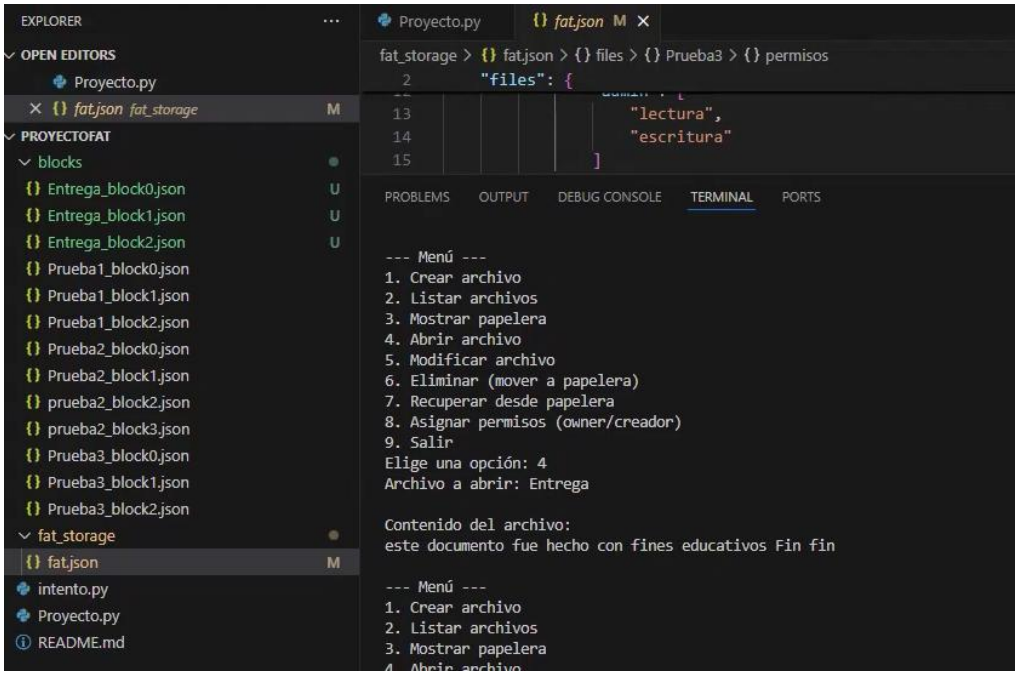
- Evite usar nombres de archivo con espacios o caracteres especiales.
- No elimine manualmente los archivos JSON en las carpetas fat_storage/ o blocks/.
- Respete el comando FIN al ingresar texto, ya que marca el final del contenido.
- Ejecute el programa siempre desde la carpeta principal.
- Puede editar el usuario actual cambiando la variable Usuario = "admin" para realizar pruebas.

