Manual de Técnico

Proyecto 1 Fase 2

Manejo e implementación de archivos
Sección P
Derek Francisco Orellana Ibáñez
202001151

Simulación del Sistema de Archivos EXT3 en la Aplicación Web

La aplicación web incluye una simulación funcional de un sistema de archivos basado en el formato EXT3, con soporte parcial para EXT2. Esta simulación fue desarrollada con el propósito de entender, emular y gestionar operaciones de bajo nivel sobre estructuras de almacenamiento similares a las que se encuentran en sistemas reales Linux.

El backend, desarrollado en Go (Golang), proporciona una API y lógica interna que interpreta comandos de manipulación de discos, estructuras de archivos y usuarios, simulando el comportamiento real de EXT3.

Creación y Manejo de Discos Virtuales

El sistema permite la creación de archivos de disco virtual con extensión .DSK, sobre los cuales se ejecutan múltiples operaciones, tales como:

- mkdisk: crea un archivo de disco con tamaño definido.
- fdisk: crea y administra particiones dentro del archivo .DSK.
- rmdisk: elimina el archivo de disco virtual.

Estas operaciones simulan la interacción con dispositivos de almacenamiento físico, generando estructuras en el archivo .DSK.

Montaje y Desmontaje

Las particiones simuladas pueden montarse y desmontarse mediante los comandos:

- mount: registra una partición activa, asignándole una identificación única.
- unmount: elimina el registro de una partición activa.

Esto permite trabajar con múltiples particiones de forma concurrente, similar a un sistema operativo real.

Estructura Interna del Sistema de Archivos EXT3

El sistema EXT3 simulado está compuesto por las siguientes estructuras clave:

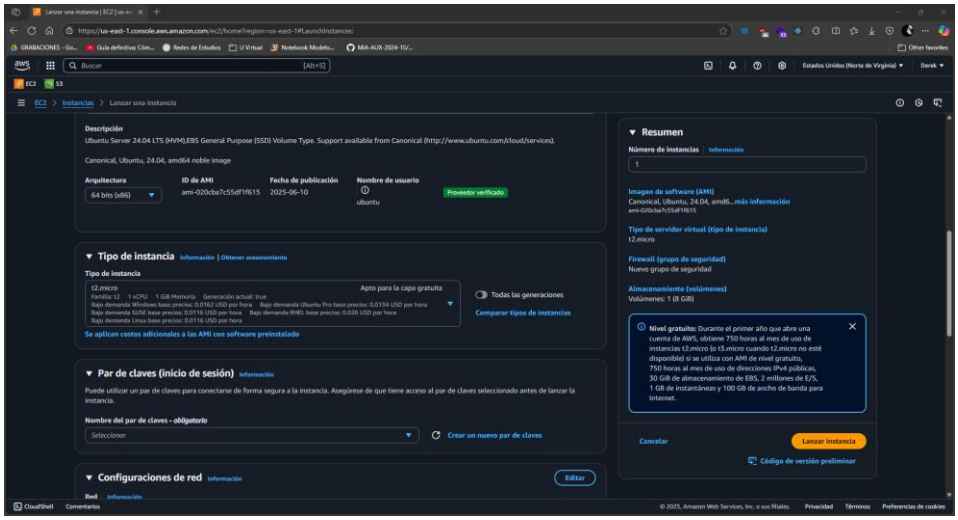
- Superbloque: contiene la metadata principal del sistema de archivos, incluyendo el conteo de inodos, bloques, bitmap, etc.
- Inodos: representan archivos o carpetas, almacenando su información de control (permisos, apuntadores a bloques, etc.).
- Bloques de datos: contienen la información real de los archivos o directorios.
- Directorios: organizan jerárquicamente los archivos, almacenando referencias a inodos.
- Journal (bitácora): utilizado únicamente en EXT3, permite registrar transacciones antes de aplicarlas, garantizando consistencia ante fallos.

