UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LABORATORIO MANEJO E IMPLEMENTACION DE ARCHIVOS, Sección "-A"
2do. Semestre de 2025
Catedrático: Ing. Juan Álvaro Díaz Ardavin
Tutor Académico: Edgar Josías Cán Ajquejay

Manual Técnico
"GoDisk"

DESARROLADO POR: Jose David Mota González

CARNET: 202306077

Tabla de contenido

I. Introducción

A. Propósito del documento

El presente manual técnico tiene como propósito describir de forma detallada el diseño, implementación y funcionamiento del sistema **GoDisk**, desarrollado como parte del curso Manejo e Implementación de Archivos. Este documento está orientado a proporcionar una guía clara para comprender la arquitectura del sistema, las estructuras de datos utilizadas, los comandos implementados y el flujo de operación del sistema de archivos EXT2 simulado.

El manual servirá como material de referencia tanto para el mantenimiento y mejora del sistema, como para comprender su diseño a nivel académico y profesional.

B. Descripción general del sistema

GoDisk es una aplicación web que simula el comportamiento de un sistema de archivos **EXT2**. Su diseño integra:

1. *Backend en Go (Golang):* encargado de manejar la lógica principal, simular discos mediante archivos binarios `[.mia]`, administrar particiones y gestionar estructuras internas como MBR, inodos, bloques y bitmaps.
2. *Frontend Web:* desarrollado con un framework moderno (React + TypeScript), que permite a los usuarios interactuar mediante un área de entrada y salida de comandos, además de cargar scripts **`[.smia]`**.
3. *Reportes con Graphviz:* que facilitan la visualización de estructuras internas del sistema, como tablas de inodos, bloques, árboles de directorios y mapas de particiones.

De esta manera, el sistema ofrece una plataforma práctica e interactiva para ejecutar operaciones típicas de administración de discos, usuarios, archivos y permisos, sin depender de hardware físico.

C. Problema que resuelve

Uno de los principales retos en el aprendizaje de sistemas de archivos es comprender cómo interactúan las estructuras internas (discos, particiones, inodos, bloques, etc.) con los comandos que el usuario ejecuta.

En entornos reales, experimentar con sistemas de archivos implica riesgos como la pérdida de información o la necesidad de hardware dedicado.

GoDisk resuelve este problema al:

- Proporcionar un entorno seguro y controlado para experimentar con el sistema de archivos EXT2.
- Permitir la creación, montaje y formateo de discos virtuales sin afectar el sistema operativo anfitrión.
- Facilitar la visualización gráfica de las estructuras internas, fortaleciendo la comprensión teórica mediante la práctica.

II. Marco Teórico

El sistema de archivos **Ext2 (Second Extended File System)** es un sistema de archivos para el kernel de Linux, diseñado por Rémy Card y lanzado en enero de 1993 como una mejora significativa sobre el sistema de archivos original MINIX y el Extended File System (Ext). Fue el primer sistema de archivos de grado comercial para Linux y rápidamente se convirtió en el predeterminado para muchas distribuciones.

1. Origen y Propósito

- **Desarrollo**: Creado por Rémy Card como parte de un proyecto de tesis, fue diseñado para superar las limitaciones de características y eficiencia de los sistemas de archivos anteriores, como el tamaño máximo de 2GB para el Ext original y las limitaciones de marcas de tiempo.
- **Filosofía**: Se diseñó siguiendo los principios del Berkeley Fast File System de BSD y con la extensibilidad en mente, dejando espacio en sus estructuras de datos en disco para futuras versiones.
- **Popularidad**: Fue el sistema de archivos predeterminado en distribuciones de Linux como Debian y Red Hat Linux antes de ser reemplazado por Ext3.

2. Estructura y Organización del Disco El sistema de archivos Ext2 organiza los datos en el disco de una manera específica para optimizar el rendimiento y la recuperabilidad:

- **Particiones**: Cada sistema de archivos Ext2 ocupa una partición completa.
- **Grupos de Bloques (Block Groups)**: Una partición Ext2 se divide en "grupos de bloques", que son secciones lógicas del disco que agrupan datos relacionados. Esta división se realiza durante el formateo del sistema de archivos.
    - Cada grupo de bloques contiene un **Superblock**, un **descriptor de grupo**, un **mapa de bits de bloques**, un **mapa de bits de inodos**, una **tabla de inodos** y **bloques de datos**.
- **Superblock**: Contiene información vital sobre la configuración y el estado general del sistema de archivos, como el número total de bloques e inodos, cuántos están

libres, el tamaño de los bloques, el tiempo del último chequeo, y el estado de montaje. Para asegurar la recuperabilidad, se guardan copias redundantes del Superblock en cada grupo de bloques.

- **Descriptores de Grupo (Group Descriptors)**: Almacenan información sobre cada grupo de bloques, incluyendo punteros a las tablas de inodos y a los mapas de bits de asignación.
- **Mapas de Bits (Block and Inode Bitmaps)**: Son listas de bits que describen qué bloques y qué inodos están en uso o disponibles para su asignación futura dentro de cada grupo de bloques.
- **Inodos (Inodes)**: Cada archivo o directorio está asociado con exactamente un inodo. Los inodos almacenan metadatos importantes sobre el archivo, incluyendo:
    - Fechas de creación, modificación y último acceso (ctime, mtime, atime).
    - Permisos, propietario y grupo.
    - Tipo de archivo (regular, directorio, dispositivo, enlace simbólico, etc.).
    - Punteros a la ubicación de los bloques de datos en el disco. Ext2 utiliza punteros directos, indirectos, doblemente indirectos y triplemente indirectos para soportar archivos de gran tamaño con un acceso eficiente.
    - **No almacenan el nombre del archivo en sí**; los nombres se guardan en las entradas de directorio.
- **Directorios**: Son archivos especiales que asocian nombres textuales de archivos con sus números de inodo correspondientes. El directorio raíz siempre se almacena en el

inodo número dos. Las implementaciones más nuevas de Ext2 también pueden usar **directorios indexados con hashes** (H-tree) para mejorar el rendimiento en directorios grandes.