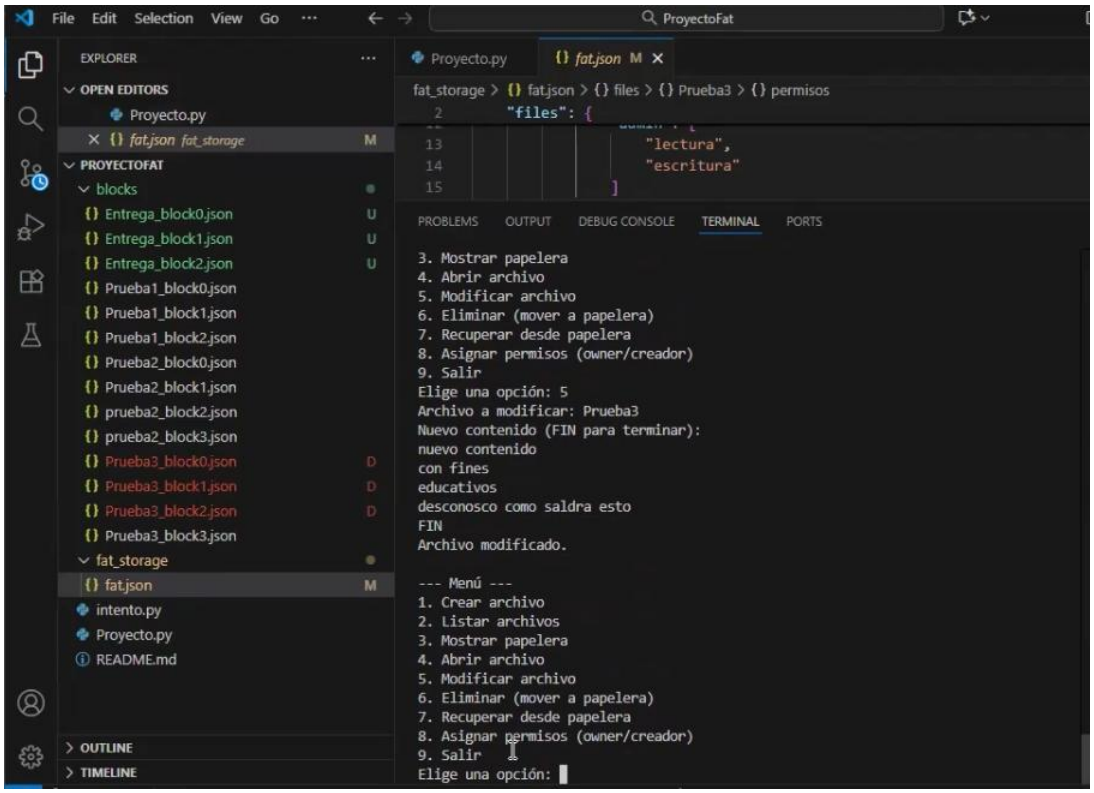
Ejemplo completo de ejecución






EXPLORER
OPEN EDITORS
Proyecto.py
fat.json fat_storage M
PROYECTOFAT
blocks
Entrega_block0.json U
Entrega_block1.json U
Entrega_block2.json U
Prueba1_block0.json
Prueba1_block1.json
Prueba1_block2.json
Prueba2_block0.json
Prueba2_block1.json
prueba2_block2.json
prueba2_block3.json
Prueba3_block0.json
Prueba3_block1.json
Prueba3_block2.json
fat_storage
fat.json M
intento.py
Proyecto.py
README.md
Proyecto.py
fat.json M
Proyecto.py > Mostrar_papelera
243 Recuperar_desde_papelera(nombre)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
9. Salir
Elige una opción: 2
=== Archivos activos ===
Prueba3 | Tamaño: 46 | Owner: admin
prueba1 | Tamaño: 31 | Owner: admin
prueba2 | Tamaño: 64 | Owner: admin
Entrega | Tamaño: 53 | Owner: admin
--- Menú ---
1. Crear archivo
2. Listar archivos
3. Mostrar papelera
4. Abrir archivo
5. Modificar archivo
6. Eliminar (mover a papelera)
7. Recuperar desde papelera
8. Asignar permisos (owner/creador)
9. Salir
Elige una opción:


EXPLORER
OPEN EDITORS
Proyecto.py
fat.json fat_storage M
PROYECTOFAT
blocks
Entrega_block0.json U
Entrega_block1.json U
Entrega_block2.json U
Prueba1_block0.json
Prueba1_block1.json
Prueba1_block2.json
Prueba2_block0.json
Prueba2_block1.json
prueba2_block2.json
prueba2_block3.json
Prueba3_block0.json
Prueba3_block1.json
Prueba3_block2.json
fat_storage
fat.json M
intento.py
Proyecto.py
README.md
Proyecto.py
fat.json M
fat_storage > fat.json > files > Prueba3 > permisos
2 "files": {
13 "lectura",
14 "escritura"
15]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
--- Menú ---
1. Crear archivo
2. Listar archivos
3. Mostrar papelera
4. Abrir archivo
5. Modificar archivo
6. Eliminar (mover a papelera)
7. Recuperar desde papelera
8. Asignar permisos (owner/creador)
9. Salir
Elige una opción: 4
Archivo a abrir: Entrega
Contenido del archivo:
este documento fue hecho con fines educativos Fin fin
--- Menú ---
1. Crear archivo
2. Listar archivos
3. Mostrar papelera


File Edit Selection View Go ···
ProyectoFat
EXPLORER
OPEN EDITORS
Proyecto.py
fat.json fat_storage M
PROYECTOFAT
blocks
Entrega_block0.json U
Entrega_block1.json U
Entrega_block2.json U
Prueba1_block0.json
Prueba1_block1.json
Prueba1_block2.json
Prueba2_block0.json
Prueba2_block1.json
prueba2_block2.json
prueba2_block3.json
Prueba3_block0.json D
Prueba3_block1.json D
Prueba3_block2.json D
Prueba3_block3.json
fat_storage
fat.json M
intento.py
Proyecto.py
README.md
OUTLINE
TIMELINE
Proyecto.py
fat.json M
fat_storage > fat.json > files > Prueba3 > permisos
"files": {
"lectura",
"escritura"
]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
3. Mostrar papelera
4. Abrir archivo
5. Modificar archivo
6. Eliminar (mover a papelera)
7. Recuperar desde papelera
8. Asignar permisos (owner/creador)
9. Salir
Elige una opción: 5
Archivo a modificar: Prueba3
Nuevo contenido (FIN para terminar):
nuevo contenido
con fines
educativos
desconosco como saldra esto
FIN
Archivo modificado.
--- Menú ---
1. Crear archivo
2. Listar archivos
3. Mostrar papelera
4. Abrir archivo
5. Modificar archivo
6. Eliminar (mover a papelera)
7. Recuperar desde papelera
8. Asignar permisos (owner/creador)
9. Salir
Elige una opción:

Conclusión

Este proyecto implementa con éxito una simulación lógica del sistema de archivos FAT, permitiendo comprender cómo se gestionan los archivos, sus bloques de datos, y los permisos de usuario en un entorno controlado.

El programa permite al estudiante visualizar de forma práctica los conceptos de asignación encadenada, control de metadatos y administración de accesos.

Con esta herramienta, se refuerza la comprensión del funcionamiento interno de los sistemas de archivos y su relación con los mecanismos de almacenamiento y seguridad de los sistemas operativos modernos.

Link a un video mostrando y explicando el sistema.

https://drive.google.com/file/d/1uwaKkuOhuNUmwaXL_EeAv3H79UCNVKe8/view?usp=sharing

Link al repositorio público de GitHub con el código fuente.

KarotSystems/ProyectoFat