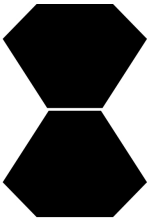


S.I.G.I.E
Chronos
Sistemas Operativos

Rol	Apellido	Nombre	Cédula Identidad	Email	Tel./Cel.
Coordinador	Chapuis	Juan	5.630.283-4	jchapuis@scuolaitaliana.edu.uy	098844043
Sub-Coordinador	Roizen	Agustin	6.339.592-9	aroizen@scuolaitaliana.edu.uy	097259510
Integrante	Marino	Bruno	5.707.620-6	bmarino@scuolaitaliana.edu.uy	092617596

Docente: Farias, Gustavo

Fecha de entrega: 03/11/2025

Tercera Entrega

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS

PRIMERA ENTREGA

a) Estudio de los diferentes roles de los usuarios del sistema

ROLES DEFINIDOS:

1. **Administrador de Sistemas (admin_sis)**
    - Responsable: Gestión general del sistema y Docker
    - Permisos: Usuarios, servicios, contenedores, backups
    - Grupos: admin_sis, docker, sudo
2. **Administrador de Seguridad (admin_sec)**
    - Responsable: Seguridad, firewall, SSH, auditoría
    - Permisos: Configuración seguridad, monitoreo, logs
    - Grupos: admin_sec, sudo
3. **Operador (operador)**
    - Responsable: Operaciones diarias del sistema
    - Permisos: Backups básicos, monitoreo, tareas programadas
    - Grupos: operadores
4. **Desarrollador (dev_user)**
    - Responsable: Desarrollo de aplicaciones en Docker
    - Permisos: Acceso a contenedores, gestión de aplicaciones
    - Grupos: desarrolladores, docker
5. **Usuario Docker (docker_user)**
    - Responsable: Gestión específica de contenedores
    - Permisos: Administración Docker limitada
    - Grupos: docker

b) Diagrama de navegabilidad e implementación de shell script para administración de usuarios y grupos

DIAGRAMA DE NAVEGABILIDAD:

SISTEMA DE ADMINISTRACIÓN S.I.G.I.E

```
│
├── GESTIÓN DE USUARIOS
│   ├── Crear usuario
│   ├── Eliminar usuario
│   ├── Modificar usuario
│   ├── Listar usuarios
│   ├── Crear grupo
│   └── Agregar usuario a grupo
│
├── GESTIÓN DOCKER
│   ├── Estado contenedores
│   ├── Logs contenedores
│   ├── Reiniciar servicios
│   ├── Gestión imágenes
│   └── Actualizar aplicaciones
│
├── CONFIGURACIÓN RED
│   ├── Ver interfaces
│   ├── Configurar IP estática
│   └── Test conectividad
│
├── GESTIÓN BACKUPS
│   ├── Backup manual
│   ├── Ver backups existentes
│   └── Estado espacio disco
│
├── MONITOR SISTEMA
│   ├── Estado servicios
│   ├── Uso de recursos
│   ├── Logs del sistema
│   └── Conexiones de red
│
└── GESTIÓN SEGURIDAD
    ├── Auditoría de seguridad
    ├── Configurar SSH seguro
    ├── Configurar firewall
    ├── Verificar integridad scripts
    └── Configurar Fail2ban
```

SCRIPT DE ADMINISTRACIÓN DE USUARIOS: `user_management.sh`

```
#!/bin/bash

function gestion_usuarios() {
    while true; do
        clear
        echo "============================================="
        echo "          GESTION DE USUARIOS Y GRUPOS"
        echo "============================================="
        echo "1. Crear usuario"
        echo "2. Eliminar usuario"
        echo "3. Modificar usuario"
        echo "4. Listar usuarios"
        echo "5. Crear grupo"
        echo "6. Agregar usuario a grupo"
        echo "7. Volver al menu principal"
        echo "============================================="

        read -p "Seleccione opcion: " opcion

        case $opcion in
            1)
                read -p "Nombre de usuario: " username
                read -p "Nombre completo: " fullname
                sudo useradd -m -c "$fullname" -s /bin/bash "$username"
                echo "Establecer contraseña para $username:"
                sudo passwd "$username"
                ;;
            2)
                read -p "Usuario a eliminar: " username
                sudo userdel -r "$username"
                echo "Usuario $username eliminado"
                ;;
            3)
                read -p "Usuario a modificar: " username
                read -p "Nuevo comentario: " comment
                sudo usermod -c "$comment" "$username"
                ;;
            4)
                echo "=== LISTA DE USUARIOS ==="
                echo "Usuarios del sistema:"
                cut -d: -f1 /etc/passwd | sort
                ;;
```

```
            5)
                read -p "Nombre del grupo: " groupname
                sudo groupadd "$groupname"
                echo "Grupo $groupname creado"
                ;;
            6)
                read -p "Usuario: " username
                read -p "Grupo: " groupname
                sudo usermod -aG "$groupname" "$username"
                echo "Usuario $username agregado al grupo $groupname"
                ;;
            7)
                break
                ;;
            *)
                echo "Opcion invalida"
                ;;
        esac
        read -p "Presione Enter para continuar..."
    done
}

gestion_usuarios
```

c) Relevamiento y justificación del Sistema Operativo a utilizar

SERVIDOR: Ubuntu Server 22.04 LTS

JUSTIFICACIÓN TÉCNICA:

- **Soporte LTS:** 5 años de soporte garantizado hasta 2027
- **Compatibilidad Docker:** Soporte nativo y optimizado para contenedores
- **Estabilidad:** Versión probada en entornos empresariales y de producción
- **Seguridad:** Actualizaciones de seguridad regulares y soporte extendido
- **Rendimiento:** Optimizado para servidores y virtualización
- **Comunidad:** Amplia documentación y soporte comunitario
- **Ecosistema:** Amplio repositorio de paquetes y herramientas

TERMINALES: Ubuntu Desktop 22.04 LTS

JUSTIFICACIÓN:

- **Uniformidad:** Mismo ecosistema que el servidor para consistencia
- **Compatibilidad Docker:** Mismas versiones y herramientas de contenerización
- **Productividad:** Interfaz gráfica para desarrollo y administración

- **Soporte:** Mismo ciclo de vida LTS que el servidor
- **Seguridad:** Políticas de seguridad homogéneas en toda la infraestructura

d) Manual de instalación del Sistema Operativo en el servidor seleccionado

MANUAL DE INSTALACIÓN UBUNTU SERVER 22.04 LTS

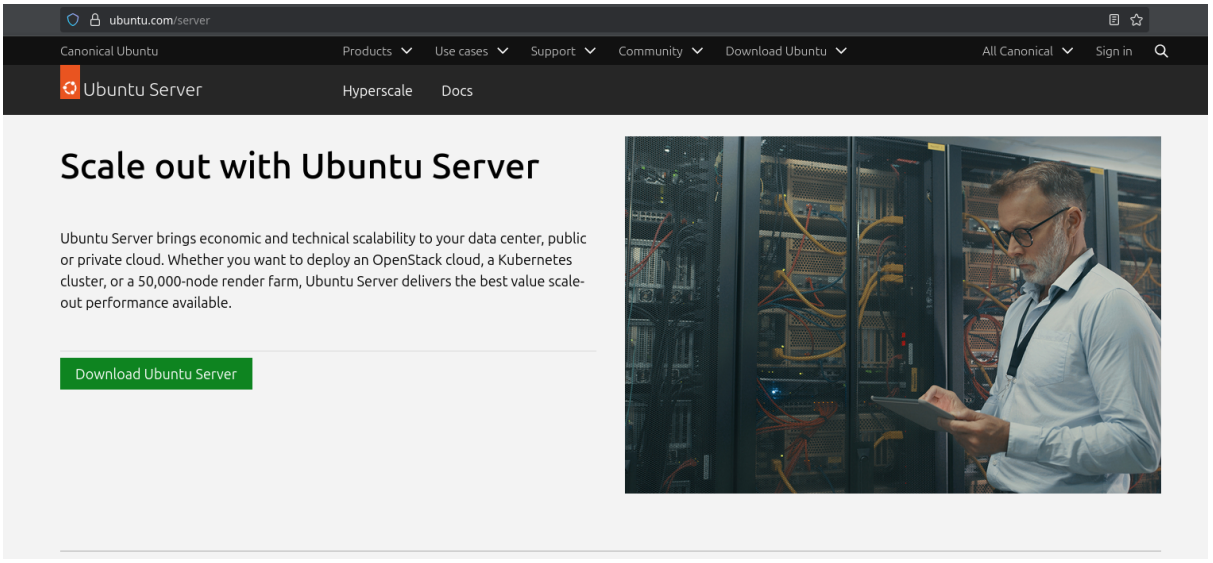
REQUISITOS MÍNIMOS DEL SISTEMA:

- Procesador: 2 cores x86_64 (4 cores recomendado para Docker)
- Memoria RAM: 4 GB (8 GB recomendado para múltiples contenedores)
- Almacenamiento: 40 GB SSD (para sistema, contenedores y backups)
- Red: Conexión Ethernet Gigabit
- Acceso: Conexión a Internet para actualizaciones

PROCEDIMIENTO DETALLADO DE INSTALACIÓN:

PASO 1: PREPARACIÓN

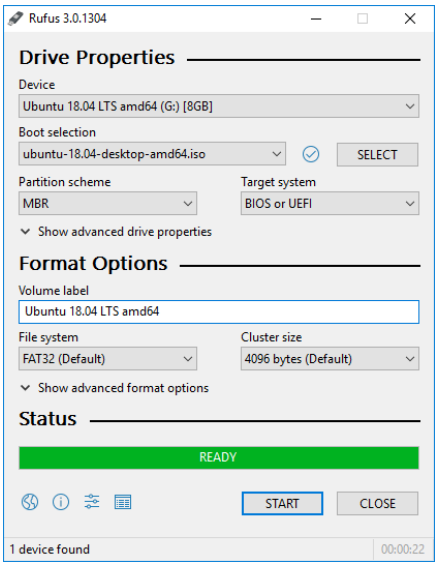
Descargar imagen ISO desde sitio oficial
wget https://releases.ubuntu.com/22.04/ubuntu-22.04.3-live-server-amd64.iso



Verificar integridad del ISO
echo "a4acf3381055baef6877a5e6c4f5ca48b16b8a9318787413b8c14b0b36370c0d ubuntu-22.04.3-live-server-amd64.iso" | sha256sum -c

# Crear USB booteable (Linux)
sudo dd if=ubuntu-22.04.3-live-server-amd64.iso of=/dev/sdX bs=4M status=progress && sync

O en Windows usar Rufus con configuración GPT para UEFI



PASO 2: INSTALACIÓN BÁSICA

1. Arrancar desde el medio de instalación
2. Seleccionar idioma: Español
3. Configurar teclado: Español latinoamericano
4. Tipo de instalación: Ubuntu Server normal
5. Configuración de red:
    - Nombre del servidor: sigie-server
    - Usuario: admin
    - Contraseña: Temporal123! (cambiar inmediatamente después)

PASO 3: CONFIGURACIÓN DE ALMACENAMIENTO

bash
Esquema de particiones recomendado con LVM:
- /boot: 1 GB (partición primaria)
- swap: 4 GB (partición lógica LVM)
- /: 35 GB (partición lógica LVM - extensible)

O esquema simple:
- /: 40 GB (partición única)

PASO 4: SELECCIÓN DE PAQUETES

- Instalar OpenSSH server (habilitado)
- No instalar servicios adicionales (solo base)
- Profile: Servidor básico

PASO 5: CONFIGURACIÓN POST-INSTALACIÓN

bash
# Actualizar sistema
sudo apt update && sudo apt upgrade -y

# Instalar Docker y herramientas esenciales
sudo apt install -y docker.io docker-compose net-tools curl wget git

# Configurar Docker para inicio automático
sudo systemctl enable docker
sudo systemctl start docker

# Agregar usuario admin al grupo docker
sudo usermod -aG docker admin

# Configurar zona horaria
sudo timedatectl set-timezone America/Argentina/Buenos_Aires

e) Implementación de contenedor para contener el script de administración de usuarios

CONTENEDOR DOCKER PARA SCRIPTS DE ADMINISTRACIÓN

Dockerfile:

FROM ubuntu:22.04

LABEL maintainer="S.I.G.I.E Team"
LABEL version="1.0"
LABEL description="Contenedor para scripts de administración S.I.G.I.E"

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y \
    sudo \
    openssh-client \
    docker.io \
    docker-compose \
    net-tools \
    ufw \
    curl \
    wget \
    git \
    python3 \
    python3-pip \
    && rm -rf /var/lib/apt/lists/*

```
RUN groupadd -r admin && useradd -r -g admin admin

WORKDIR /usr/local/admin-scripts

COPY *.sh ./

RUN chmod +x *.sh && \
    chown -R admin:admin .

USER admin

VOLUME ["/var/run/docker.sock", "/etc/ssh", "/backups", "/var/log"]

CMD ["./menu_unificado.sh"]
```

docker-compose.yml:

```
yaml
version: '3.8'

services:
  admin-scripts:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: sigie-admin-container
    hostname: sigie-admin
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - /etc/shadow:/etc/shadow:ro
      - /etc/ssh:/etc/ssh:ro
      - /backups:/backups
      - /var/log:/var/log:ro
      - /usr/local/admin-scripts:/usr/local/admin-scripts:ro
    network_mode: host
    privileged: true
    restart: unless-stopped
    environment:
      - ENVIRONMENT=production
      - LOG_LEVEL=INFO
```

```
  chronos-app:
    image: chronos:latest
    container_name: chronos-app
    restart: unless-stopped
    ports:
      - "3000:3000"
    volumes:
      - /docker/chronos/data:/app/data
    environment:
      - NODE_ENV=production
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:3000/health"]
      interval: 30s
      timeout: 10s
      retries: 3

networks:
  default:
    name: sigie-network
```

SEGUNDA ENTREGA

a) Usuarios del sistema creados de acuerdo al estudio de roles

SCRIPT DE CREACIÓN DE USUARIOS: `crear_usuarios.sh`

```
#!/bin/bash

echo "============================================="
echo "      CREACION DE USUARIOS S.I.G.I.E"
echo "============================================="

declare -A usuarios=(
```

```
    ["admin_sis"]="Administrador de Sistemas"
    ["admin_sec"]="Administrador de Seguridad"
    ["operador"]="Operador del Sistema"
    ["dev_user"]="Usuario Desarrollo"
    ["docker_user"]="Usuario Docker"
)

echo "Creando grupos del sistema..."
sudo groupadd -f admin_sis
sudo groupadd -f admin_sec
sudo groupadd -f operadores
sudo groupadd -f desarrolladores
sudo groupadd -f docker

for usuario in "${!usuarios[@]}"; do
    if ! id "$usuario" &>/dev/null; then
        echo "Creando usuario: $usuario"
        sudo useradd -m -s /bin/bash -c "${usuarios[$usuario]}" "$usuario"

        case $usuario in
            admin_sis)
                sudo usermod -aG admin_sis,sudo,docker "$usuario"
                echo "Grupos: admin_sis, sudo, docker"
                ;;
            admin_sec)
                sudo usermod -aG admin_sec,sudo "$usuario"
                echo "Grupos: admin_sec, sudo"
                ;;
            operador)
                sudo usermod -aG operadores "$usuario"
                echo "Grupos: operadores"
                ;;
            dev_user)
                sudo usermod -aG desarrolladores,docker "$usuario"
                echo "Grupos: desarrolladores, docker"
                ;;
            docker_user)
                sudo usermod -aG docker "$usuario"
                echo "Grupos: docker"
                ;;
        esac

        echo "Estableciendo contraseña temporal para $usuario..."
        echo "$usuario:Temp123!" | sudo chpasswd
```

```
        echo "Usuario $usuario creado exitosamente"
        echo "-----------------------------------------------"
    else
        echo "Usuario $usuario ya existe"
    fi
done

echo "Configurando permisos sudo..."
sudo tee /etc/sudoers.d/admin_roles > /dev/null << 'EOF'
%admin_sis ALL=(ALL) /usr/bin/systemctl, /usr/bin/docker,
/usr/local/bin/docker-compose, /usr/local/admin-scripts/
%admin_sec ALL=(ALL) /usr/sbin/ufw, /usr/bin/passwd, /usr/bin/systemctl restart
ssh, /usr/bin/fail2ban-client
%operadores ALL=(ALL) /usr/local/admin-scripts/menu_usuario.sh, /usr/bin/passwd,
/usr/local/admin-scripts/backup_sistema.sh
%docker ALL=(ALL) /usr/bin/docker, /usr/local/bin/docker-compose
EOF

sudo chmod 440 /etc/sudoers.d/admin_roles

echo "================================================"
echo "      USUARIOS CREADOS EXITOSAMENTE"
echo "================================================"
echo "admin_sis    - Administrador de Sistemas (sudo, docker)"
echo "admin_sec    - Administrador de Seguridad (sudo)"
echo "operador     - Operador del Sistema"
echo "dev_user     - Usuario Desarrollo (docker)"
echo "docker_user  - Usuario Docker"
echo ""
echo "Contraseñas temporales: Temp123!"
echo "Cambiar en primer login con: passwd"
```

b) Menús para los usuarios del Sistema, 1ra. Versión navegable y aplicable

SCRIPT MENÚ USUARIO: `menu_usuario.sh`

```
#!/bin/bash

function menu_principal() {
    while true; do
        clear
        echo "================================================"
        echo "          MENU USUARIO - S.I.G.I.E"
        echo "================================================"
```

```
    echo "Usuario actual: $(whoami)"
    echo "Servidor: $(hostname)"
    echo "============================================="
    echo "1. Cambiar contraseña"
    echo "2. Ver espacio en disco"
    echo "3. Ver procesos del usuario"
    echo "4. Ver informacion del sistema"
    echo "5. Ver contenedores activos"
    echo "6. Salir"
    echo "============================================="

    read -p "Opcion: " opcion

    case $opcion in
      1)
        passwd
        ;;
      2)
        echo "=== ESPACIO EN DISCO ==="
        df -h /home
        ;;
      3)
        echo "=== PROCESOS DE $(whoami) ==="
        ps -u $(whoami) -o pid,ppid,cmd,%mem,%cpu --sort=-%cpu | head -10
        ;;
      4)
        echo "=== INFORMACION DEL SISTEMA ==="
        echo "Hostname: $(hostname)"
        echo "Sistema: $(lsb_release -d | cut -f2)"
        echo "Kernel: $(uname -r)"
        echo "Uptime: $(uptime -p)"
        echo "Docker: $(docker --version 2>/dev/null || echo 'No disponible')"
        ;;
      5)
        echo "=== CONTENEDORES ACTIVOS ==="
        docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}" 2>/dev/null ||
echo "Docker no disponible"
        ;;
      6)
        echo "Saliendo del sistema..."
        exit 0
        ;;
      *)
        echo "Opcion invalida"
```

```
            ;;
        esac
        read -p "Presione Enter para continuar..."
    done
}

if [[ $- == *i* ]]; then
    menu_principal
else
    echo "Ejecutar en modo interactivo: bash menu_usuario.sh"
fi
```

c) Menú para el Operador del Centro de Cómputos (Administrador del Sistema), 1ra.Versión