- **Asignación de Datos**: Ext2 intenta asignar nuevos directorios en el mismo grupo de bloques que su directorio padre y los archivos en el mismo grupo que sus entradas de directorio para minimizar los tiempos de búsqueda.

3. Características Principales

- **Simplicidad y Estabilidad**: Se le atribuye por su sencillez, estabilidad y buen rendimiento, lo que le permitió manejar varias tareas de manera admirable.
- **Sin Journaling**: A diferencia de sus sucesores Ext3 y Ext4, **Ext2 no incorpora un sistema de Journaling**. Esto significa que, tras apagados inesperados o fallos del sistema, el tiempo de recuperación puede ser más largo y puede haber una mayor probabilidad de pérdida de datos.
- **Rendimiento**: Es conocido por ser un sistema de archivos rápido. Su falta de Journaling lo hace ideal para medios de almacenamiento basados en flash (como tarjetas SD y unidades USB) donde minimiza el número de escrituras, prolongando la vida útil del dispositivo.
- **Compatibilidad**: El controlador de Ext4 es totalmente compatible con Ext2, lo que permite montar sistemas de archivos Ext2 con el controlador de Ext4 para obtener beneficios como soporte de fechas de 64 bits.

4. Limitaciones y Desventajas

- **Falta de Journaling**: Es su principal desventaja, lo que lleva a comprobaciones de consistencia más largas (fsck) después de un cierre sucio y un mayor riesgo de pérdida de datos. Un sistema Ext2 puede experimentar paradas periódicas durante el arranque mientras ejecuta fsck.
- **Fechas de 32 bits**: El controlador de Ext2 soporta marcas de tiempo solo hasta el 19 de enero de 2038.
- **Deprecación del Controlador**: Debido a la limitación de fechas de 32 bits, el controlador del sistema de archivos Ext2 ha sido marcado como obsoleto en el kernel de Linux 6.9. Se recomienda a los usuarios actualizar a Ext4 o montar sus sistemas de archivos Ext2 con el controlador de Ext4 para obtener soporte para fechas de 64 bits.
- **Rendimiento en Directorios Grandes**: Sin el indexado de directorios, el rendimiento de Ext2 puede ser un problema en directorios con un gran número de archivos (>10,000).
- **Límites de Tamaño**: Los límites teóricos incluyen un tamaño máximo de volumen de 2-32 TiB y un tamaño máximo de archivo de 16 GiB - 2 TiB, dependiendo del tamaño del bloque.

5. Evolución (Ext3 y Ext4) Ext2 sentó las bases para el desarrollo de sus sucesores, Ext3 y Ext4:

- **Ext3**: Introdujo el **Journaling** en 2001, mejorando drásticamente la fiabilidad y la velocidad de recuperación después de un fallo. Era compatible hacia atrás con Ext2, permitiendo actualizaciones sin reformatear.
- **Ext4**: Introducido en 2008, abordó las crecientes demandas de almacenamiento, añadiendo soporte para archivos más grandes (hasta 16TB individual y 1EB de volumen), gestión más eficiente de directorios grandes, marcas de tiempo mejoradas, sumas de verificación para el journal y asignación diferida.

III. Arquitectura del Sistema

A. Visión general

El sistema **GoDisk** fue diseñado bajo una arquitectura cliente–servidor.

- El **Frontend** proporciona la interfaz gráfica en un entorno web donde los usuarios pueden introducir comandos manualmente o cargar scripts `[.smia]`.
- El **backend**, desarrollado en **Go (Golang)**, se encarga de interpretar los comandos, ejecutar las operaciones sobre el sistema de archivos EXT2 simulado y devolver los resultados al Frontend.
- Los **archivos `[.mia]`** actúan como discos virtuales, almacenando todas las estructuras internas (MBR, EBR, Superbloque, inodos, bloques, bitmaps).

De esta manera, el sistema combina un entorno accesible e interactivo para el usuario con un motor eficiente y seguro que simula las operaciones internas de un sistema de archivos real.

B. Frontend

El **Frontend** tiene como responsabilidades principales:

- Proveer un área de entrada de comandos y scripts.
- Mostrar en el área de salida los resultados de los comandos procesados.
- Permitir la ejecución secuencial de scripts **`[.smia]`**, manteniendo la visualización de mensajes y comentarios.
- Gestionar la comunicación con el backend mediante llamadas a APIs RESTful.

C. Backend en Go

El **backend** es el núcleo del sistema y está desarrollado en **Go (Golang)**.

Sus responsabilidades incluyen:

- Interpretar los comandos recibidos desde el Frontend, mediate el uso de la herramienta **ANTLR4**, en el Backend.
- Gestionar los archivos `[.mia]` que simulan los discos.
- Implementar las estructuras internas del EXT2: MBR, EBR, Superbloque, inodos, bloques y bitmaps.
- Manejar usuarios, permisos, carpetas y archivos.
- Generar reportes visuales mediante **Graphviz**, devolviendo archivos de imagen o texto al Frontend.

El backend expone una API RESTful, asegurando que cada operación pueda ser llamada desde la interfaz web de forma clara y modular.

D. Flujo de interacción Frontend–Backend

1. El usuario ingresa un comando o carga un script en el Frontend.
2. El Frontend envía la solicitud al backend mediante una API REST.
3. El backend procesa el comando, actualiza o consulta las estructuras del archivo `[.mia]`.

4. El resultado de la operación se devuelve al Frontend y se muestra en el área de salida.
5. En caso de comandos de reporte, el backend genera archivos gráficos o de texto con Graphviz, que pueden visualizarse o descargarse desde la interfaz.