Despliegue a AWS

Creación de la instancia EC2

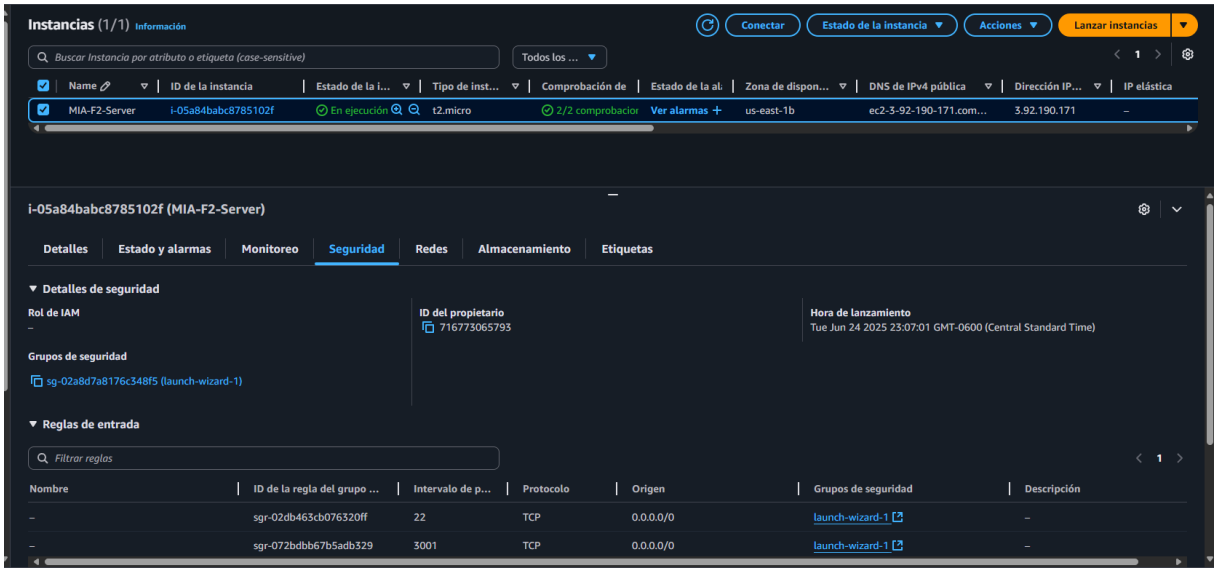
1. Acceder a la consola de AWS y navegar al servicio EC2.
2. Crear una nueva instancia con el sistema operativo Ubuntu Server.
3. Durante la creación de la instancia:
4. Se generó un par de claves (key pair) para el acceso seguro por SSH.

Se configuró el grupo de seguridad con las reglas necesarias (puertos abiertos como el 22 para SSH y el 3001).



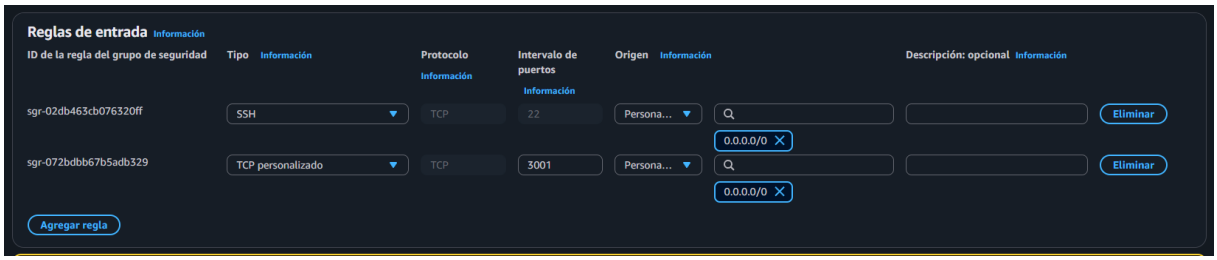
configuración de seguridad

1. Escoger la instancia
2. Ir a la pestaña de seguridad
3. Editar reglas de entrada



En editar reglas de entrada escogeremos las siguientes opciones:

- Tcp personalizada
- Puerto 3001 (o el puerto que utiliza el proyecto backend)
- Origen 0.0.0.0/0 para aceptar cualquier ip entrante



Guardamos los reglas de entrada

Conexión a la instancia EC2 usando Termius

1. Abrir Termius.
2. Agregar una nueva conexión SSH con los siguientes datos:
   - IP pública de la instancia EC2.
   - Usuario: ubuntu
   - Archivo .pem generado previamente.

Preparación del entorno en la instancia EC2

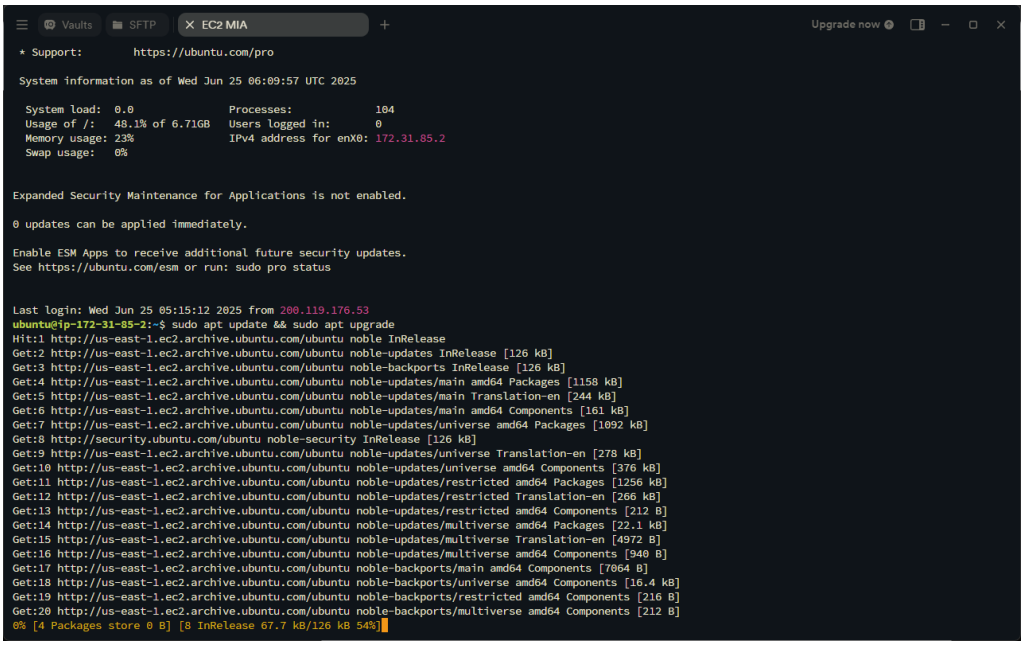
Una vez conectado, se ejecutaron los siguientes comandos para actualizar los paquetes:

```
sudo apt update

sudo apt upgrade
```

