Universidad de san Carlos de Guatemala

Manejo de implementación de archivos

Daniel Monterroso

Tema Manual Tecnico

Luis Fernando Gomez Rendon

201801391

Manual Técnico: Sistema de Archivos EXT3 Simulado

Descripción de la Arquitectura del Sistema

Arquitectura General

El sistema sigue una arquitectura cliente-servidor con los siguientes componentes principales:

Frontend: Aplicación React.js que proporciona la interfaz de usuario

Backend: Servidor Go (Golang) que procesa los comandos y gestiona el sistema de archivos

AWS: Infraestructura de despliegue en la nube

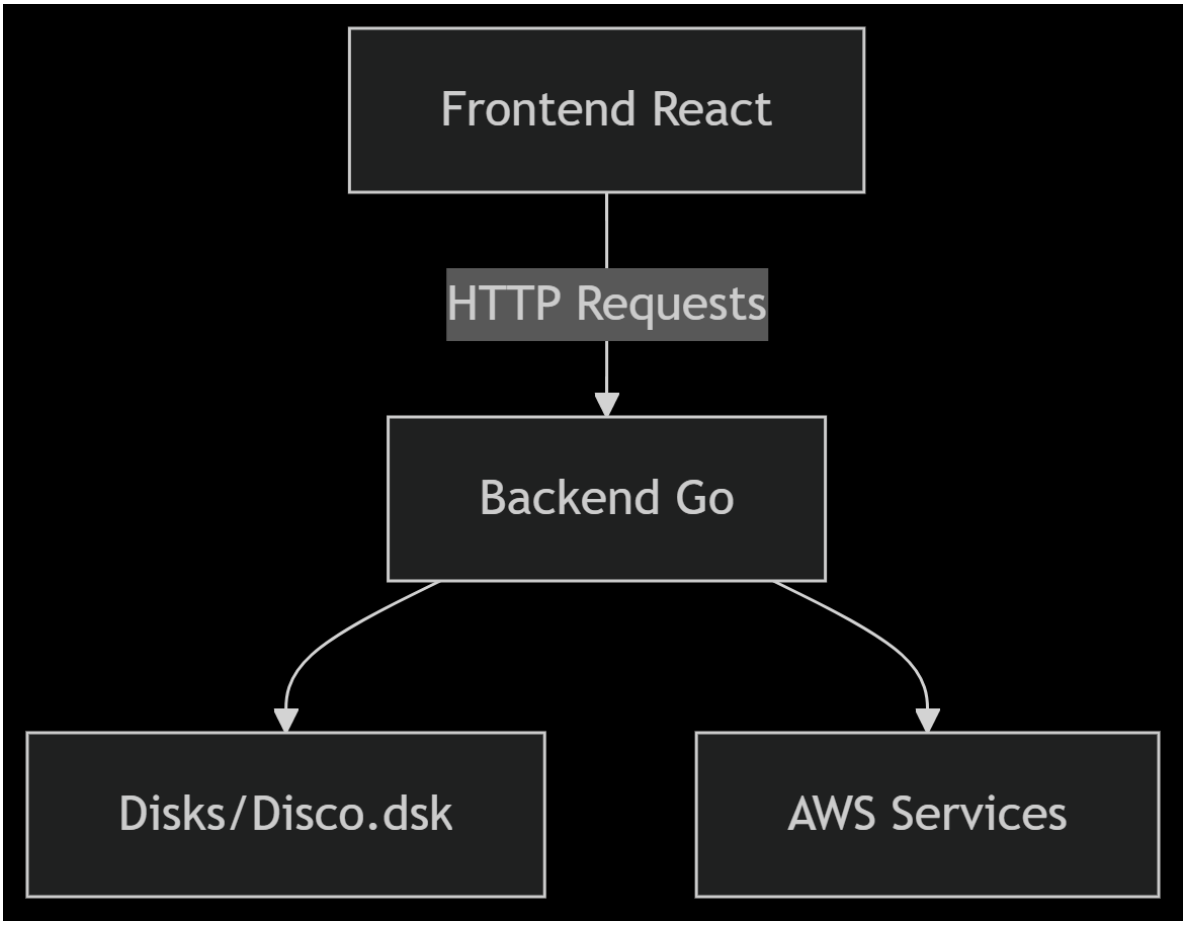
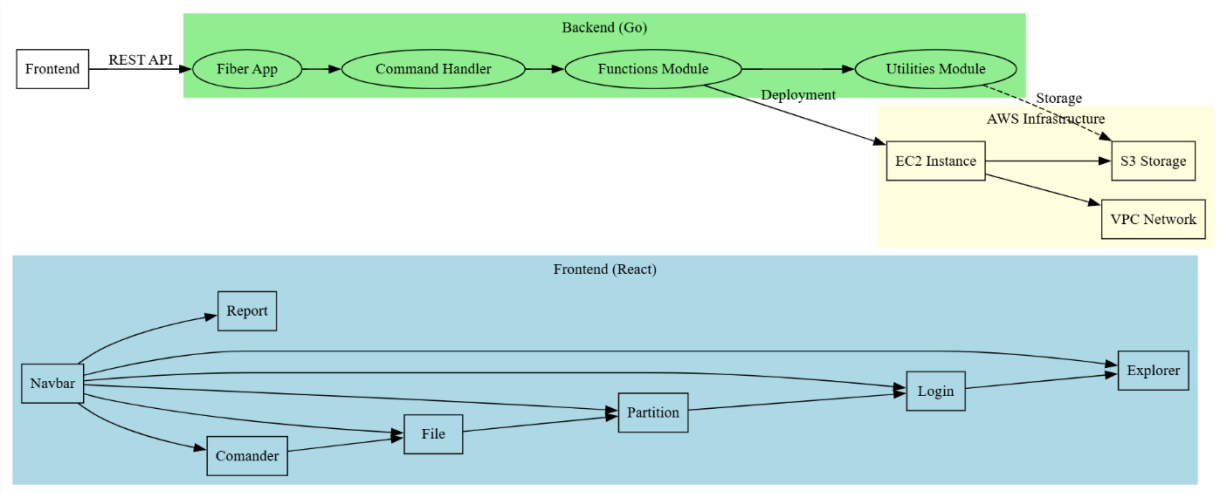


