

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Manejo e Implementación de Archivos 2S2025

Go disk Manual Técnico

Keitlyn Valentina Tunchez Castañeda
202201139

1. Introducción

Este manual describe la arquitectura, las estructuras de datos y los comandos implementados en la aplicación **ExtreamFS**, cuyo objetivo es simular un sistema de archivos basado en **EXT2**.

La herramienta integra un **backend en Go (Golang)** y un **frontend web** moderno, permitiendo ejecutar comandos, administrar discos virtuales y visualizar reportes generados con **Graphviz**, todo en un entorno local.

2. Arquitectura del Sistema

2.1 Descripción General

La solución sigue una **arquitectura cliente–servidor**:

- **Frontend:**

Desarrollado con un framework moderno (React, Angular o Vue.js).

- Área de entrada de comandos y carga de scripts .mia.
- Área de salida para mostrar resultados.
- Botones para ejecutar scripts y visualizar reportes.

- **Backend:**

Implementado en **Go (Golang)**, responsable de:

- Simular el sistema de archivos EXT2 sobre archivos binarios .mia.
- Exponer **APIs RESTful** para que el frontend ejecute comandos.
- Generar reportes en formato gráfico (Graphviz) o texto.

La comunicación entre frontend y backend se realiza mediante **HTTP/JSON**.

3. Estructuras de Datos

El sistema simula un disco a través de un archivo binario .mia. Dentro de este se almacenan las estructuras básicas de un sistema de archivos EXT2:

3.1 MBR (Master Boot Record)

Contiene información global del disco:

- mbr_tamano: tamaño total del disco en bytes.
- mbr_fecha_creacion: fecha y hora de creación.
- mbr_dsk_signature: identificador único.
- dsk_fit: tipo de ajuste (B, F, W).
- mbr_partitions: arreglo de hasta 4 particiones.

3.2 Particiones

- **Primarias y Extendidas** (una sola extendida por disco).
- Las **lógicas** se gestionan mediante EBR.

Cada partición guarda:

- part_status, part_type, part_fit, part_start, part_s, part_name, part_correlative, part_id.

3.3 EBR (Extended Boot Record)

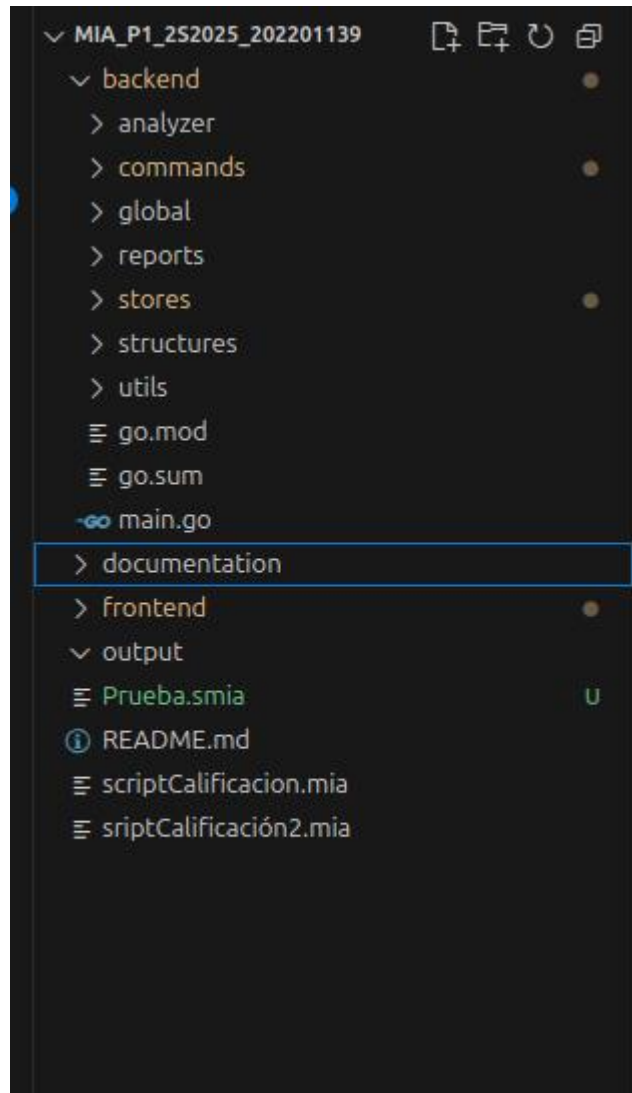
Descriptor para particiones lógicas, con:

- part_mount, part_fit, part_start, part_s, part_next, part_name.

3.4 Sistema de Archivos EXT2

Dentro de una partición formateada se crean:

- **Superbloque:** metadatos del sistema (inodos, bloques, bitmaps, posiciones de inicio).
- **Inodos:** información de cada archivo/carpeta (propietario, permisos, fechas, apuntadores a bloques).
- **Bloques:**
 - Bloque carpeta (registros b_content con nombre e inodo).
 - Bloque archivo (contenido en 64 bytes).
 - Bloque de apuntadores (enlaces indirectos simple/doble/triple).
- **Bitmaps:** control de inodos y bloques (0 = libre, 1 = ocupado).



4. Comandos Implementados

Todos los comandos se escriben en el área de entrada o en scripts .smia. No distinguen mayúsculas/minúsculas en el nombre del comando. Los parámetros pueden ir en cualquier orden; si contienen espacios, se encierran entre comillas.

4.1 Administración de Discos

- **mkdisk**: crea un archivo .mia de tamaño fijo (parámetros: -size, -path, opcionales -unit, -fit).
- **rmdisk**: elimina un archivo de disco (-path).

- **fdisk**: crea/modifica/elimina particiones (-size, -path, -name, opcionales -unit, -type, -fit).
- **mount**: monta una partición y genera un id único (-path, -name).
- **mounted**: lista las particiones montadas.

4.2 Sistema de Archivos

- **mkfs**: formatea una partición montada en EXT2, creando el archivo inicial users.txt (-id, opcional -type=full).

4.3 Gestión de Usuarios y Grupos

- **login, logout**: iniciar y cerrar sesión (-user, -pass, -id).
- **mkgrp, rmgrp**: crear/eliminar grupos (-name).
- **mkusr, rmusr**: crear/eliminar usuarios (-user, -pass, -grp).
- **chgrp**: cambiar grupo de un usuario (-user, -grp).

El archivo users.txt mantiene el registro de grupos y usuarios.

4.4 Archivos y Carpetas

- **mkfile**: crea archivos (-path, opcionales -r, -size, -cont).
- **mkdir**: crea carpetas (-path, opcional -p).
- **cat**: muestra contenido de archivos (-file1, -file2, ...).

4.5 Reportes

Generados con **Graphviz**:

- **rep**: genera reportes como mbr, disk, inode, block, bm_inode, bm_block, tree, sb, file, ls.
Parámetros: -id, -path, -name, opcional -path_file_ls.

5. Flujo de Trabajo Recomendado

1. **Crear discos**: mkdisk.
2. **Administrar particiones**: fdisk, luego mount.
3. **Formatear partición**: mkfs.
4. **Crear usuarios/grupos**: mkgrp, mkusr.