E. Diagrama de arquitectura

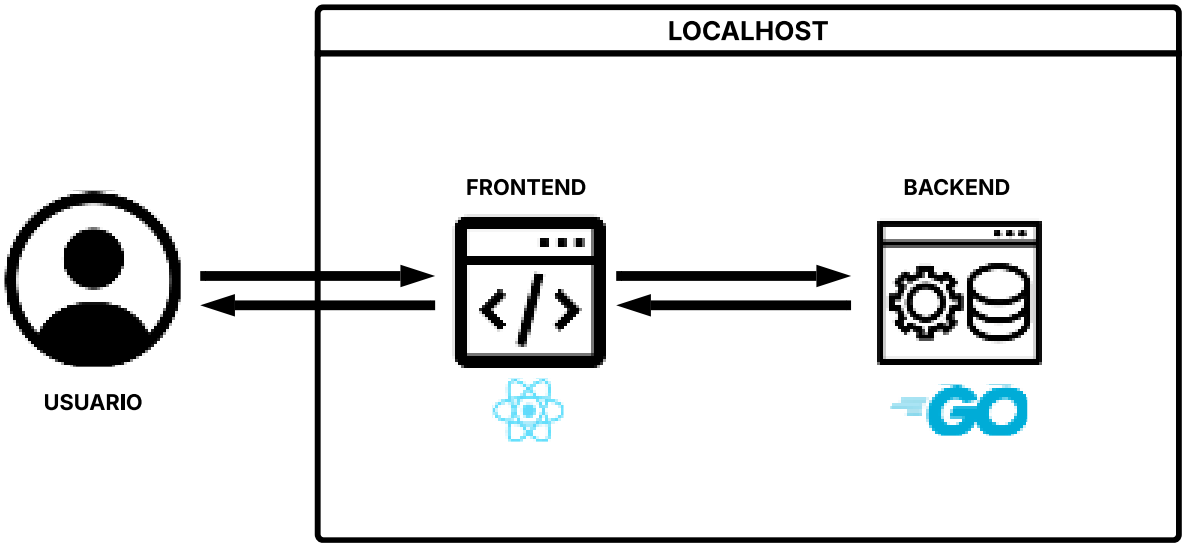

Figura 1: Diagrama general

IV. Estructuras de Datos Implementadas

A. Discos y Particiones

1. Master Boot Record (MBR)

El **MBR** es la primera estructura escrita en un disco virtual `[.mia]`.

Contiene:

- Tamaño total del disco.
- Fecha de creación.
- Identificador único (disk signature).
- Ajuste de particiones (Best, First o Worst fit).
- Tabla de hasta 4 particiones.

Su función es gestionar la información general del disco y permitir el acceso a las particiones.

2. Particiones

Cada partición es una división lógica dentro del disco. Pueden ser:

- **Primarias:** permiten almacenar información directamente.
- **Extendidas:** contienen particiones lógicas, manejadas a través de estructuras EBR.
- **Lógicas:** se crean dentro de una extendida y permiten segmentar aún más el disco.

3. Extended Boot Record (EBR)

El **EBR** describe cada partición lógica dentro de una extendida, formando una lista enlazada. Contiene:

- Nombre de la partición.
- Estado (montada o no).
- Tamaño y posición en disco.
- Referencia a la siguiente partición lógica.

B. Sistema de Archivos EXT2

1. Superbloque

El **Superbloque** describe la configuración del sistema de archivos.

Almacena:

- Cantidad total de inodos y bloques.
- Cantidad de inodos y bloques libres.
- Fechas de montaje y desmontaje.
- Tamaños de estructuras.
- Punteros al inicio de bitmaps, tablas de inodos y bloques.

2. Inodos

Los **inodos** son estructuras que describen archivos o carpetas. Incluyen:

- UID y GID (usuario y grupo propietario).
- Tamaño del archivo.
- Tiempos de creación, modificación y acceso.
- Punteros directos e indirectos a bloques.
- Tipo (archivo o carpeta).
- Permisos (UGO en octal).

Cada archivo o carpeta en el sistema debe tener un inodo asociado.

3. Bloques (carpeta, archivo, apuntadores)

Existen tres tipos:

- **Bloques de carpetas:** guardan referencias a archivos y subcarpetas.
- **Bloques de archivos:** almacenan directamente contenido (64 bytes cada bloque).
- **Bloques de apuntadores:** permiten manejar bloques indirectos (simple, doble, triple).

4. Bitmaps (inodos y bloques)

Los **bitmaps** indican disponibilidad de estructuras:

- Bitmap de inodos: 0 = libre, 1 = ocupado.
- Bitmap de bloques: 0 = libre, 1 = ocupado.

Estos permiten al sistema localizar rápidamente dónde escribir nuevas estructuras

C. Relación entre estructuras en los archivos .mia

El sistema organiza la información de la siguiente manera:

- El **disco** `[.mia]` contiene el **MBR** en el primer sector.
- El **MBR** referencia particiones (primarias o extendidas).
- Una **partición formateada en EXT2** inicia con un **Superbloque**, seguido de bitmaps, tabla de inodos y bloques de datos.
- Cada **inodo** apunta a bloques, que pueden ser carpetas o archivos.
- Los **bitmaps** garantizan el control de recursos usados/libres.