SCRIPT MENÚ ADMINISTRADOR: `menu_admin.sh`

```
#!/bin/bash

function mostrar_menu_admin() {
    clear
    echo "============================================="
    echo "       MENU ADMINISTRADOR - S.I.G.I.E"
    echo "============================================="
    echo "1. Gestion de usuarios"
    echo "2. Gestion Docker"
    echo "3. Monitor del sistema"
    echo "4. Configuracion de red"
    echo "5. Backup y restauracion"
    echo "6. Gestion de seguridad"
    echo "7. Salir"
    echo "============================================="
}

function gestion_usuarios() {
    /usr/local/admin-scripts/user_management.sh
}

function gestion_docker() {
    while true; do
        clear
        echo "=== GESTION DOCKER ==="
        echo "1. Estado contenedores"
        echo "2. Logs contenedor principal"
```

```
        echo "3. Reiniciar contenedores"
        echo "4. Estadisticas Docker"
        echo "5. Actualizar aplicacion"
        echo "6. Volver al menu principal"

        read -p "Seleccione opcion: " docker_op

        case $docker_op in
            1)
                docker ps -a
                ;;
            2)
                docker logs chronos-app --tail 20
                ;;
            3)
                cd /docker/chronos && docker-compose restart
                ;;
            4)
                docker stats --no-stream
                ;;
            5)
                /usr/local/admin-scripts/actualizar_web.sh
                ;;
            6)
                break
                ;;
            *)
                echo "Opcion invalida"
                ;;
        esac
        read -p "Presione Enter para continuar..."
    done
}

function monitor_sistema() {
    echo "=== MONITOR DEL SISTEMA ==="
    echo "1. Estado de servicios"
    echo "2. Uso de recursos"
    echo "3. Logs del sistema"
    echo "4. Espacio en disco"
    read -p "Seleccione opcion: " monitor_op

    case $monitor_op in
        1)
```

```
            echo "=== SERVICIOS ==="
            systemctl status docker ssh | grep -E "Active:"
            ;;
        2)
            echo "=== RECURSOS DEL SISTEMA ==="
            echo "CPU:"
            top -bn1 | grep "Cpu(s)"
            echo -e "\nMemoria:"
            free -h
            echo -e "\nDisco:"
            df -h / /home /backups /var/lib/docker
            ;;
        3)
            echo "=== ULTIMOS LOGS DEL SISTEMA ==="
            sudo tail -20 /var/log/syslog
            ;;
        4)
            echo "=== ESPACIO EN DISCO DETALLADO ==="
            df -h
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
}

function configuracion_red() {
    /usr/local/admin-scripts/config_red.sh
}

function backup_restauracion() {
    echo "=== BACKUP Y RESTAURACION ==="
    echo "1. Ejecutar backup manual"
    echo "2. Ver backups existentes"
    echo "3. Ver estado de espacio"
    echo "4. Ver logs de backup"
    read -p "Seleccione opcion: " backup_op

    case $backup_op in
        1)
            sudo /usr/local/admin-scripts/backup_sistema.sh
            ;;
        2)
            echo "=== BACKUPS EXISTENTES ==="
```

```
            find /backups -maxdepth 1 -type d -name "202*" 2>/dev/null | sort -r | head -5
            ;;
        3)
            df -h /backups
            ;;
        4)
            tail -20 /var/log/admin-system/backup.log 2>/dev/null || echo "No hay logs de
backup"
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
}

function gestion_seguridad() {
    echo "=== GESTION DE SEGURIDAD ==="
    echo "1. Ver estado firewall"
    echo "2. Ver logs de seguridad"
    echo "3. Ver intentos de acceso"
    echo "4. Verificar integridad scripts"
    read -p "Seleccione opcion: " security_op

    case $security_op in
        1)
            sudo ufw status verbose
            ;;
        2)
            sudo tail -20 /var/log/auth.log
            ;;
        3)
            sudo grep "Failed" /var/log/auth.log | tail -10
            ;;
        4)
            sudo /usr/local/admin-scripts/verify_script_integrity.sh --verify
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
}

while true; do
    mostrar_menu_admin
```

```
    read -p "Seleccione una opcion (1-7): " opcion

    case $opcion in
        1) gestion_usuarios ;;
        2) gestion_docker ;;
        3) monitor_sistema ;;
        4) configuracion_red ;;
        5) backup_restauracion ;;
        6) gestion_seguridad ;;
        7)
            echo "Saliendo del menu administrativo..."
            exit 0
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
    read -p "Presione Enter para continuar..."
done
```

d) Configuraciones de red en las terminales y el servidor

SCRIPT CONFIGURACIÓN RED: `config_red.sh`

```
#!/bin/bash

function mostrar_interfaces() {
    echo "=== INTERFACES DE RED ==="
    ip -o addr show | awk '{print $2, $4}'
    echo ""
    echo "=== TABLA DE RUTAS ==="
    ip route show
}

function config_ip_static() {
    interface=$(ip -o link show | awk -F: '$2 !~ "lo|vir|docker"{print $2; exit}' | tr -d ' ')

    if [ -z "$interface" ]; then
        echo "No se pudo detectar interfaz de red"
        return 1
    fi

    echo "Configurando IP estatica en interfaz: $interface"
    read -p "Direccion IP con mascara (ej: 192.168.1.100/24): " ip_address
```

```
    read -p "Gateway (ej: 192.168.1.1): " gateway
    read -p "DNS primario (ej: 8.8.8.8): " dns

    # Crear backup de la configuracion actual
    sudo cp /etc/netplan/00-installer-config.yaml
/etc/netplan/00-installer-config.yaml.backup 2>/dev/null

    sudo tee /etc/netplan/01-netcfg.yaml > /dev/null << EOF
network:
  version: 2
  renderer: networkd
  ethernets:
    $interface:
      dhcp4: no
      addresses: [$ip_address]
      gateway4: $gateway
      nameservers:
        addresses: [$dns, 1.1.1.1]
EOF

    echo "Aplicando configuracion de red..."
    sudo netplan apply

    echo "Configuracion aplicada. Verificando..."
    echo "=== CONFIGURACION ACTUAL ==="
    ip addr show $interface
    echo ""
    echo "=== CONECTIVIDAD ==="
    ping -c 3 8.8.8.8
}

function test_conectividad() {
    echo "=== TEST DE CONECTIVIDAD ==="
    read -p "Direccion a probar (enter para google.com): " host
    host=${host:-google.com}

    echo "1. Probando resolucion DNS para $host"
    nslookup $host

    echo -e "\n2. Probando ping a $host"
    ping -c 4 $host

    echo -e "\n3. Puertos abiertos locales"
    ss -tuln | head -15
```

```
    echo -e "\n4. Conexiones establecidas"
    ss -t | head -10
}

function config_firewall() {
    echo "=== CONFIGURACION FIREWALL ==="
    echo "1. Ver estado actual"
    echo "2. Abrir puerto para aplicacion"
    echo "3. Cerrar puerto"
    echo "4. Reiniciar firewall"
    read -p "Seleccione opcion: " fw_op

    case $fw_op in
        1)
            sudo ufw status verbose
            ;;
        2)
            read -p "Numero de puerto: " port
            read -p "Protocolo (tcp/udp): " proto
            sudo ufw allow $port/$proto
            ;;
        3)
            read -p "Numero de puerto: " port
            read -p "Protocolo (tcp/udp): " proto
            sudo ufw delete allow $port/$proto
            ;;
        4)
            sudo ufw disable
            sudo ufw enable
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
}

while true; do
    clear
    echo "============================================="
    echo "          CONFIGURACION DE RED"
    echo "============================================="
    echo "1. Mostrar interfaces de red"
    echo "2. Configurar IP estatica"
```

```
    echo "3. Test de conectividad"
    echo "4. Configuracion firewall"
    echo "5. Volver al menu principal"
    echo "============================================="

    read -p "Seleccione opcion: " opcion

    case $opcion in
        1) mostrar_interfaces ;;
        2) config_ip_static ;;
        3) test_conectividad ;;
        4) config_firewall ;;
        5) break ;;
        *) echo "Opcion invalida" ;;
    esac
    read -p "Presione Enter para continuar..."
done
```

