**1. Descripción del proceso de desarrollo del script**

El script fue desarrollado para la gestión de usuarios y sus respectivos roles (admin o user).   
Su propósito es crear estructuras de datos simples que representen usuarios y grupos, asignar permisos sobre un directorio llamado "./srv/trueque",   
y llevar un registro de todas las acciones mediante un archivo de log.

**2. Elección de comandos y estructura lógica**

Se utilizaron comandos esenciales de Bash como mkdir, echo, chmod, y estructuras condicionales para validar entradas y controlar el flujo.   
La lógica general incluye: validación de argumentos, creación de usuario y grupo, asignación de permisos y registro de operaciones.

**3. Justificación de decisiones tomadas durante el desarrollo**

Se eligió Bash por su disponibilidad y facilidad para gestionar archivos en sistemas Unix.   
Se optó por no usar usuarios reales del sistema para evitar requerir permisos administrativos.

**1. Declaración de directorios y archivos principales**

USERS\_DIR="users\_db"  
GROUPS\_DIR="groups\_db"  
PERMISSIONS\_FILE="srv/trueque/.permiso"  
LOG\_FILE="gestion\_usuarios-log"  
  
- USERS\_DIR: carpeta donde se almacenarán los archivos individuales de usuarios.  
- GROUPS\_DIR: carpeta para archivos que representen grupos.  
- PERMISSIONS\_FILE: archivo donde se registran los permisos asociados a cada grupo.  
- LOG\_FILE: archivo donde se guardan las acciones realizadas con marcas de tiempo.

**2. Crear directorios si no existen**

mkdir -p "$USERS\_DIR" "$GROUPS\_DIR" "srv/trueque"  
  
- mkdir -p crea los directorios necesarios, si no existen aún. La opción -p evita errores si el directorio ya existe.

**3. Obtener argumentos desde la línea de comandos**

USERNAME=$1  
GROUPNAME=$2  
  
- Toma el primer argumento como nombre de usuario y el segundo como grupo al que pertenecerá.

**4. Verificar argumentos obligatorios**

if [[ -z "$USERNAME" || -z "$GROUPNAME" ]]; then  
 echo "Uso: $0 <usuario> <grupo>"  
 exit 1  
fi  
  
- Si alguno de los argumentos está vacío, se muestra un mensaje de uso correcto y se detiene la ejecución con exit 1.

**5. Definir rutas completas para archivos**

USER\_FILE="$USERS\_DIR/$USERNAME.txt"  
GROUP\_FILE="$GROUPS\_DIR/$GROUPNAME.txt"  
  
- Se construyen los paths completos de los archivos que representan al usuario y al grupo.

**6. Obtener la fecha y hora actuales**

DATE=$(date +"%Y-%m-%d %H:%M:%S")  
  
- Guarda la fecha y hora actuales en el formato deseado para agregar al log.

**7. Crear archivo de usuario si no existe**

if [[ ! -f "$USER\_FILE" ]]; then  
 echo "Usuario: $USERNAME" > "$USER\_FILE"  
 echo "Grupo: $GROUPNAME" >> "$USER\_FILE"  
 echo "$DATE - Usuario $USERNAME creado en grupo $GROUPNAME" >> "$LOG\_FILE"  
else  
 echo "$DATE - Usuario $USERNAME ya existe" >> "$LOG\_FILE"  
fi  
  
- Crea un archivo que contiene la información del usuario y registra en el log.  
- Si ya existe, solo registra que el usuario ya está creado.

**8. Crear archivo de grupo si no existe**

if [[ ! -f "$GROUP\_FILE" ]]; then  
 echo "Miembros del grupo $GROUPNAME:" > "$GROUP\_FILE"  
 echo "$DATE - Grupo $GROUPNAME creado" >> "$LOG\_FILE"  
fi  
  
- Similar al anterior, pero para grupos: crea un archivo donde se listarán sus miembros.

**9. Agregar usuario al grupo si aún no está listado**

if ! grep -q "$USERNAME" "$GROUP\_FILE"; then  
 echo "$USERNAME" >> "$GROUP\_FILE"  
 echo "$DATE - Usuario $USERNAME agregado al grupo $GROUPNAME" >> "$LOG\_FILE"  
else  
 echo "$DATE - Usuario $USERNAME ya está en el grupo $GROUPNAME" >> "$LOG\_FILE"  
fi  
  
- Usa grep -q para verificar si el usuario ya está en la lista del grupo.  
- Si no está, lo agrega al archivo y lo registra en el log.

**10. Crear o actualizar archivo de permisos**

if [[ ! -f "$PERMISSIONS\_FILE" ]]; then  
 echo "$GROUPNAME:rw" > "$PERMISSIONS\_FILE"  
 echo "$DATE - Permisos para $GROUPNAME creados" >> "$LOG\_FILE"  
elif ! grep -q "^$GROUPNAME:" "$PERMISSIONS\_FILE"; then  
 echo "$GROUPNAME:rw" >> "$PERMISSIONS\_FILE"  
 echo "$DATE - Permisos para $GROUPNAME agregados" >> "$LOG\_FILE"  
fi  
  
- Si el archivo .permiso no existe, lo crea y asigna permisos rw al grupo.  
- Si ya existe, pero el grupo aún no figura, lo agrega también.  
- rw indica permiso de lectura y escritura.

**Video de la ejecución del script**

Video en GitHub de como funciona y tambien documento de texto con el codigo