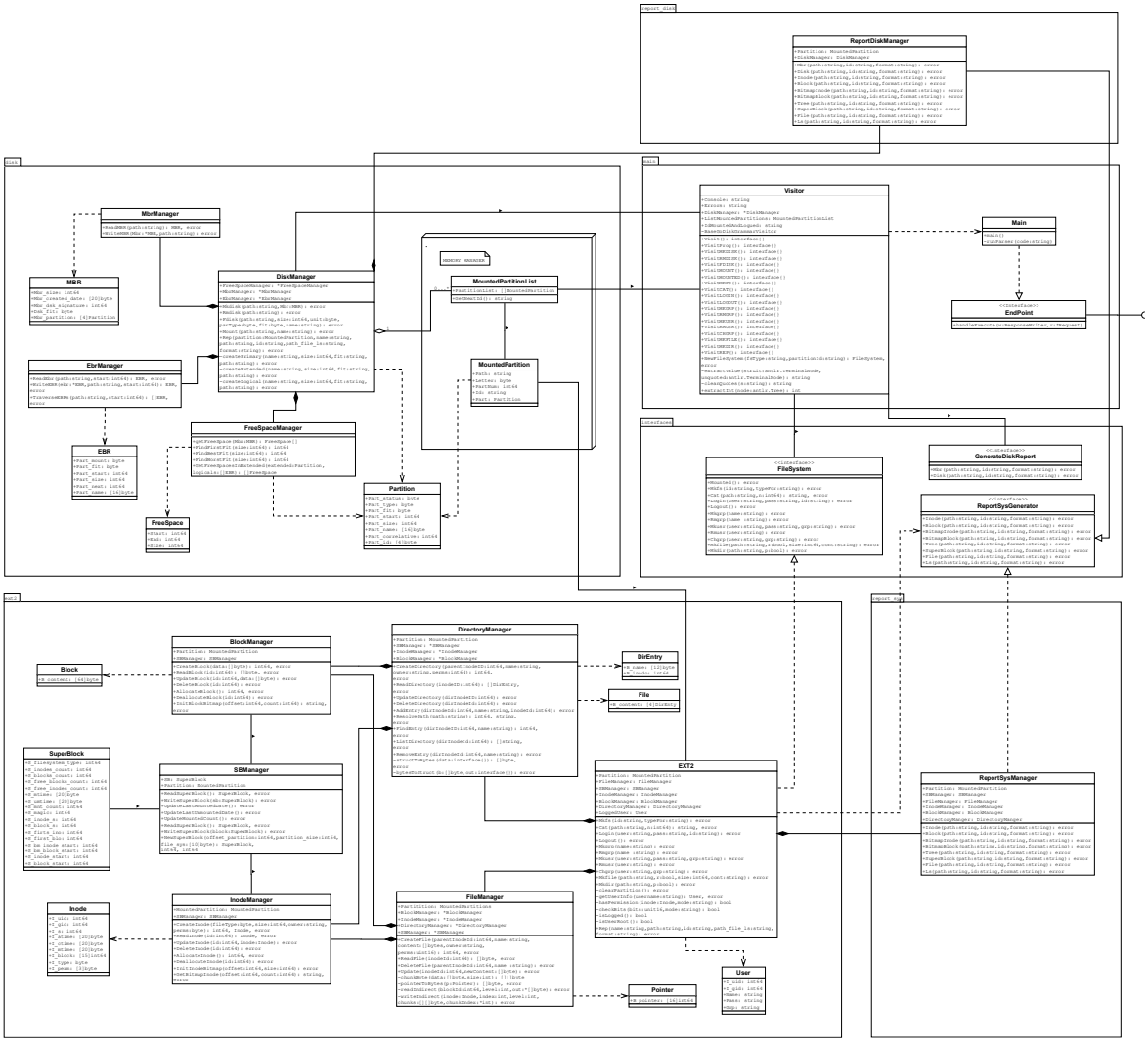

Figura 2: Diagrama de clases (UML).

V. Comandos Implementados

A. Administración de Discos

- FDISK**:** Administra particiones dentro de un disco (creación, eliminación, redimensionamiento).
- MOUNT**:** Monta una partición en el sistema y le asigna un identificador.
- MOUNTED**:** Lista las particiones actualmente montadas.

B. Administración del Sistema de Archivos

- MKDIR: Crea nuevas carpetas en el sistema de archivos.
- MKFILE: Crea archivos con contenido definido o tamaño específico.

C. Manejo de Usuarios y Grupos

- LOGIN: Inicia sesión en una partición montada, validando usuario y contraseña.
- LOGOUT: Cierra la sesión actual.
- MKGRP: Crea un nuevo grupo de usuarios.
- RMGRP: Elimina un grupo existente.
- MKUSR: Crea un usuario, asignado a un grupo.
- RMUSR: Elimina un usuario del sistema.
- CHGRP: Cambia el grupo asociado a un usuario.

D. Manejo de Archivos y Carpetas

- MKDISK**:** Crea un nuevo disco virtual `[.mia]` con un tamaño definido.
- RMDISK**:** Elimina un disco virtual existente.

E. Reportes

- REP: Genera reportes visuales o textuales de las estructuras internas (MBR, inodos, bloques, Superbloque, etc.) usando Graphviz.

VI. Reportes

El sistema **GoDisk** cuenta con un conjunto de reportes que permiten visualizar y comprender mejor el estado interno de los discos y del sistema de archivos EXT2. Estos reportes se generan mediante el comando **REP**, y se apoyan en la librería **Graphviz** para su representación gráfica.

A continuación, se describen los principales reportes implementados:

1. MBR

   Muestra la información del **Master Boot Record**, incluyendo tamaño del disco, fecha de creación, tipo de ajuste y las particiones existentes.

2. Disk

   Presenta una representación gráfica del disco, mostrando de forma secuencial las particiones (primarias, extendidas y lógicas) y los espacios libres.

3. Inode

   Genera un reporte con la tabla de **inodos**, detallando los atributos principales de cada uno: usuario, grupo, tamaño, fechas, permisos y apuntadores a bloques.

4. Block

   Muestra los distintos **bloques** del sistema (carpetas, archivos y apuntadores), con su respectiva información.

5. Bitmap Inode y Bitmap Block

   Reportes en formato de texto que muestran el estado de los **bitmaps**, indicando qué inodos y bloques están libres u ocupados.

6. Tree

   Representa el sistema de archivos en forma de árbol, mostrando directorios y archivos con sus relaciones jerárquicas.

7. SB (Superbloque)

   Despliega la información del **Superbloque**, incluyendo los conteos de estructuras, posiciones en disco y referencias a bitmaps, inodos y bloques.

8. File

   Permite generar un reporte con el contenido de un archivo específico dentro del sistema.

9. Ls

   Muestra un listado de los archivos y carpetas presentes en un directorio, incluyendo permisos, propietario, grupo y fecha de modificación.

VII. Diseño e Implementación

A. Estructura del código Backend

El backend fue desarrollado en **Go (Golang)**, organizado en módulos y paquetes para separar responsabilidades.