Diagrama de Componentes



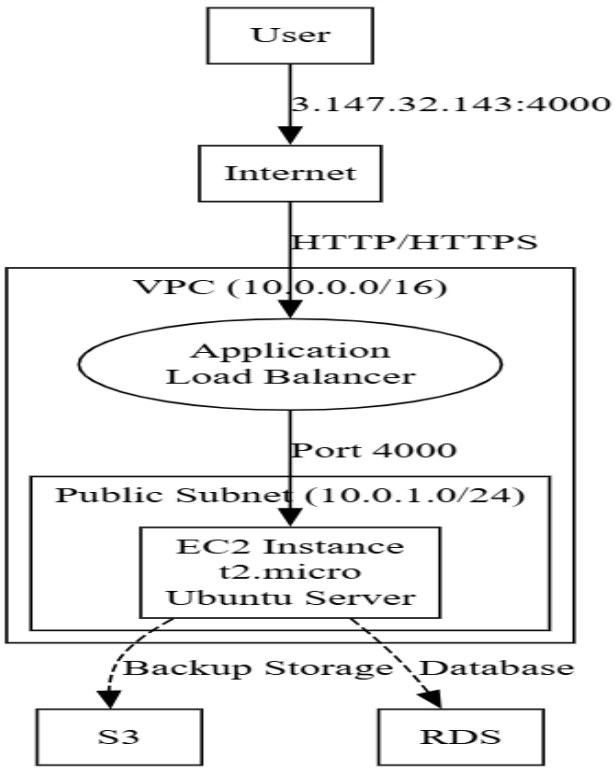
Flujo de Comunicación

1. **Frontend → Backend**:
    - Envío de comandos individuales mediante POST a /command
    - Carga de archivos de script mediante POST a /upload
    - Solicitud de información de discos mediante GET a /disks
    - Solicitud de reportes mediante GET a /reports
2. **Backend → Frontend**:
    - Respuestas con resultados de comandos
    - Envío de listados de discos y particiones
    - Generación y envío de reportes gráficos

Despliegue en AWS

La aplicación está desplegada en AWS con la siguiente configuración:

1. **EC2**: Instancia t2.micro ejecutando Ubuntu Server
2. **Security Groups**: Configuración de puertos (4000 para backend)
3. **Route 53**: DNS para el endpoint público
4. **S3**: Almacenamiento opcional para backups



Explicación de las Estructuras de Datos

MBR (Master Boot Record)

Estructura principal que contiene información sobre las particiones del disco:

```
type MBR struct {
   Mbr_tamano        int64
   Mbr_fecha_creacion [16]byte
   Mbr_dsk_signature  int64
   Dsk_fit           byte
   Mbr_partition     [4]Partition
}
```

- **Mbr_tamano**: Tamaño total del disco en bytes
- **Mbr_fecha_creacion**: Fecha de creación del disco
- **Mbr_dsk_signature**: Identificador único del disco
- **Dsk_fit**: Algoritmo de ajuste (Best, First o Worst)
- **Mbr_partition**: Array de 4 estructuras Partition

Partition

Estructura que define una partición en el disco:

go

Copy

Download

```
type Partition struct {
    Part_status byte
    Part_type   byte
    Part_fit    byte
    Part_start  int64
    Part_size   int64
    Part_name   [16]byte
}
```