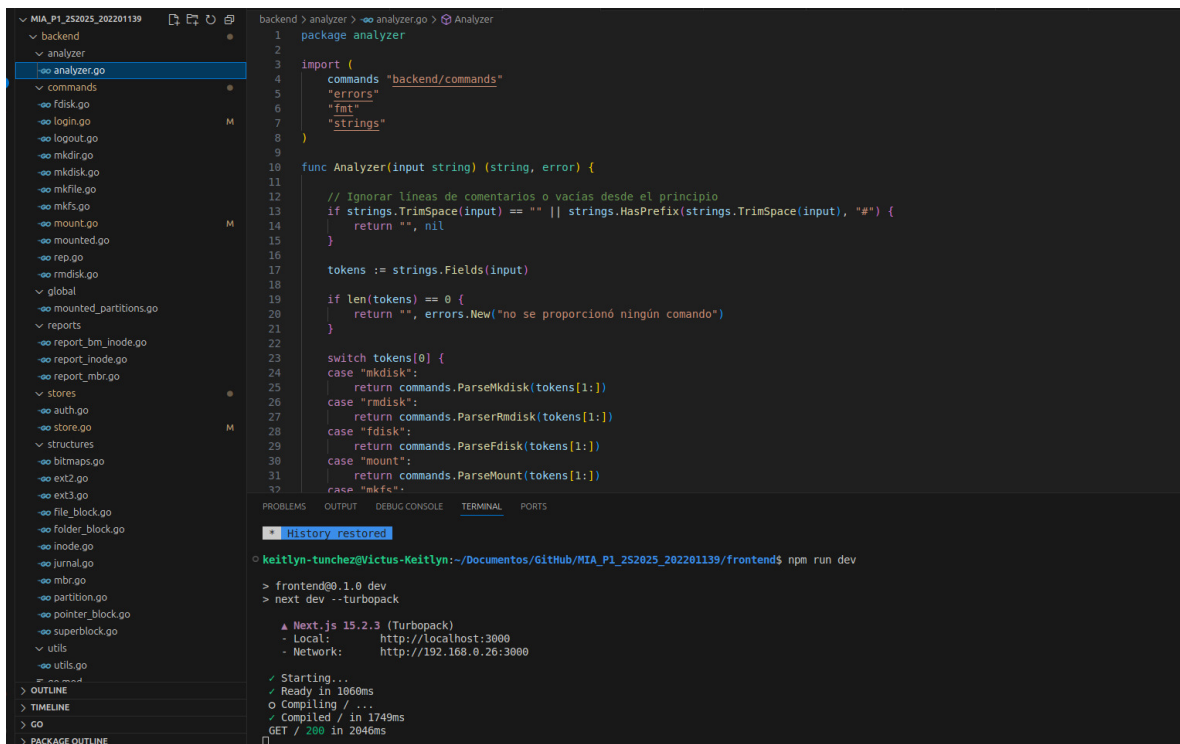
5. **Administrar archivos:** mkdir, mfile, cat.
6. **Generar reportes:** rep para validar estructuras internas.



6. Requerimientos Técnicos

- **Backend:** Go (Golang).
- **Frontend:** Framework moderno (React, Angular o Vue.js).
- **Reportes:** Graphviz.
- **Sistema Operativo:** GNU/Linux o MacOS.
- **Discos simulados:** Archivos binarios .mia.

Frontend:



The screenshot shows a VS Code editor with a Go project. The left sidebar displays a file explorer with a tree view of the project structure. The main editor area shows the code for `analyzer.go`. The terminal window at the bottom shows the output of running `npm run dev` in the frontend directory.

```
backend > analyzer > -o analyzer.go > Analyzer
1 package analyzer
2
3
4 import (
5     "errors"
6     "fmt"
7     "strings"
8 )
9
10 func Analyzer(input string) (string, error) {
11
12     // Ignorar líneas de comentarios o vacías desde el principio
13     if strings.TrimSpace(input) == "" || strings.HasPrefix(strings.TrimSpace(input), "#") {
14         return "", nil
15     }
16
17     tokens := strings.Fields(input)
18
19     if len(tokens) == 0 {
20         return "", errors.New("no se proporcionó ningún comando")
21     }
22
23     switch tokens[0] {
24     case "mkdisk":
25         return commands.ParseMkdisk(tokens[1:])
26     case "rmdisk":
27         return commands.ParserRmdisk(tokens[1:])
28     case "fdisk":
29         return commands.ParseFdisk(tokens[1:])
30     case "mount":
31         return commands.ParseMount(tokens[1:])
32     case "mkfs":
33         return commands.ParseMkfs(tokens[1:])
34     }
```