Las principales áreas de implementación son:

- **Manejo de discos y particiones:** lectura y escritura del MBR, creación y eliminación de particiones, gestión de EBR.
- **Sistema de archivos EXT2:** creación del Superbloque, bitmaps, tabla de inodos y bloques de datos.

- **Operaciones de usuario y seguridad:** login, logout, creación/eliminación de usuarios y grupos.
- **Gestión de archivos y carpetas:** funciones para crear directorios, archivos y manejar sus permisos.
- **Reportes:** generación de archivos DOT y su conversión a imágenes mediante Graphviz.

B. API REST

El backend expone sus funcionalidades a través de una **API RESTful**, lo que permite la comunicación con el Frontend.

- El Frontend envía peticiones y el backend procesa la operación en los archivos `[.mia]`.
- Las respuestas incluyen mensajes de éxito o error, y en el caso de reportes, archivos o rutas para su visualización.

C. Frontend

El Frontend, desarrollado con un framework web moderno, proporciona la interfaz de interacción con el usuario.

Sus componentes principales son:

- **Área de entrada:** donde el usuario escribe comandos o carga scripts **`[.smia]`**.
- **Área de salida:** donde se muestran mensajes, errores y resultados.
- **Gestión de scripts:** permite ejecutar secuencias de comandos de forma automática.
- **Visualización de reportes:** integración con el backend para mostrar los archivos generados por Graphviz.

D. Ejecución de un comando paso a paso

El proceso completo de ejecución de un comando sigue los siguientes pasos:

1. El usuario ingresa un comando en la interfaz del Frontend.
2. El Frontend envía la instrucción al backend mediante la API REST.
3. El backend interpreta el comando, localiza la estructura correspondiente en el archivo `[.mia]` y ejecuta la operación solicitada.
4. Se actualizan las estructuras internas (MBR, Superbloque, inodos, bloques, etc.) según corresponda.
5. El backend responde al Frontend con el resultado (mensaje, error, o archivo de reporte).
6. El Frontend muestra la información al usuario.

E. Diseño en Capas del Sistema EXT2

Para la implementación del sistema de archivos se adoptó un **modelo en tres capas**, que permite separar responsabilidades y mantener el código modular y escalable:

1. **Block e Inode Manager (bajo nivel):**
    - Responsable de la manipulación directa de estructuras básicas como bloques e inodos.
    - Incluye operaciones de lectura, escritura y control mediante bitmaps.
    - Representa la capa más cercana al disco virtual (.mia).
2. **File y Directory Manager (nivel medio):**
    - Encargado de administrar archivos y directorios.
    - Utiliza las funciones de la capa de bajo nivel para localizar inodos y bloques disponibles.

- Implementa la lógica de creación, eliminación y acceso a archivos y carpetas.

3. **File System (alto nivel):**
    - Define las operaciones principales del sistema, como mkfile, mkdir, cat, entre otras.
    - Interactúa con el usuario (a través de los comandos recibidos) y traduce esas instrucciones en acciones concretas sobre archivos y directorios.
    - Representa la capa más abstracta y cercana a la experiencia del usuario.

Este enfoque permitió mantener un **acoplamiento bajo** entre las distintas funciones y facilitó la depuración del sistema, ya que cada capa cumple un rol bien definido.

VIII. Manual de Uso

A. Instalación y requisitos del sistema

1. Requisitos del sistema
    - Sistema operativo: Linux.
    - Tener instalado **Go (Golang)** para ejecutar el backend.
    - Navegador web moderno para acceder al Frontend.
    - Librería **Graphviz** instalada para la generación de reportes.
2. Instalación
    - Clonar o descargar el repositorio del proyecto.
    - Compilar y ejecutar el backend en Go:

```
bash
go run main.go
```

- Iniciar el Frontend de React (previa instalación npm).

```bash
npm start
```

- Abrir el navegador en la dirección del servidor

B. Uso de la Interfaz Web

- El **área de entrada** permite escribir comandos individuales o cargar archivos de script `[.smia]`.
- El **área de salida** muestra mensajes de confirmación, errores o resultados de los comandos.
- Al ejecutar comandos de **reportes**, se generan archivos gráficos que se pueden visualizar desde el Frontend.

C. Flujo de ejemplo

Un flujo típico de trabajo en **GoDisk** es:

1. Crear un disco

```bash
mkdisk -size=50 -unit=M -path=/home/user/disk1.mia
```

2. Crear y montar una partición

```bash
fdisk -size=10 -unit=M -path=/home/user/disk1.mia -name=Part1
mount -path=/home/user/disk1.mia -name=Part1
```

3. Formatear la partición en EXT2

```bash
mkfs -id=vd1
```

4. Crear usuarios y grupos

```bash
mkgrp -name=admin
mkusr -user=jose -pwd=123 -grp=admin
```

5. Iniciar sesión

```bash
login -user=jose -pwd=123 -id=vd1
```

6. Crear carpetas y archivos

```bash
mkdir -path=/home/docs
mkfile -path=/home/docs/info.txt -size=100
```

7. Generar reportes

```bash
rep -id=vd1 -path=/home/user/reports/mbr.png -name=mbr
```

IX. Limitaciones y Consideraciones

A. Restricciones del proyecto

El desarrollo de **GoDisk** presenta algunas restricciones importantes a considerar:

- Al estar basado en la emulación de **EXT2**, un sistema de archivos antiguo no ofrece soporte para el manejo de archivos de gran tamaño, lo que limita su aplicabilidad a escenarios simples.
- El tiempo de testeo del sistema fue reducido, lo cual puede implicar que ciertas operaciones no estén cubiertas exhaustivamente.
- No se realizaron pruebas en un entorno de producción real, únicamente en entornos controlados de desarrollo.

B. Diferencias con EXT2 real

Aunque GoDisk se basa en el sistema de archivos **EXT2**, existen diferencias relevantes con respecto a su implementación real:

- No se utilizó una **tabla de inodos** ni una tabla de **bloques** como en EXT2; en su lugar, se implementó únicamente el control a través de **bitmaps**.
- No se implementó el **Block Group Descriptor**, presente en la especificación real de EXT2 para manejar grupos de bloques de forma más eficiente.