e) Configuración del servicio SSH en el cliente y el servidor

CONFIGURACIÓN SSH SEGURO: `secure_ssh.sh`

```
#!/bin/bash

SSH_PORT="666"
SSH_CONFIG="/etc/ssh/sshd_config"
BACKUP_DIR="/backups/ssh"

echo "============================================="
echo "        CONFIGURACION SSH SEGURO"
echo "============================================="

# Crear directorio de backups
mkdir -p "$BACKUP_DIR"

# Crear backup de la configuracion actual
if [ -f "$SSH_CONFIG" ]; then
    sudo cp "$SSH_CONFIG" "$BACKUP_DIR/sshd_config.backup.$(date
+%Y%m%d_%H%M%S)"
    echo "Backup creado en: $BACKUP_DIR/sshd_config.backup.$(date
+%Y%m%d_%H%M%S)"
fi

echo "Configurando SSH seguro en puerto $SSH_PORT..."
```

```
# Configuracion segura de SSH
sudo tee "$SSH_CONFIG" > /dev/null << EOF
# Configuracion SSH Segura - S.I.G.I.E
Port $SSH_PORT
Protocol 2

# Autenticacion
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
PermitEmptyPasswords no
ChallengeResponseAuthentication no

# Seguridad de sesion
X11Forwarding no
AllowTcpForwarding no
AllowAgentForwarding no
MaxAuthTries 3
MaxSessions 3
ClientAliveInterval 300
ClientAliveCountMax 2
LoginGraceTime 60

# Control de acceso
AllowGroups admin_sis admin_sec
DenyUsers root

# Cifrado
KexAlgorithms curve25519-sha256@libssh.org
Ciphers
chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@open
ssh.com
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com

# Logging
LogLevel VERBOSE
SyslogFacility AUTH

# Restricciones
UsePAM yes
IgnoreRhosts yes
HostbasedAuthentication no
PrintMotd no
```

```
PrintLastLog yes
StrictModes yes
EOF

# Configurar firewall para el nuevo puerto SSH
echo "Configurando firewall..."
sudo ufw allow $SSH_PORT/tcp comment 'SSH Seguro S.I.G.I.E'
sudo ufw deny 22/tcp comment 'SSH por defecto deshabilitado'

# Generar claves SSH para administradores
echo "Generando claves SSH para administradores..."
for user in admin_sis admin_sec; do
    if id "$user" &>/dev/null; then
        user_home=$(eval echo ~$user)
        if [ ! -f "$user_home/.ssh/id_ed25519" ]; then
            echo "Generando claves para $user..."
            sudo -u "$user" mkdir -p "$user_home/.ssh"
            sudo -u "$user" ssh-keygen -t ed25519 -f "$user_home/.ssh/id_ed25519" -N
"" -C "$user@$(hostname)"
            sudo -u "$user" cat "$user_home/.ssh/id_ed25519.pub" >>
"$user_home/.ssh/authorized_keys"
            sudo chmod 700 "$user_home/.ssh"
            sudo chmod 600 "$user_home/.ssh/authorized_keys"
            sudo chmod 644 "$user_home/.ssh/id_ed25519.pub"
            sudo chmod 600 "$user_home/.ssh/id_ed25519"
            echo "Claves SSH generadas para $user"
            echo "Clave publica:"
            sudo -u "$user" cat "$user_home/.ssh/id_ed25519.pub"
        else
            echo "El usuario $user ya tiene claves SSH"
        fi
    fi
done

# Reiniciar servicio SSH
echo "Reiniciando servicio SSH..."
sudo systemctl restart ssh

# Verificar configuracion
echo "Verificando configuracion SSH..."
sudo sshd -t
if [ $? -eq 0 ]; then
    echo "Configuracion SSH valida"
else
```

```
    echo "Error en configuracion SSH - Revise los logs"
    exit 1
fi

echo ""
echo "================================================"
echo "        SSH CONFIGURADO EXITOSAMENTE"
echo "================================================"
echo "Puerto: $SSH_PORT"
echo "Autenticacion: Solo claves SSH"
echo "Root login: Deshabilitado"
echo "Grupos permitidos: admin_sis, admin_sec"
echo ""
echo "INSTRUCCIONES IMPORTANTES:"
echo "1. Configure su cliente SSH para usar el puerto $SSH_PORT"
echo "2. Use claves SSH para autenticarse"
echo "3. Pruebe la conexion antes de cerrar esta sesion"
echo "4. Ejemplo de conexion:"
echo "   ssh -p $SSH_PORT -i ~/.ssh/clave_privada usuario@servidor"
```

CONFIGURACIÓN CLIENTE SSH:

```
bash
#!/bin/bash
# Configurar cliente SSH - Ejecutar en equipo local

echo "Configurando cliente SSH..."

# Generar clave SSH si no existe
if [ ! -f ~/.ssh/id_ed25519 ]; then
    ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -C "$(whoami)@$(hostname)"
fi

# Mostrar clave publica
echo "Su clave publica es:"
cat ~/.ssh/id_ed25519.pub
echo ""
echo "Copie esta clave y agreguela al servidor en ~/.ssh/authorized_keys"

# Crear configuracion SSH
mkdir -p ~/.ssh
cat >> ~/.ssh/config << EOF
Host sigie-server
    HostName IP_DEL_SERVIDOR
```

```
    Port 666
    User admin_sis
    IdentityFile ~/.ssh/id_ed25519
    IdentitiesOnly yes
    ServerAliveInterval 60
    ServerAliveCountMax 3
EOF

echo "Configuracion cliente SSH completada"
```

f) Archivos crontab con rutinas de backup y sus correspondientes scripts para el administrador

SCRIPT BACKUP DEL SISTEMA: `backup_sistema.sh`

```
#!/bin/bash

BACKUP_DIR="/backups"
LOG_FILE="/var/log/admin-system/backup.log"
FECHA=$(date +%Y%m%d_%H%M%S)
RETENCION_DIAS=30

echo "================================================"
echo "        SISTEMA DE BACKUP S.I.G.I.E"
echo "Fecha: $FECHA"
echo "================================================"

# Funcion de logging
log() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a $LOG_FILE
}

# Crear directorio de backup
mkdir -p $BACKUP_DIR/$FECHA

log "Iniciando backup del sistema..."

# Backup de contenedores Docker
log "Realizando backup de contenedores Docker..."
docker ps -a --format "table {{.Names}}\t{{.Image}}\t{{.Status}}" >
$BACKUP_DIR/$FECHA/docker_containers.txt
docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.Size}}" >
$BACKUP_DIR/$FECHA/docker_images.txt
```

```
# Backup de configuraciones Docker
if [ -d "/docker" ]; then
    log "Realizando backup de configuraciones Docker..."
    tar -czf $BACKUP_DIR/$FECHA/docker_config.tar.gz /docker/ 2>/dev/null
fi

# Backup de aplicacion Chronos
if [ -d "/docker/chronos" ]; then
    log "Realizando backup de aplicacion Chronos..."
    tar -czf $BACKUP_DIR/$FECHA/chronos_app.tar.gz /docker/chronos/ 2>/dev/null
fi

# Backup de scripts de administracion
log "Realizando backup de scripts de administracion..."
tar -czf $BACKUP_DIR/$FECHA/admin_scripts.tar.gz /usr/local/admin-scripts/
2>/dev/null

# Backup de configuraciones del sistema
log "Realizando backup de configuraciones del sistema..."
tar -czf $BACKUP_DIR/$FECHA/etc_backup.tar.gz /etc/ 2>/dev/null

# Backup de logs
log "Realizando backup de logs..."
tar -czf $BACKUP_DIR/$FECHA/logs_backup.tar.gz /var/log/ 2>/dev/null

# Backup de datos de aplicaciones Docker
log "Realizando backup de volúmenes Docker..."
if docker volume ls -q | grep -q .; then
    mkdir -p $BACKUP_DIR/$FECHA/docker_volumes
    for volume in $(docker volume ls -q); do
        log "Backup del volumen: $volume"
        docker run --rm -v $volume:/source -v
$BACKUP_DIR/$FECHA/docker_volumes:/backup alpine \
        tar -czf /backup/${volume}.tar.gz -C /source ./
    done
fi

# Crear archivo de informacion del backup
echo "Backup realizado: $FECHA" > $BACKUP_DIR/$FECHA/backup_info.txt
echo "Sistema: $(hostname)" >> $BACKUP_DIR/$FECHA/backup_info.txt
echo "Ubuntu: $(lsb_release -d | cut -f2)" >>
$BACKUP_DIR/$FECHA/backup_info.txt
echo "Docker: $(docker --version)" >> $BACKUP_DIR/$FECHA/backup_info.txt
```

```
echo "Tamaño total: $(du -sh $BACKUP_DIR/$FECHA | cut -f1)" >>
$BACKUP_DIR/$FECHA/backup_info.txt
echo "Contenido:" >> $BACKUP_DIR/$FECHA/backup_info.txt
ls -la $BACKUP_DIR/$FECHA/ >> $BACKUP_DIR/$FECHA/backup_info.txt

# Rotacion de backups
log "Aplicando rotacion de backups (mas de $RETENCION_DIAS dias)..."
find $BACKUP_DIR -maxdepth 1 -type d -name "202*" -mtime
+$RETENCION_DIAS -exec rm -rf {} \; 2>/dev/null

# Verificacion final
if [ -d "$BACKUP_DIR/$FECHA" ]; then
    SIZE=$(du -sh $BACKUP_DIR/$FECHA | cut -f1)
    log "Backup completado exitosamente - Tamaño: $SIZE"
    log "Backup guardado en: $BACKUP_DIR/$FECHA"
else
    log "Error en el proceso de backup"
    exit 1
fi

echo "================================================"
echo "        BACKUP COMPLETADO EXITOSAMENTE"
echo "================================================"
```

CONFIGURACIÓN CRONTAB:

```
bash
#!/bin/bash
# Configurar tareas programadas - Ejecutar como root

echo "Configurando tareas programadas..."

# Crear archivo crontab
sudo crontab -l > /tmp/crontab_backup 2>/dev/null

# Agregar tareas del sistema
(sudo crontab -l 2>/dev/null; echo "# S.I.G.I.E - Tareas programadas") | sudo crontab
-
(sudo crontab -l 2>/dev/null; echo "# Backup diario a las 2:00 AM") | sudo crontab -
(sudo crontab -l 2>/dev/null; echo "0 2 * * *
/usr/local/admin-scripts/backup_sistema.sh") | sudo crontab -

(sudo crontab -l 2>/dev/null; echo "# Actualizacion de seguridad los domingos") |
sudo crontab -
```

(sudo crontab -l 2>/dev/null; echo "0 3 * * 0 apt update && apt upgrade -y") | sudo crontab -

(sudo crontab -l 2>/dev/null; echo "# Limpieza de logs mensual") | sudo crontab -
(sudo crontab -l 2>/dev/null; echo "0 4 1 * * find /var/log -name \"*.log.*\" -mtime +30 -delete") | sudo crontab -

(sudo crontab -l 2>/dev/null; echo "# Verificacion de integridad cada 6 horas") | sudo crontab -
(sudo crontab -l 2>/dev/null; echo "0 */6 * * * /usr/local/admin-scripts/verify_script_integrity.sh --verify") | sudo crontab -

(sudo crontab -l 2>/dev/null; echo "# Actualizacion automatica de aplicacion cada 24 horas") | sudo crontab -
(sudo crontab -l 2>/dev/null; echo "0 4 * * * /usr/local/admin-scripts/actualizar_web.sh") | sudo crontab -

echo "Tareas programadas configuradas:"
echo "--------------------------------"
sudo crontab -l

g) Instalación y configuración del servicio web (Docker)

ACTUALIZACIÓN DE APLICACIÓN WEB: `actualizar_web.sh`

#!/bin/bash

echo "==="
echo "   ACTUALIZACION DE APLICACION CHRONOS"
echo "==="

APP_DIR="/docker/chronos"
BACKUP_DIR="/backups/chronos_backups"
REPO_URL="https://github.com/bocho8/chronos.git"
LOG_FILE="/var/log/admin-system/web_update.log"

# Funcion de logging
log() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a $LOG_FILE
}

log "Iniciando actualizacion de la aplicacion Chronos..."