- **Part_status**: Indicador de estado (activo/inactivo)
- **Part_type**: Tipo de partición (Primaria/Extendida/Logica)
- **Part_fit**: Algoritmo de ajuste para la partición
- **Part_start**: Byte de inicio de la partición
- **Part_size**: Tamaño de la partición en bytes
- **Part_name**: Nombre de la partición (16 caracteres max)

Inodo

Estructura que representa un archivo o directorio en el sistema EXT3:

go

Copy

Download

```
type Inodo struct {
    I_uid   int64
    I_gid   int64
    I_size  int64
```

```
    l_atime [16]byte
    l_ctime [16]byte
    l_mtime [16]byte
    l_block [15]int64
    l_type  byte
    l_perm  int64
}
```

- **l_uid**: ID del usuario propietario
- **l_gid**: ID del grupo propietario
- **l_size**: Tamaño del archivo/directorio
- **l_atime**: Fecha de último acceso
- **l_ctime**: Fecha de creación
- **l_mtime**: Fecha de última modificación
- **l_block**: Array de punteros a bloques (12 directos, 1 indirecto, 1 doble indirecto, 1 triple indirecto)
- **l_type**: Tipo (0=archivo, 1=directorio)
- **l_perm**: Permisos (en formato octal)

Bloque de Carpeta

Estructura para almacenar contenido de directorios:

go

Copy

Download

```
type BloqueCarpeta struct {
    B_content [4]Content
}
```

Donde Content es:

go

Copy

Download

```
type Content struct {
    B_name  [12]byte
    B_inodo int64
}
```

Bloque de Archivo

Estructura para almacenar contenido de archivos:

go

Copy

Download

```
type BloqueArchivo struct {
    B_content [64]byte
}
```

Superbloque

Estructura que contiene metadatos del sistema de archivos:

go

Copy

Download

```
type SuperBloque struct {
    S_filesystem_type   int64
    S_inodes_count      int64
    S_blocks_count      int64
    S_free_blocks_count int64
    S_free_inodes_count int64
    S_mtime             [16]byte
    S_umtime            [16]byte
    S_mnt_count         int64
    S_magic             int64
    S_inode_size        int64
```

S_block_size int64

S_first_ino int64

S_first_blo int64

S_bm_inode_start int64

S_bm_block_start int64

S_inode_start int64

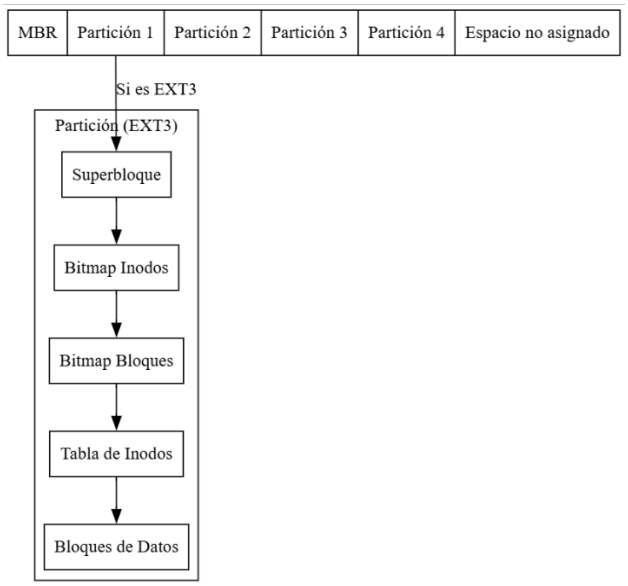
S_block_start int64

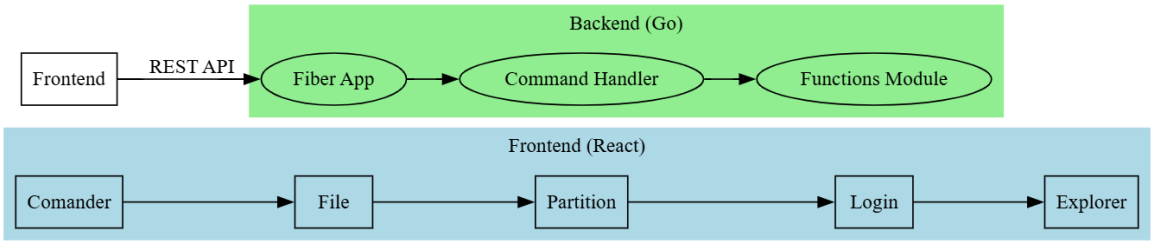
}

Organización del archivo .dsk

El archivo binario .dsk sigue la siguiente estructura:

1. **MBR**: Primeros 128 bytes del archivo
2. **Particiones**: Según lo definido en el MBR
3. **Sistemas de archivos**: Dentro de las particiones que los contengan
    - Superbloque
    - Bitmap de inodos
    - Bitmap de bloques
    - Tabla de inodos
    - Bloques de datos




Backend (Go)
Frontend
REST API
Fiber App
Command Handler
Functions Module
Frontend (React)
Comander
File
Partition
Login
Explorer