Estas simplificaciones fueron necesarias para mantener el proyecto dentro de los alcances definidos en el curso y permitir una implementación más manejable.

C. Posibles mejoras futuras

Aunque **GoDisk** cumple con los objetivos planteados en el proyecto, existen varias áreas que podrían ampliarse o mejorarse en un futuro:

- **Soporte para archivos de mayor tamaño:** implementar cálculos más precisos sobre los límites actuales y añadir soporte para archivos grandes mediante bloques indirectos dobles y triples.
- **Implementación de tablas completas de inodos y bloques:** actualmente solo se utilizan bitmaps, pero incorporar estas estructuras permitiría acercarse más al estándar real de EXT2.
- **Incorporación del Block Group Descriptor:** mejorar la organización interna de los bloques y la eficiencia en el manejo de inodos, alineándose con la arquitectura del EXT2 real.
- **Mayor cobertura de pruebas:** diseñar pruebas automatizadas que cubran casos límite (archivos grandes, múltiples usuarios, eliminación masiva, etc.).

- **Pruebas en entornos de producción simulados:** validar la estabilidad del sistema bajo carga y múltiples usuarios concurrentes.
- **Mejoras en el Frontend:** incluir paneles visuales para explorar directorios de manera gráfica, además de la consola de comandos.
- **Compatibilidad multiplataforma extendida:** empaquetar el sistema en contenedores (ej. Docker) para facilitar la ejecución en distintos sistemas operativos sin configuraciones manuales.
- **Seguridad y auditoría:** implementar bitácoras de operaciones, control de permisos más estricto y cifrado en los archivos `[.mia]`.

X. Plan de Mantenimiento

El mantenimiento de **GoDisk** es fundamental para garantizar su correcto funcionamiento a lo largo del tiempo, adaptarlo a nuevas necesidades y corregir posibles fallos. A continuación, se establecen las principales directrices de mantenimiento:

1. **Mantenimiento Correctivo**

   Identificación y corrección de errores detectados durante la ejecución de comandos.

   Solución de problemas relacionados con la manipulación de archivos binarios `[.mia]`.

   Ajustes en los reportes generados cuando se encuentren inconsistencias gráficas o de datos.

2. **Mantenimiento Preventivo**

   Revisión periódica del código para detectar posibles vulnerabilidades o malas prácticas.

Pruebas regulares de compatibilidad con diferentes sistemas operativos y versiones de Go.

Verificación de la correcta generación de reportes con Graphviz.

3. **Mantenimiento Evolutivo**

   Incorporación de nuevas funcionalidades, como soporte para archivos de mayor tamaño o manejo más avanzado de permisos.

   Ampliación del Frontend con herramientas gráficas para la exploración de directorios.

   Integración con tecnologías de contenedores (Docker) para mejorar la portabilidad y despliegue del sistema.

4. **Mantenimiento Adaptativo**

   Ajustes en el sistema para adaptarse a nuevas versiones de dependencias externas (Graphviz, React + TypeScript, librerías de Go).

   Incorporación de cambios que faciliten la ejecución en entornos más modernos y productivos.

XI. Conclusiones

A. Aprendizajes técnicos

Durante el desarrollo de **GoDisk** se fortalecieron conocimientos clave en el área de sistemas operativos y manejo de archivos:

- Implementación de un sistema de archivos tipo **EXT2**, comprendiendo el rol del MBR, Superbloque, inodos, bloques y bitmaps.
- Uso de **Go (Golang)** para manipular archivos binarios y representar estructuras de bajo nivel.

- Diseño de una **API REST** que permite la comunicación entre Frontend y backend bajo una arquitectura cliente-servidor.
- Generación de **reportes gráficos** con Graphviz para visualizar estructuras internas que normalmente son abstractas.
- Se comprendió la importancia de aplicar un **diseño en capas** en la construcción de un sistema de archivos, lo cual permitió separar las operaciones de bajo, medio y alto nivel, facilitando la organización del código y su mantenibilidad.

B. Aprendizajes prácticos

El proyecto también representó un reto en términos de gestión y organización:

- La planificación del tiempo fue clave para poder implementar las distintas fases (manejo de discos, sistema de archivos, usuarios, reportes).
- Fue necesario organizar el código en módulos claros para evitar duplicidad y facilitar el mantenimiento.
- Se enfrentaron retos técnicos como la representación simplificada de ciertas estructuras y la depuración de errores al trabajar con archivos binarios.
- El proyecto mostró la importancia de realizar pruebas constantes durante la implementación para asegurar la estabilidad del sistema.

XII. Anexos

REPORTE DE MBR	
mbr_tamano	52428800
mbr_fecha_creacion	2025-09-14 19:05:19
mbr_disk_signature	6147972721345830244
Particion	
part_status	1
part_type	p
part_fit	b
part_start	256
part_size	10485760
part_name	Part11
Particion	
part_status	1
part_type	p
part_fit	b
part_start	10486016
part_size	10485760
part_name	Part12
Particion	
part_status	49
part_type	p
part_fit	b
part_start	20971776
part_size	10485760
part_name	Part13
Particion	
part_status	49
part_type	p
part_fit	b
part_start	31457536
part_size	10485760
part_name	Part14