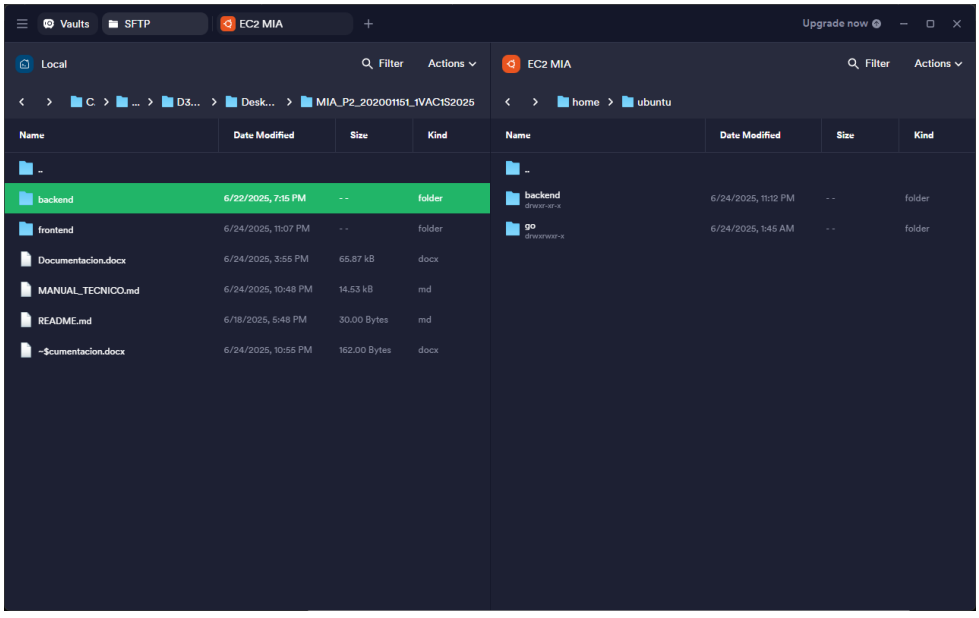
Luego, se instaló Golang con el siguiente comando (dependiendo de la versión que se desee):

```
sudo apt install golang-go
```



Transferencia del proyecto Backend

Utilizando la funcionalidad SFTP de Termius, se transfirió la carpeta Backend desde el equipo local hacia la instancia EC2, ubicándola en el directorio del usuario.



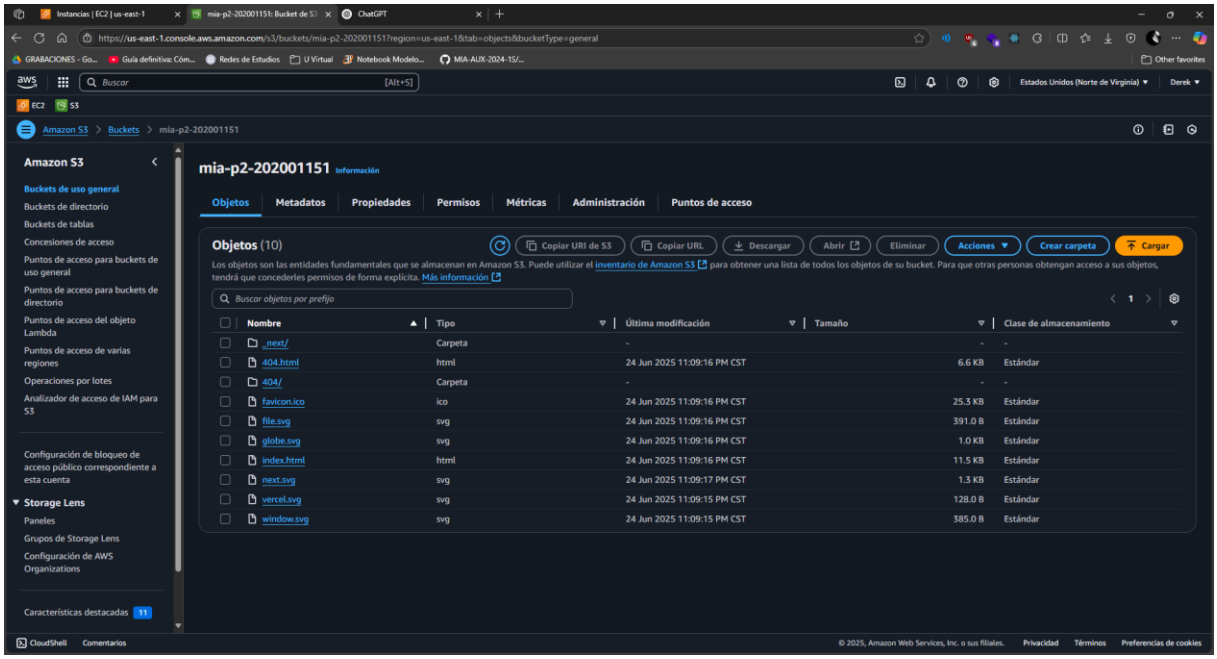
Creación y configuración del bucket S3 para el Frontend