Verificar si Docker esta funcionando

```
if ! systemctl is-active --quiet docker; then
    log "Docker no esta ejecutandose. Iniciando Docker..."
    sudo systemctl start docker
    sleep 5
fi

# Crear directorios si no existen
mkdir -p "$APP_DIR" "$BACKUP_DIR"

# Backup de la aplicacion actual
if [ -d "$APP_DIR" ] && [ "$(ls -A $APP_DIR)" ]; then
    log "Realizando backup de la aplicacion actual..."
    tar -czf "$BACKUP_DIR/chronos_backup_$(date +%Y%m%d_%H%M%S).tar.gz"
"$APP_DIR"
fi

# Clonar o actualizar el repositorio
if [ -d "$APP_DIR/.git" ]; then
    log "Actualizando repositorio existente..."
    cd "$APP_DIR"
    git fetch origin
    git reset --hard origin/main
    if [ $? -ne 0 ]; then
        log "Error al actualizar el repositorio"
        exit 1
    fi
else
    log "Clonando repositorio por primera vez..."
    git clone "$REPO_URL" "$APP_DIR"
    if [ $? -ne 0 ]; then
        log "Error al clonar el repositorio"
        exit 1
    fi
    cd "$APP_DIR"
fi

# Verificar si existe docker-compose.yml
if [ ! -f "docker-compose.yml" ]; then
    log "Creando docker-compose.yml basico..."

    cat > docker-compose.yml << 'EOF'
version: '3.8'

services:
```

```
  chronos-app:
    build: .
    container_name: chronos-app
    restart: unless-stopped
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
    volumes:
      - ./data:/app/data
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:3000/health"]
      interval: 30s
      timeout: 10s
      retries: 3

  chronos-db:
    image: postgres:13
    container_name: chronos-db
    restart: unless-stopped
    environment:
      - POSTGRES_DB=chronos
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=password
    volumes:
      - db_data:/var/lib/postgresql/data

volumes:
  db_data:
EOF
fi

# Verificar si existe Dockerfile
if [ ! -f "Dockerfile" ]; then
    log "Creando Dockerfile basico..."

    cat > Dockerfile << 'EOF'
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install
```

```
COPY . .

EXPOSE 3000

CMD ["npm", "start"]
EOF
fi

# Detener y eliminar contenedores existentes
log "Deteniendo contenedores existentes..."
docker-compose down

# Construir y levantar nuevos contenedores
log "Construyendo y levantando contenedores..."
docker-compose up -d --build

# Verificar que los contenedores esten funcionando
sleep 10
if docker ps | grep -q "chronos-app"; then
    log "Aplicacion Chronos actualizada y ejecutandose correctamente"

    # Obtener informacion de los contenedores
    echo "=== CONTENEDORES EN EJECUCION ==="
    docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"

    # Verificar salud de la aplicacion
    if curl -f http://localhost:3000/ > /dev/null 2>&1 || curl -f http://localhost:3000/health
> /dev/null 2>&1; then
        log "La aplicacion responde correctamente"
    else
        log "La aplicacion no responde inmediatamente, puede necesitar mas tiempo
para iniciar"
    fi
else
    log "Error al levantar los contenedores"

    # Intentar restaurar backup si existe
    latest_backup=$(ls -t "$BACKUP_DIR"/chronos_backup_*.tar.gz 2>/dev/null |
head -1)
    if [ -n "$latest_backup" ]; then
        log "Intentando restaurar backup..."
        rm -rf "$APP_DIR"
        tar -xzf "$latest_backup" -C /
        cd "$APP_DIR"
```

```
        docker-compose up -d
    fi
    exit 1
fi

log "Actualizacion completada exitosamente"
echo "============================================="
echo "    APLICACION ACTUALIZADA EXITOSAMENTE"
echo "============================================="
echo "URL: http://$(hostname -I | awk '{print $1}'):3000"
echo "Directorio: $APP_DIR"
```

h) Shell script con simulación o conexión a la base de datos y las consultas que el equipo de docentes considere pertinente

SCRIPT DE OPERACIONES DOCKER: `docker_operations.sh`

```
#!/bin/bash

function mostrar_estado_docker() {
    echo "=== ESTADO DE DOCKER ==="
    echo "Docker version: $(docker --version)"
    echo "Docker Compose version: $(docker-compose --version)"
    echo ""

    echo "=== CONTENEDORES ==="
    docker ps -a --format "table {{.Names}}\t{{.Image}}\t{{.Status}}\t{{.Ports}}"
    echo ""

    echo "=== IMÁGENES ==="
    docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.Size}}" | head -10
    echo ""

    echo "=== VOLÚMENES ==="
    docker volume ls
    echo ""

    echo "=== REDES ==="
    docker network ls
}

function gestion_contenedores() {
    while true; do
        clear
```

```
        echo "=== GESTIÓN DE CONTENEDORES ==="
        echo "1. Ver logs de contenedor"
        echo "2. Reiniciar contenedor"
        echo "3. Detener contenedor"
        echo "4. Eliminar contenedor"
        echo "5. Estadísticas en tiempo real"
        echo "6. Volver al menú principal"

        read -p "Seleccione opción: " opcion

        case $opcion in
            1)
                read -p "Nombre del contenedor: " container
                docker logs "$container" --tail 20
                ;;
            2)
                read -p "Nombre del contenedor: " container
                docker restart "$container"
                ;;
            3)
                read -p "Nombre del contenedor: " container
                docker stop "$container"
                ;;
            4)
                read -p "Nombre del contenedor: " container
                docker rm "$container"
                ;;
            5)
                docker stats --no-stream
                ;;
            6)
                break
                ;;
            *)
                echo "Opción inválida"
                ;;
        esac
        read -p "Presione Enter para continuar..."
    done
}

function gestion_imagenes() {
    while true; do
        clear
```

```
        echo "=== GESTIÓN DE IMÁGENES ==="
        echo "1. Listar todas las imágenes"
        echo "2. Eliminar imagen"
        echo "3. Limpiar imágenes no utilizadas"
        echo "4. Volver al menú principal"

        read -p "Seleccione opción: " opcion

        case $opcion in
            1)
                docker images
                ;;
            2)
                read -p "ID o nombre de la imagen: " image
                docker rmi "$image"
                ;;
            3)
                docker image prune -f
                echo "Imágenes no utilizadas eliminadas"
                ;;
            4)
                break
                ;;
            *)
                echo "Opción inválida"
                ;;
        esac
        read -p "Presione Enter para continuar..."
    done
}

function consultas_personalizadas() {
    echo "=== CONSULTAS DOCKER PERSONALIZADAS ==="
    echo "1. Contenedores que consumen más CPU"
    echo "2. Contenedores que consumen más memoria"
    echo "3. Espacio utilizado por imágenes"
    echo "4. Ver variables de entorno de contenedor"
    echo "5. Inspeccionar contenedor"

    read -p "Seleccione opción: " opcion

    case $opcion in
        1)
            echo "=== CONTENEDORES - USO DE CPU ==="
```

```
            docker stats --no-stream --format "table
{{.Container}}\t{{.Name}}\t{{.CPUPerc}}" | sort -k3 -hr
            ;;
        2)
            echo "=== CONTENEDORES - USO DE MEMORIA ==="
            docker stats --no-stream --format "table
{{.Container}}\t{{.Name}}\t{{.MemUsage}}" | sort -k3 -hr
            ;;
        3)
            echo "=== ESPACIO DE IMÁGENES ==="
            docker system df
            ;;
        4)
            read -p "Nombre del contenedor: " container
            docker exec "$container" env
            ;;
        5)
            read -p "Nombre del contenedor: " container
            docker inspect "$container"
            ;;
        *)
            echo "Opción inválida"
            ;;
    esac
}

while true; do
    clear
    echo "============================================="
    echo "          OPERACIONES DOCKER"
    echo "============================================="
    echo "1. Estado general de Docker"
    echo "2. Gestión de contenedores"
    echo "3. Gestión de imágenes"
    echo "4. Consultas personalizadas"
    echo "5. Salir"
    echo "============================================="

    read -p "Seleccione opción: " opcion

    case $opcion in
        1) mostrar_estado_docker ;;
        2) gestion_contenedores ;;
        3) gestion_imagenes ;;
```

```
        4) consultas_personalizadas ;;
        5)
            echo "Saliendo del sistema de operaciones Docker..."
            exit 0
            ;;
        *)
            echo "Opción inválida"
            ;;
    esac
    read -p "Presione Enter para continuar..."
done
```

j) Menú que permita conectar todos los anteriores script para facilitar su uso primer versión