Reporte de MBR

Figura 3: Reporte MBR

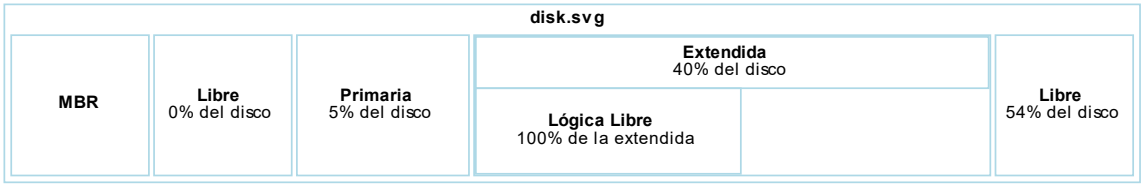

Figura 4: Reporte DISK

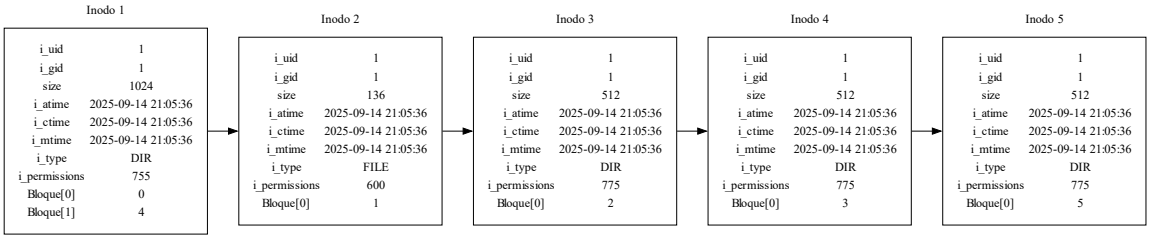

Figura 5: Reporte INODE

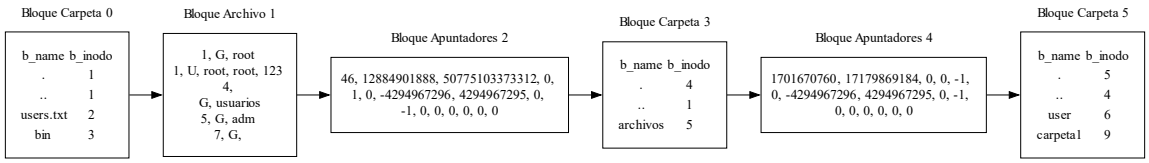

Figura 6: Reporte BLOCK

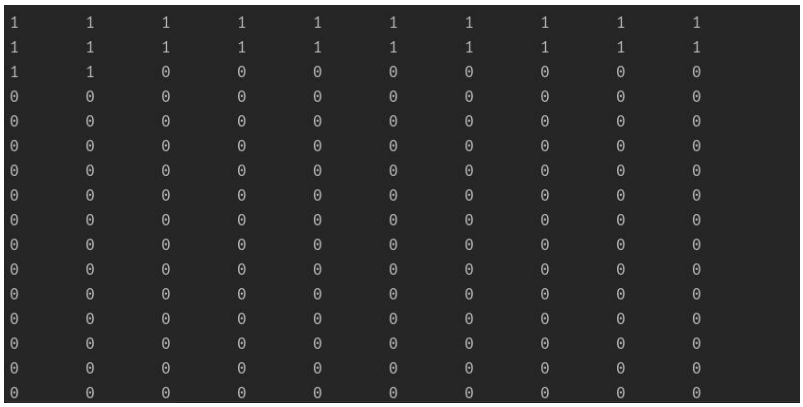

Figura 7: Reporte BM_INODE

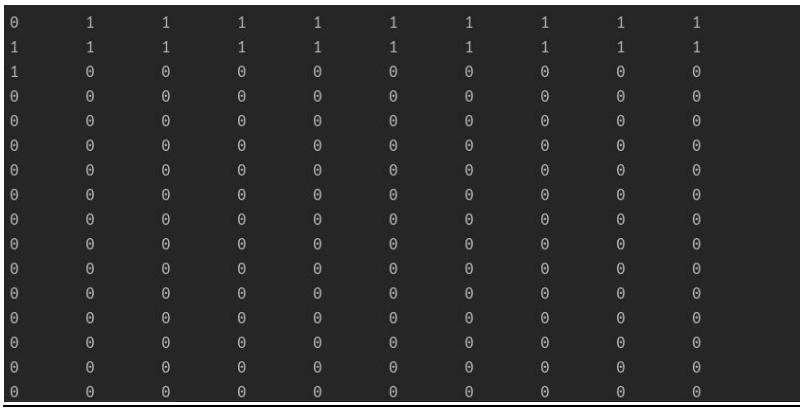

Figura 8: Reporte BM_BLOCK

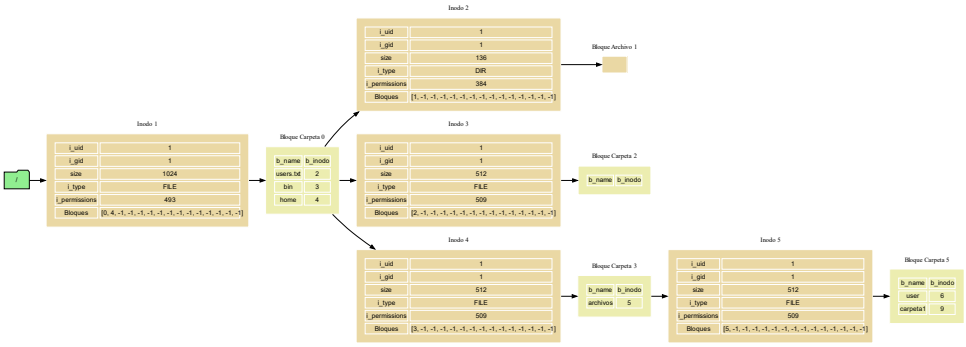

Figura 8: Reporte TREE

REPORTE SUPERBLOCK	
s_filesystem_type	EXT2
s_inodes_count	2560
s_blocks_count	20479
s_free_blocks_count	20457
s_free_inodes_count	2540
s_mtime	-
s_umtime	-
s_mnt_count	0
s_magic	61267
s_inode_size	256
s_block_size	512
s_first_ino	1
s_first_blo	1
s_bm_inode_start	768
s_bm_block_start	1088
s_inode_start	3648
s_block_start	659008