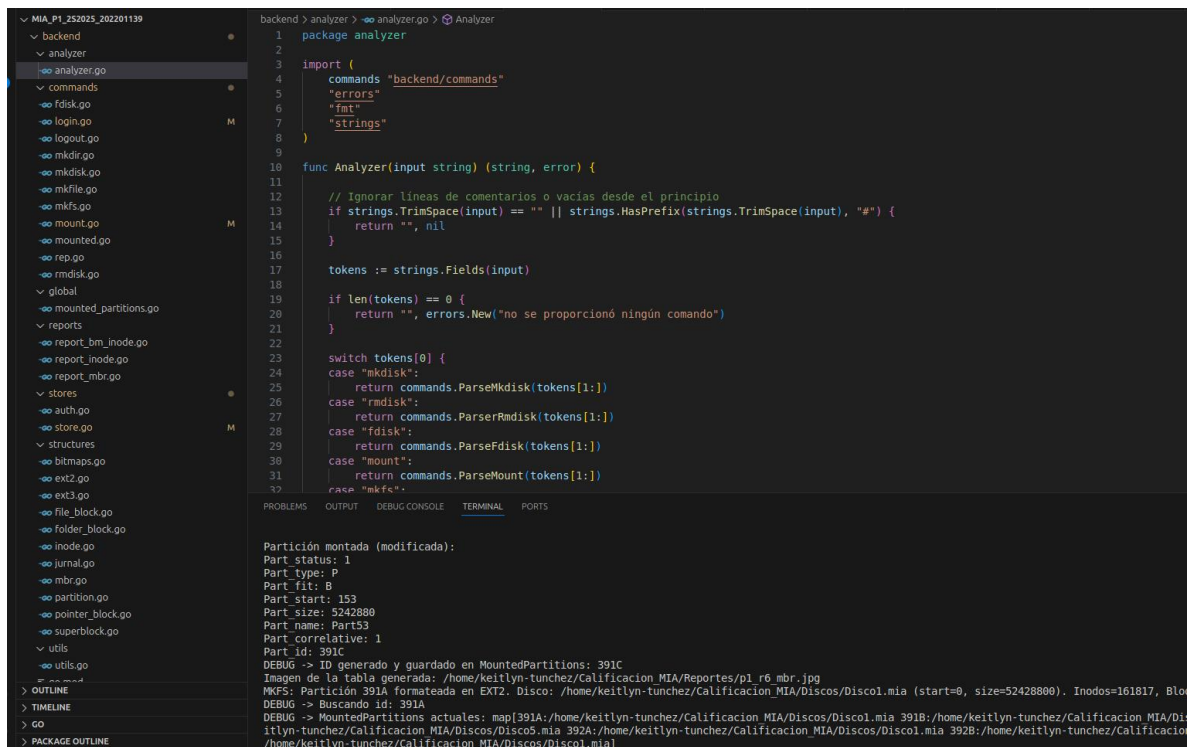
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

History restored

```
o keitlyn-tunchez@Victus-Keitlyn:~/Documentos/GitHub/MIA_P1_2S2025_202201139/frontend$ npm run dev
> frontend@0.1.0 dev
> next dev --turbo
▲ Next.js 15.2.3 (Turbopack)
- Local:    http://localhost:3000
- Network:  http://192.168.0.26:3000

✓ Starting...
✓ Ready in 1060ms
○ Compiling / ...
✓ Compiled / in 1749ms
GET / 200 in 2046ms
[]
```

Backend:



7. Mantenimiento y Extensibilidad

- El proyecto está diseñado para ejecución local, pero la arquitectura permite futuras mejoras:
 - Exportar reportes desde la interfaz.
 - Extender el sistema de archivos a EXT3.
 - Implementar control de permisos avanzados (chmod, chown en futuras fases).

8. Conclusión

ExtreamFS es una herramienta académica que permite comprender a fondo la implementación de un sistema de archivos EXT2, simulando la creación y manejo de discos, particiones, usuarios y archivos de forma práctica.