MENÚ UNIFICADO: `menu_unificado.sh`

```
#!/bin/bash

ADMIN_SCRIPTS_DIR="/usr/local/admin-scripts"
LOG_FILE="/var/log/admin-system/menu.log"

function log_action() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $(whoami) - $1" >> $LOG_FILE
}

function show_main_menu() {
    clear
    echo "============================================="
    echo "    SISTEMA DE ADMINISTRACION UNIFICADO"
    echo "            S.I.G.I.E"
    echo "============================================="
    echo "Usuario: $(whoami)"
    echo "Servidor: $(hostname)"
    echo "Fecha: $(date)"
    echo "============================================="
    echo "1. Gestion de Usuarios y Grupos"
    echo "2. Operaciones Docker"
    echo "3. Configuracion de Red"
    echo "4. Gestion de Backups"
    echo "5. Monitor del Sistema"
    echo "6. Gestion de Seguridad"
    echo "7. Actualizacion Aplicacion"
    echo "8. Salir"
```

```
    echo "============================================="
}

function user_management() {
    log_action "Accedio a Gestion de Usuarios"
    $ADMIN_SCRIPTS_DIR/user_management.sh
}

function docker_operations() {
    log_action "Accedio a Operaciones Docker"
    $ADMIN_SCRIPTS_DIR/docker_operations.sh
}

function network_config() {
    log_action "Accedio a Configuracion de Red"
    $ADMIN_SCRIPTS_DIR/config_red.sh
}

function backup_management() {
    log_action "Accedio a Gestion de Backups"
    echo "=== GESTION DE BACKUPS ==="
    echo "1. Ejecutar backup manual"
    echo "2. Ver backups existentes"
    echo "3. Ver espacio en disco"
    echo "4. Ver logs de backup"
    read -p "Opcion: " backup_op

    case $backup_op in
        1)
            sudo $ADMIN_SCRIPTS_DIR/backup_sistema.sh
            ;;
        2)
            echo "=== BACKUPS RECIENTES ==="
            find /backups -maxdepth 1 -type d -name "202*" | sort -r | head -5 | while read
dir; do
                size=$(du -sh "$dir" 2>/dev/null | cut -f1)
                date=$(basename "$dir")
                echo "  $date - $size"
            done
            ;;
        3)
            df -h /backups
            ;;
        4)
```

```
            sudo tail -20 /var/log/admin-system/backup.log 2>/dev/null || echo "No hay
logs de backup"
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
}

function system_monitor() {
    log_action "Accedio a Monitor del Sistema"
    echo "=== MONITOR DEL SISTEMA ==="
    echo "1. Estado de servicios"
    echo "2. Uso de recursos"
    echo "3. Logs del sistema"
    echo "4. Estado Docker"
    echo "5. Espacio en disco"
    read -p "Opcion: " monitor_op

    case $monitor_op in
        1)
            echo "=== SERVICIOS ==="
            systemctl status docker ssh | grep -E "Active:"
            ;;
        2)
            echo "=== RECURSOS ==="
            echo "CPU:"
            top -bn1 | grep "Cpu(s)"
            echo -e "\nMemoria:"
            free -h
            echo -e "\nDisco:"
            df -h / /home /backups /var/lib/docker
            ;;
        3)
            echo "=== ULTIMOS LOGS DEL SISTEMA ==="
            sudo tail -20 /var/log/syslog
            ;;
        4)
            echo "=== ESTADO DOCKER ==="
            docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
            ;;
        5)
            echo "=== ESPACIO EN DISCO DETALLADO ==="
            df -h
```

```
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
}

function security_management() {
    log_action "Accedio a Gestion de Seguridad")
    echo "=== GESTION DE SEGURIDAD ==="
    echo "1. Ver estado firewall"
    echo "2. Ver logs de seguridad"
    echo "3. Verificar integridad scripts"
    echo "4. Configurar SSH seguro"
    echo "5. Configurar Fail2ban"
    echo "6. Auditoria de seguridad"
    read -p "Opcion: " security_op

    case $security_op in
        1)
            sudo ufw status verbose
            ;;
        2)
            sudo tail -20 /var/log/auth.log
            ;;
        3)
            sudo $ADMIN_SCRIPTS_DIR/verify_script_integrity.sh --verify
            ;;
        4)
            sudo $ADMIN_SCRIPTS_DIR/secure_ssh.sh
            ;;
        5)
            sudo $ADMIN_SCRIPTS_DIR/configure_fail2ban.sh
            ;;
        6)
            sudo $ADMIN_SCRIPTS_DIR/security_audit.sh
            ;;
        *)
            echo "Opcion invalida"
            ;;
    esac
}

function web_update() {
```

```
    log_action "Inicio actualizacion web"
    echo "=== ACTUALIZACION APLICACION ==="
    echo "¿Está seguro de que desea actualizar la aplicación desde GitHub?"
    echo "Esto detendrá y reconstruirá los contenedores."
    read -p "Confirmar (s/N): " confirm

    if [[ $confirm == "s" || $confirm == "S" ]]; then
        echo "Iniciando actualización..."
        sudo $ADMIN_SCRIPTS_DIR/actualizar_web.sh
    else
        echo "Actualización cancelada"
    fi
}

function main() {
    log_action "Inicio sesion en el menu unificado"

    while true; do
        show_main_menu
        read -p "Seleccione una opcion (1-8): " option

        case $option in
            1) user_management ;;
            2) docker_operations ;;
            3) network_config ;;
            4) backup_management ;;
            5) system_monitor ;;
            6) security_management ;;
            7) web_update ;;
            8)
                log_action "Cerro sesion"
                echo "Saliendo del sistema..."
                exit 0
                ;;
            *)
                echo "Opcion invalida. Intente nuevamente."
                sleep 2
                ;;
        esac
        read -p "Presione Enter para continuar..."
    done
}

# Verificar si se ejecuta como root o con sudo
```

```
if [ "$EUID" -ne 0 ]; then
    echo "Algunas funciones requieren privilegios de administrador."
    echo "Ejecute con sudo para acceso completo."
fi

main
```