1. En la consola de AWS, ir a Amazon S3 y crear un nuevo bucket.
2. Configurar el bucket con las siguientes propiedades:
    - Habilitar el hosting de sitio web estático.
    - Definir el documento de índice (ej. index.html) y el documento de error (ej. error.html).
3. Configurar la política de acceso del bucket para permitir el acceso público (si aplica):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::NOMBRE_DEL_BUCKET/*"
    }
  ]
}
```

Carga del Frontend

1. Subir todos los archivos estáticos del frontend (HTML, CSS, JS, imágenes) al bucket S3.
2. Verificar que el sitio web esté funcionando accediendo al endpoint del sitio estático generado por S3.



Descripción de la Arquitectura del Sistema

Componentes Principales del Sistema

El sistema se divide en dos componentes fundamentales:

- Frontend (cliente): es una aplicación web estática desarrollada con tecnologías como HTML, CSS y JavaScript. Es responsable de proporcionar una interfaz gráfica para que el usuario interactúe con el sistema de archivos simulado.
- Backend (servidor): es una API desarrollada en Go (Golang), encargada de procesar los comandos, gestionar las estructuras del sistema EXT3 simulado y responder con los resultados. Incluye la lógica para la creación de discos, montaje de particiones, manejo de archivos, autenticación de usuarios, y generación de reportes.

Flujo de Comunicación entre Componentes

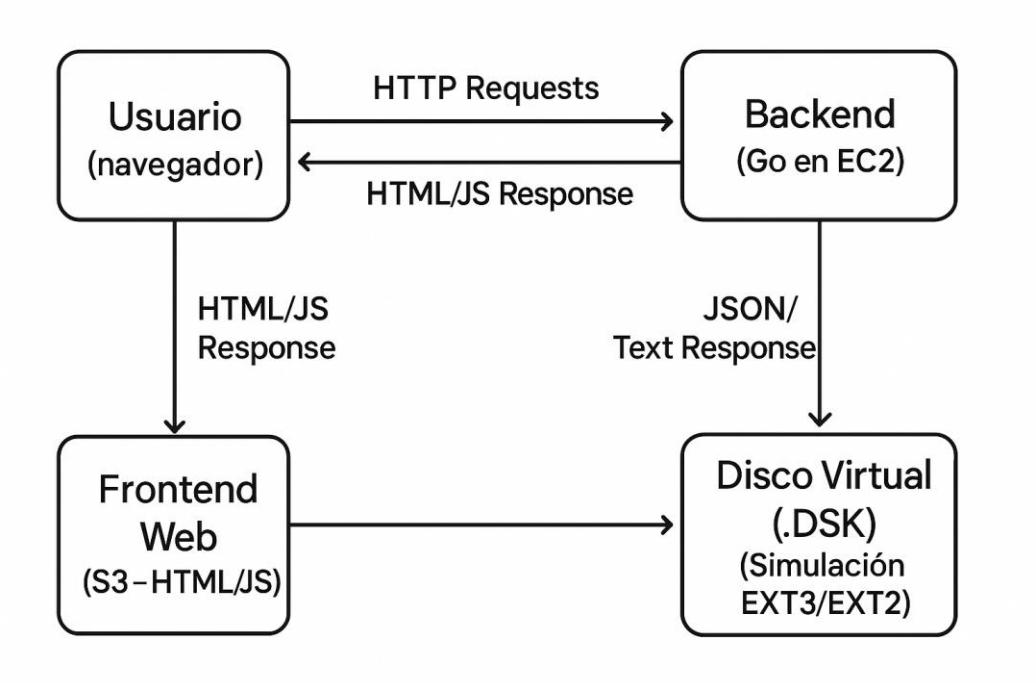
El usuario accede a la interfaz web almacenada en un bucket S3 de AWS configurado como sitio web estático.