Reporte de SuperBlock

Figura 9: Reporte SB

LS Report							
Directorio: /home/archivos/user/docs							
Permisos	Owner	Grupo	Size (en Bytes)	Fecha	Hora	Tipo	Name
drwxrwxr-x	UID:1	GID:1	1024	14/09/2025	19:05	Carpeta	.
drwxrwxr-x	UID:1	GID:1	512	14/09/2025	19:05	Carpeta	..
drwxrwxr-x	UID:1	GID:1	512	14/09/2025	19:05	Carpeta	usac
-rw-rw-r--	UID:1	GID:0	75	14/09/2025	19:05	Archivo	Tarea.txt
-rw-rw-r--	UID:1	GID:0	768	14/09/2025	19:05	Archivo	Tarea2.txt
-rw-rw-r--	UID:1	GID:0	17	14/09/2025	19:05	Archivo	Tarea3.txt

Figura 10: Reporte LS

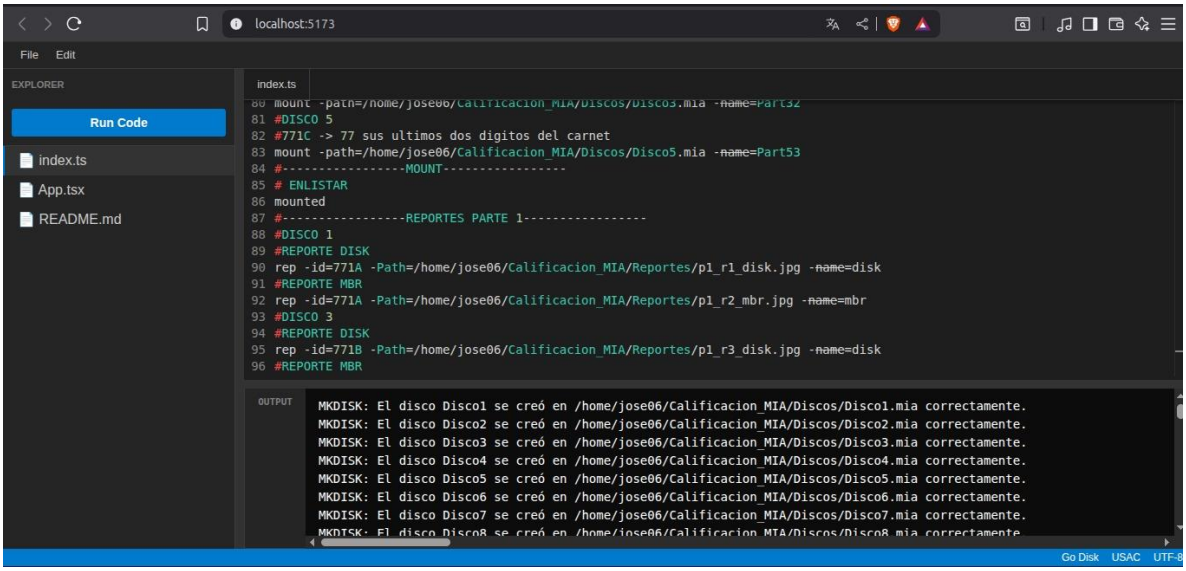

Figura 11: Interfaz de GoDisk

Sprint	Objetivos principales	Actividades	Fecha de inicio
Sprint 1: Implementación del Backend Básico	Construir la base del sistema de discos virtuales y particiones.	-MKDISK -FDISK -MOUNT -MBR, EBR	28/07/2025
Sprint 2: Sistema de Archivos EXT2	Implementar el sistema de archivos.	- **MKFS** -Implementación de **inodos, bloques y bitmaps**. **-MKUSR, RMUSR, MKGRP, RMGRP**.	4/08/2025
Sprint 3: Frontend y Comunicación con el Backend	Desarrollar la interfaz y conectarla con el sistema.	- I**nterfaz WEB** - Conexión con backend mediante **APIs RESTful**. - Visualización de resultados de comandos.	11/08/2025
Sprint 4: Comandos de Archivos y Carpetas	Gestionar archivos y directorios dentro del FS.	- Creación de archivos y carpetas -Manejo de permisos y usuarios -Implementación del comando **CAT** para lectura de archivos.	13/08/2025
Sprint 5: Generación de Reportes	Permitir visualización gráfica de estructuras.	**-REP** **-MBR, disco, inodos, bloques, etc.**.	1/09/2025
Sprint 6: Pruebas y Documentación	Garantizar calidad y entregar documentación final.	- **Pruebas** del sistema. - **Depuración de errores**. - Redacción del **Manual Técnico** (arquitectura, estructuras, comandos).	12/09/2025

Figura 11: Cronograma de actividades

XIII. Bibliografía

1. Bootlin. (2024, 25 de Marzo). *ext2 filesystem driver now marked as deprecated*. Recuperado de https://bootlin.com/blog/ext2-filesystem-driver-now-marked-as-deprecated bootlin.com
2. Wikipedia. (s. f.). *Ext2*. En *Wikipedia, The Free Encyclopedia*. Recuperado de https://en.wikipedia.org/wiki/Ext2 Wikipedia
3. Unix & Linux Stack Exchange. (2013, 15 de Junio). Unable to mount ext2 hard drive. Recuperado de https://unix.stackexchange.com/questions/79530/unable-to-mount-ext2-hard-drive Unix & Linux Stack Exchange
4. Number Analytics. (s. f.). *Mastering Ext2 for Enhanced Digital Forensics*. Recuperado de https://www.numberanalytics.com/blog/mastering-ext2-digital-forensics Number Analytics
5. EaseUS. (s. f.). *Complete Ext2/Ext3 Recovery Guide: Linux Data Recovery*. Recuperado de https://www.easeus.com/data-recovery/recover-ext2-ext3-drive.html EaseUS
6. Linux Journal. (s. f.). *A Non-Technical Look inside the EXT2 File System*. Recuperado de https://www.linuxjournal.com/article/2151 linuxjournal.com
7. L-Soft. (s. f.). *A Deep Dive into Linux File Systems: Ext2, Ext3, and Ext4*. Recuperado de https://www.lsoft.net/posts/linux-Ext2-Ext3-Ext4 lsoft.net
8. The Second Extended File System. (s. f.). *The Second Extended Filesystem — The Linux Kernel documentation / ext2 specification*. Recuperado de https://www.nongnu.org/ext2-doc/ext2.pdf