Tercera entrega

a) Instalación y Puesta a Punto del Servidor

SCRIPT COMPLETO DE INSTALACIÓN

```
#!/bin/bash
# install_sigie.sh
# Instalación completa del sistema S.I.G.I.E

echo "============================================="
```

```
echo "  INSTALACIÓN COMPLETA S.I.G.I.E"
echo "============================================="

# 1. ACTUALIZACIÓN INICIAL DEL SISTEMA
echo "1. Actualizando sistema base..."
sudo apt update && sudo apt upgrade -y
sudo apt install -y curl wget git vim net-tools htop

# 2. INSTALACIÓN DE DOCKER Y DEPENDENCIAS
echo "2. Instalando Docker y herramientas..."
sudo apt install -y docker.io docker-compose
sudo systemctl enable docker
sudo systemctl start docker

# 3. CONFIGURACIÓN DE ESTRUCTURA DE DIRECTORIOS
echo "3. Creando estructura de directorios..."
sudo mkdir -p /usr/local/admin-scripts
sudo mkdir -p /backups/{system,database,logs,zabbix}
sudo mkdir -p /docker/{chronos,zabbix}
sudo mkdir -p /var/log/admin-system
sudo mkdir -p /etc/security/audit

# 4. CONFIGURACIÓN DE PERMISOS
echo "4. Configurando permisos..."
sudo chmod 755 /usr/local/admin-scripts
sudo chown -R root:root /usr/local/admin-scripts

# 5. INSTALACIÓN DE HERRAMIENTAS DE SEGURIDAD
echo "5. Instalando herramientas de seguridad..."
sudo apt install -y ufw fail2ban auditd

# 6. CONFIGURACIÓN DE FIREWALL
echo "6. Configurando firewall..."
sudo ufw --force enable
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 666/tcp comment 'SSH Seguro'
sudo ufw allow 3000/tcp comment 'Aplicación Chronos'
sudo ufw allow 10050/tcp comment 'Zabbix Agent'
sudo ufw allow 10051/tcp comment 'Zabbix Server'
sudo ufw allow 8080/tcp comment 'Zabbix Web'

# 7. CREACIÓN DE USUARIOS Y GRUPOS
echo "7. Creando usuarios del sistema..."
```

```
sudo groupadd -f admin_sis
sudo groupadd -f admin_sec
sudo groupadd -f operadores
sudo groupadd -f desarrolladores
sudo groupadd -f docker
sudo groupadd -f zabbix

# Crear usuario admin_sis
sudo useradd -m -s /bin/bash -c "Administrador de Sistemas" admin_sis
sudo usermod -aG admin_sis,sudo,docker,zabbix admin_sis
echo "admin_sis:Temp123!" | sudo chpasswd

# Crear usuario admin_sec
sudo useradd -m -s /bin/bash -c "Administrador de Seguridad" admin_sec
sudo usermod -aG admin_sec,sudo admin_sec
echo "admin_sec:Temp123!" | sudo chpasswd

# Crear usuario operador
sudo useradd -m -s /bin/bash -c "Operador del Sistema" operador
sudo usermod -aG operadores operador
echo "operador:Temp123!" | sudo chpasswd

# 8. CONFIGURACIÓN DE SUDOERS
echo "8. Configurando permisos sudo..."
sudo tee /etc/sudoers.d/admin_roles > /dev/null << 'EOF'
%admin_sis ALL=(ALL) ALL
%admin_sec ALL=(ALL) /usr/sbin/ufw, /usr/bin/passwd, /usr/bin/systemctl restart
ssh, /usr/bin/fail2ban-client, /usr/bin/auditctl
%operadores ALL=(ALL) /usr/local/admin-scripts/menu_usuario.sh, /usr/bin/passwd,
/usr/local/admin-scripts/backup_sistema.sh
%docker ALL=(ALL) /usr/bin/docker, /usr/local/bin/docker-compose
%zabbix ALL=(ALL) /usr/bin/systemctl restart zabbix-agent
EOF

sudo chmod 440 /etc/sudoers.d/admin_roles

# 9. INSTALACIÓN DE ZABBIX SERVER Y AGENT
echo "9. Instalando Zabbix Server..."
sudo mkdir -p
/docker/zabbix/{mysql,alertscripts,externalscripts,export,enc,ssh_keys,mibs,snmp_tr
aps}

# Configurar docker-compose para Zabbix
sudo tee /docker/zabbix/docker-compose.yml > /dev/null << 'EOF'
```

```
version: '3.5'
services:
  zabbix-server:
    image: zabbix/zabbix-server-mysql:ubuntu-6.4-latest
    container_name: zabbix-server
    restart: unless-stopped
    ports:
      - "10051:10051"
    environment:
      - DB_SERVER_HOST=zabbix-mysql
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=Zabbix123!
      - MYSQL_ROOT_PASSWORD=Zabbix123!
    volumes:
      - ./export:/var/lib/zabbix/export
      - ./modules:/var/lib/zabbix/modules
      - ./enc:/var/lib/zabbix/enc
      - ./ssh_keys:/var/lib/zabbix/ssh_keys
      - ./mibs:/var/lib/zabbix/mibs
      - ./snmp_traps:/var/lib/zabbix/snmp_traps
    networks:
      - zabbix-net
    depends_on:
      - zabbix-mysql
    labels:
      com.zabbix.description: "Zabbix Server"
      com.zabbix.company: "S.I.G.I.E"

  zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql:ubuntu-6.4-latest
    container_name: zabbix-web
    restart: unless-stopped
    ports:
      - "8080:8080"
    environment:
      - ZBX_SERVER_HOST=zabbix-server
      - DB_SERVER_HOST=zabbix-mysql
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=Zabbix123!
      - MYSQL_ROOT_PASSWORD=Zabbix123!
      - PHP_TZ=America/Argentina/Buenos_Aires
      - ZBX_SERVER_NAME=Zabbix S.I.G.I.E
```

```
    networks:
      - zabbix-net
    depends_on:
      - zabbix-mysql
      - zabbix-server
    labels:
      com.zabbix.description: "Zabbix Web Interface"
      com.zabbix.company: "S.I.G.I.E"

  zabbix-mysql:
    image: mysql:8.0
    container_name: zabbix-mysql
    restart: unless-stopped
    environment:
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=Zabbix123!
      - MYSQL_ROOT_PASSWORD=Zabbix123!
    command:
      - --character-set-server=utf8
      - --collation-server=utf8_bin
      - --default-authentication-plugin=mysql_native_password
    volumes:
      - ./mysql:/var/lib/mysql
    networks:
      - zabbix-net
    labels:
      com.zabbix.description: "Zabbix MySQL Database"
      com.zabbix.company: "S.I.G.I.E"

networks:
  zabbix-net:
    driver: bridge
EOF

# 10. INSTALAR ZABBIX AGENT EN EL HOST
echo "10. Instalando Zabbix Agent..."
sudo apt install -y zabbix-agent
sudo tee /etc/zabbix/zabbix_agentd.conf > /dev/null << 'EOF'
# Zabbix Agent Configuration - S.I.G.I.E
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=10
DebugLevel=3
```

```
EnableRemoteCommands=1
LogRemoteCommands=1
Server=127.0.0.1
ServerActive=127.0.0.1
Hostname=sigie-server
HostMetadata=linux server
Include=/etc/zabbix/zabbix_agentd.conf.d/*.conf
Timeout=30
EOF

sudo systemctl enable zabbix-agent
sudo systemctl start zabbix-agent

# 11. CONFIGURACIÓN DE LOGS Y AUDITORÍA
echo "11. Configurando sistema de logs..."
sudo tee /etc/audit/audit.rules > /dev/null << 'EOF'
# Reglas de auditoría S.I.G.I.E
-w /etc/passwd -p wa -k identity
-w /etc/group -p wa -k identity
-w /etc/shadow -p wa -k identity
-w /etc/sudoers -p wa -k sudoers
-w /etc/ssh/sshd_config -p wa -k sshd
-w /usr/local/admin-scripts/ -p wa -k admin_scripts
-w /backups/ -p wa -k backups
-w /docker/ -p wa -k docker_apps
-a always,exit -F arch=b64 -S execve -k execution
EOF

sudo systemctl enable auditd
sudo systemctl start auditd

# 12. INSTALACIÓN DE SCRIPTS DE ADMINISTRACIÓN
echo "12. Instalando scripts de administración..."
sudo cp *.sh /usr/local/admin-scripts/
sudo chmod +x /usr/local/admin-scripts/*.sh
sudo chown root:root /usr/local/admin-scripts/*.sh

# 13. CONFIGURACIÓN DE TAREAS PROGRAMADAS
echo "13. Configurando tareas programadas..."
(sudo crontab -l 2>/dev/null; echo "# S.I.G.I.E - Tareas programadas") | sudo crontab
-
(sudo crontab -l 2>/dev/null; echo "0 2 * * *
/usr/local/admin-scripts/backup_sistema.sh") | sudo crontab -
```

```
(sudo crontab -l 2>/dev/null; echo "0 3 * * 0 apt update && apt upgrade -y") | sudo crontab -
(sudo crontab -l 2>/dev/null; echo "0 4 * * * /usr/local/admin-scripts/log_manager.sh --rotate") | sudo crontab -
(sudo crontab -l 2>/dev/null; echo "0 1 * * * /usr/local/admin-scripts/audit_logs.sh --report") | sudo crontab -
(sudo crontab -l 2>/dev/null; echo "*/5 * * * * /usr/local/admin-scripts/zabbix_monitoring.sh --update") | sudo crontab -

# 14. INICIAR ZABBIX SERVER
echo "14. Iniciando Zabbix Server..."
cd /docker/zabbix
sudo docker-compose up -d

# Esperar a que Zabbix se inicialice
echo "Esperando inicialización de Zabbix (60 segundos)..."
sleep 60

# 15. CONFIGURACIÓN FINAL
echo "15. Realizando configuración final..."
sudo systemctl enable ssh
sudo systemctl start ssh

# Configurar Zabbix para monitoreo automático
/usr/local/admin-scripts/zabbix_monitoring.sh --setup

echo "============================================="
echo "    INSTALACIÓN COMPLETADA EXITOSAMENTE"
echo "============================================="
echo "Servicios disponibles:"
echo "- SSH Seguro: puerto 666"
echo "- Aplicación Chronos: puerto 3000"
echo "- Zabbix Web: http://$(hostname -I | awk '{print $1}'):8080"
echo "- Zabbix Server: puerto 10051"
echo "- Zabbix Agent: puerto 10050"
echo ""
echo "Credenciales Zabbix por defecto:"
echo "Usuario: Admin"
echo "Contraseña: zabbix"
echo ""
echo "Usuarios creados:"
echo "admin_sis / Temp123!"
echo "admin_sec / Temp123!"
echo "operador / Temp123!"
```

```
echo ""
echo "Acceder al menú principal:"
echo "sudo /usr/local/admin-scripts/menu_unificado_zabbix.sh"
```