- La interfaz permite ingresar comandos o acciones (ej. mkdisk, mkdir, cat).
- Estas acciones son enviadas mediante solicitudes HTTP (normalmente POST) al backend.
- El backend procesa los comandos, interactúa con los archivos .DSK, y devuelve una respuesta al frontend.
- El frontend interpreta la respuesta y actualiza la interfaz para el usuario.

Arquitectura de Despliegue en AWS

Para hacer disponible la aplicación en la nube, se utilizó la siguiente arquitectura:

- Backend
- Servicio: AWS EC2
- Configuración:
    - Instancia de Ubuntu Server.
    - Backend en Go alojado y ejecutado en segundo plano.
    - Conexión SSH mediante par de claves y gestión con Termius.
    - Comunicación habilitada por el puerto correspondiente (3001).
- Frontend
- Servicio: AWS S3
- Configuración:
    - Bucket con alojamiento de sitio web estático.
    - Archivos HTML, JS y CSS del frontend cargados.
    - Políticas públicas configuradas para acceso sin autenticación.
- Conexión
- El frontend (en S3) hace solicitudes HTTP al backend alojado en la instancia EC2, utilizando la IP pública.



Endpoints Backend

- /api/status: Verifica el estado del servidor y la sesión del usuario (GET).
- /api/execute: Ejecuta comandos en el servidor (POST).
- /api/drives: Obtiene información de los discos y particiones (GET).
- /api/drives/info: Obtiene información detallada de los discos (GET).
- /api/drives/{driveletter}: Obtiene información de un disco específico (GET).
- /api/drives/{driveletter}/partitions: Obtiene las particiones de un disco (GET).
- /api/login: Maneja el inicio de sesión (POST).
- /api/logout: Maneja el cierre de sesión (POST).
- /api/find: Busca archivos en el servidor (POST).
- /api/cat: Muestra el contenido de un archivo (POST).

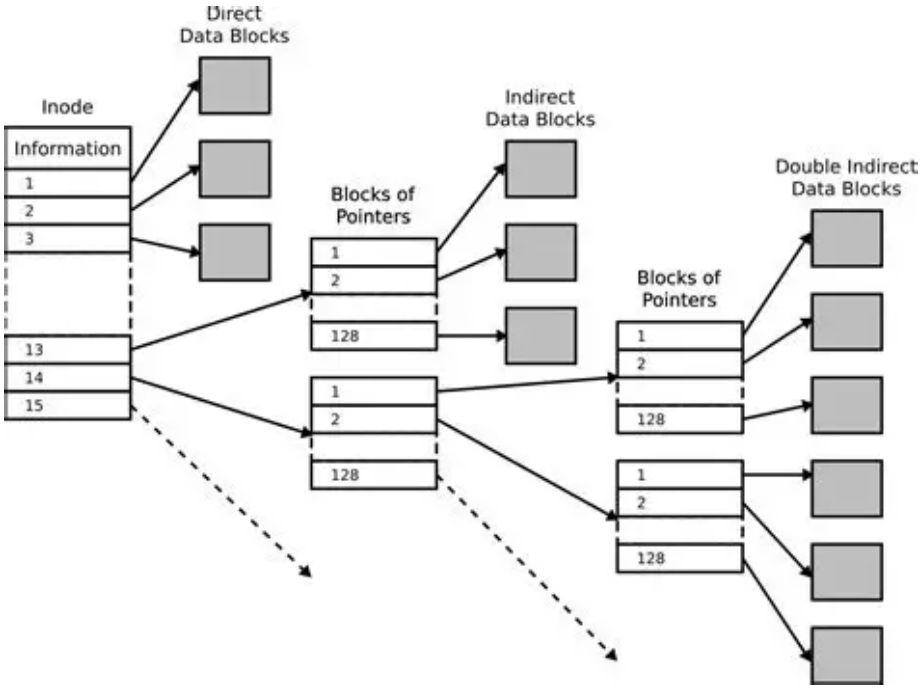
Estructuras de Datos en el Sistema de Archivos Simulado (EXT2/EXT3)

MBR (Master Boot Record)

El Master Boot Record (MBR) es la primera estructura almacenada al inicio del archivo binario .DSK. En esta simulación, tiene un tamaño fijo de 159 bytes y cumple la función de definir la organización general del disco.

Inodos

Los inodos son estructuras de control que almacenan metadatos sobre los archivos y directorios del sistema. Cada archivo o directorio está asociado a un único inodo.



Bloques de Datos

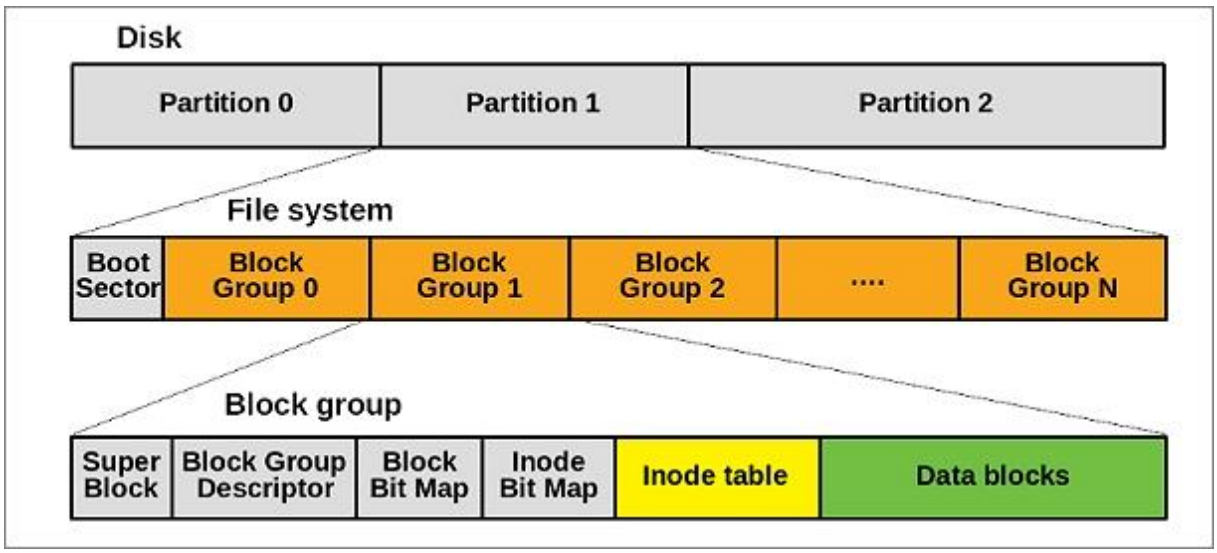
Los bloques de datos contienen el contenido real de los archivos y directorios.

Gestión de Espacio y Archivos

Para simular adecuadamente un sistema EXT2/EXT3:

- Se utiliza un bitmap de bloques y otro de inodos libres.
- El sistema implementa funciones para:
  - Buscar bloques disponibles
  - Asignar bloques contiguos o dispersos
  - Actualizar los punteros del inodo

En el caso de EXT3, se añade una estructura adicional de **journaling**


Disk
Partition 0
Partition 1
Partition 2
File system
Boot Sector
Block Group 0
Block Group 1
Block Group 2
....
Block Group N
Block group
Super Block
Block Group Descriptor
Block Bit Map
Inode Bit Map
Inode table
Data blocks