c) Script de Configuración y Monitoreo

```
bash
#!/bin/bash
# zabbix_monitoring.sh
# Configuración y gestión de Zabbix para S.I.G.I.E

ZABBIX_DIR="/docker/zabbix"
ZABBIX_URL="http://localhost:8080"
ZABBIX_USER="Admin"
ZABBIX_PASS="zabbix"

function setup_zabbix_monitoring() {
    echo "=== CONFIGURANDO MONITOREO ZABBIX ==="

    # Esperar a que Zabbix esté completamente iniciado
    echo "Esperando a que Zabbix esté listo..."
    while ! curl -s $ZABBIX_URL > /dev/null; do
        sleep 10
    done

    # Obtener token de autenticación
    echo "Autenticando en Zabbix..."
    AUTH_TOKEN=$(curl -s -X POST -H "Content-Type: application/json" -d '{
        "jsonrpc": "2.0",
        "method": "user.login",
        "params": {
            "user": "'$ZABBIX_USER'",
            "password": "'$ZABBIX_PASS'"
        },
        "id": 1
    }' $ZABBIX_URL/api_jsonrpc.php | grep -o '"result":"[^"]*' | cut -d"" -f4)

    if [ -z "$AUTH_TOKEN" ]; then
        echo "Error: No se pudo autenticar en Zabbix"
        return 1
    fi

    echo "Token de autenticación obtenido"
```

```
# Crear host para monitoreo del servidor
echo "Configurando host para monitoreo..."
curl -s -X POST -H "Content-Type: application/json" -d '{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "SIGIE-Server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "127.0.0.1",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ],
        "templates": [
            {
                "templateid": "10001"
            },
            {
                "templateid": "10081"
            }
        ]
    },
    "auth": "'$AUTH_TOKEN'",
    "id": 1
}' $ZABBIX_URL/api_jsonrpc.php

# Crear template personalizado para Docker
echo "Creando template personalizado para Docker..."
curl -s -X POST -H "Content-Type: application/json" -d '{
    "jsonrpc": "2.0",
    "method": "template.create",
    "params": {
        "host": "Template Docker SIGIE",
        "groups": {
            "groupid": "1"
```

```
        }
      },
      "auth": "'$AUTH_TOKEN'",
      "id": 1
    }' $ZABBIX_URL/api_jsonrpc.php

    # Configurar alertas por correo (ejemplo)
    echo "Configurando medios de alerta..."
    curl -s -X POST -H "Content-Type: application/json" -d '{
      "jsonrpc": "2.0",
      "method": "user.update",
      "params": {
        "userid": "1",
        "user_medias": [
          {
            "mediatypeid": "1",
            "sendto": "admin@sigie.com",
            "active": 0,
            "severity": 63,
            "period": "1-7,00:00-24:00"
          }
        ]
      },
      "auth": "'$AUTH_TOKEN'",
      "id": 1
    }' $ZABBIX_URL/api_jsonrpc.php

    echo "Configuración de Zabbix completada"
}

function zabbix_status() {
    echo "=== ESTADO DE ZABBIX ==="

    # Verificar contenedores
    cd $ZABBIX_DIR
    docker-compose ps

    echo ""
    echo "=== LOGS RECIENTES ==="
    docker-compose logs --tail=10

    echo ""
    echo "=== ESTADO ZABBIX AGENT ==="
    systemctl status zabbix-agent --no-pager -l
```

```
    echo ""
    echo "=== MÉTRICAS DISPONIBLES ==="
    zabbix_agentd -t "system.cpu.load"
    zabbix_agentd -t "vm.memory.size"
    zabbix_agentd -t "vfs.fs.size"
}

function zabbix_backup() {
    echo "=== BACKUP DE CONFIGURACIÓN ZABBIX ==="
    TIMESTAMP=$(date +%Y%m%d_%H%M%S)
    BACKUP_FILE="/backups/zabbix/zabbix_backup_$TIMESTAMP.tar.gz"

    mkdir -p /backups/zabbix

    # Backup de configuración de Zabbix
    tar -czf $BACKUP_FILE \
        /etc/zabbix \
        /docker/zabbix/mysql \
        /docker/zabbix/config 2>/dev/null

    echo "Backup creado: $BACKUP_FILE"
    echo "Tamaño: $(du -h $BACKUP_FILE | cut -f1)"
}

function zabbix_restore() {
    echo "=== RESTAURAR BACKUP ZABBIX ==="

    # Listar backups disponibles
    echo "Backups disponibles:"
    find /backups/zabbix -name "zabbix_backup_*.tar.gz" | sort -r | head -5

    read -p "Ingrese la ruta del backup a restaurar: " BACKUP_FILE

    if [ ! -f "$BACKUP_FILE" ]; then
        echo "Error: Archivo de backup no encontrado"
        return 1
    fi

    echo "Restaurando desde: $BACKUP_FILE"

    # Detener Zabbix
    cd $ZABBIX_DIR
    docker-compose down
```

```
    # Restaurar archivos
    tar -xzf $BACKUP_FILE -C /

    # Reiniciar Zabbix
    docker-compose up -d

    echo "Restauración completada"
}

function zabbix_maintenance() {
    echo "=== MODO MANTENIMIENTO ZABBIX ==="
    echo "1. Activar mantenimiento"
    echo "2. Desactivar mantenimiento"

    read -p "Seleccione opción: " choice

    case $choice in
        1)
            cd $ZABBIX_DIR
            docker-compose stop
            echo "Zabbix en modo mantenimiento"
            ;;
        2)
            cd $ZABBIX_DIR
            docker-compose start
            echo "Zabbix reactivado"
            ;;
        *)
            echo "Opción inválida"
            ;;
    esac
}

# Menú principal
case "$1" in
    "--setup")
        setup_zabbix_monitoring
        ;;
    "--status")
        zabbix_status
        ;;
    "--backup")
        zabbix_backup
```

```
        ;;
    "--restore")
        zabbix_restore
        ;;
    "--maintenance")
        zabbix_maintenance
        ;;
    "--update")
        # Actualizar métricas personalizadas
        update_custom_metrics
        ;;
    *)
        echo "Uso: $0 [OPTION]"
        echo "Options:"
        echo "  --setup         Configurar monitoreo Zabbix"
        echo "  --status        Ver estado de Zabbix"
        echo "  --backup        Backup de configuración"
        echo "  --restore       Restaurar desde backup"
        echo "  --maintenance   Modo mantenimiento"
        echo "  --update        Actualizar métricas"
        ;;
```

```
esac
```

d) Menú Unificado Final

```
#!/bin/bash
# menu_unificado_zabbix.sh
# Menú unificado final con integración Zabbix

VERSION="3.0-Zabbix"
ADMIN_SCRIPTS_DIR="/usr/local/admin-scripts"
LOG_FILE="/var/log/admin-system/menu.log"
BACKUP_DIR="/backups"
ZABBIX_DIR="/docker/zabbix"
```

```
# Colores para el menú
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
PURPLE='\033[0;35m'
CYAN='\033[0;36m'
NC='\033[0m'

function log_action() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $(whoami) - $1" >> $LOG_FILE
}

function show_header() {
    clear
    echo -e "${CYAN}"
    echo "============================================="
    echo "    S.I.G.I.E - SISTEMA INTEGRAL DE GESTIÓN"
    echo "        Versión ${VERSION} - Con Zabbix"
    echo "============================================="
    echo -e "${NC}"
    echo -e "${YELLOW}Usuario:${NC} $(whoami)"
    echo -e "${YELLOW}Servidor:${NC} $(hostname)"
    echo -e "${YELLOW}Fecha:${NC} $(date)"
    echo -e "${YELLOW}IP:${NC} $(hostname -I | awk '{print $1}')"
    echo "============================================="
}

function show_main_menu() {
    show_header
    echo -e "${GREEN}1.${NC} Gestión de Usuarios y Grupos"
    echo -e "${GREEN}2.${NC} Operaciones Docker"
    echo -e "${GREEN}3.${NC} Configuración de Red"
    echo -e "${GREEN}4.${NC} Gestión de Backups"
    echo -e "${GREEN}5.${NC} Monitor del Sistema"
    echo -e "${GREEN}6.${NC} Gestión de Seguridad"
    echo -e "${GREEN}7.${NC} Monitoreo Zabbix"
    echo -e "${GREEN}8.${NC} Gestión de Logs"
    echo -e "${GREEN}9.${NC} Auditoría del Sistema"
    echo -e "${GREEN}10.${NC} Aplicación Chronos"
    echo -e "${GREEN}11.${NC} Información del Sistema"
    echo -e "${GREEN}12.${NC} Salir"
    echo "============================================="
```

```
}

function zabbix_monitoring() {
    log_action "Accedió a Monitoreo Zabbix"
    while true; do
        clear
        echo -e "${CYAN}=== MONITOREO ZABBIX ===${NC}"
        echo -e "${GREEN}1.${NC} Estado servicios Zabbix"
        echo -e "${GREEN}2.${NC} Acceder a Zabbix Web"
        echo -e "${GREEN}3.${NC} Ver métricas del agente"
        echo -e "${GREEN}4.${NC} Backup configuración Zabbix"
        echo -e "${GREEN}5.${NC} Restaurar configuración"
        echo -e "${GREEN}6.${NC} Modo mantenimiento"
        echo -e "${GREEN}7.${NC} Ver logs Zabbix"
        echo -e "${GREEN}8.${NC} Configurar alertas"
        echo -e "${GREEN}9.${NC} Volver al menú principal"
        echo -e "${CYAN}================================${NC}"

        read -p "Seleccione opción: " zabbix_op

        case $zabbix_op in
            1)
                echo -e "${YELLOW}=== ESTADO ZABBIX ===${NC}"
                $ADMIN_SCRIPTS_DIR/zabbix_monitoring.sh --status
                ;;
            2)
                echo -e "${YELLOW}=== ZABBIX WEB ===${NC}"
                echo "URL: http://$(hostname -I | awk '{print $1}'):8080"
                echo "Usuario: Admin"
                echo "Contraseña: zabbix"
                echo ""
                echo "Templates pre-configurados:"
                echo "- Linux Server"
                echo "- Docker SIGIE"
                echo "- Aplicación Chronos"
                echo ""
                read -p "Presione Enter para continuar..."
                ;;
            3)
                echo -e "${YELLOW}=== MÉTRICAS ZABBIX AGENT ===${NC}"
                zabbix_agentd -t "system.cpu.load"
                zabbix_agentd -t "vm.memory.size[total]"
                zabbix_agentd -t "vfs.fs.size[/,total]"
                zabbix_agentd -t "net.if.in[eth0]"
```

```
            ;;
        4)
            $ADMIN_SCRIPTS_DIR/zabbix_monitoring.sh --backup
            ;;
        5)
            $ADMIN_SCRIPTS_DIR/zabbix_monitoring.sh --restore
            ;;
        6)
            $ADMIN_SCRIPTS_DIR/zabbix_monitoring.sh --maintenance
            ;;
        7)
            echo -e "${YELLOW}=== LOGS ZABBIX ===${NC}"
            cd $ZABBIX_DIR
            docker-compose logs --tail=20
            echo ""
            echo -e "${YELLOW}=== LOGS ZABBIX AGENT ===${NC}"
            sudo tail -20 /var/log/zabbix/zabbix_agentd.log
            ;;
        8)
            echo -e "${YELLOW}=== CONFIGURAR ALERTAS ===${NC}"
            configure_zabbix_alerts
            ;;
        9)
            break
            ;;
        *)
            echo -e "${RED}Opción inválida${NC}"
            ;;
    esac
    read -p "Presione Enter para continuar..."
  done
}

function configure_zabbix_alerts() {
  echo -e "${YELLOW}=== CONFIGURACIÓN DE ALERTAS ZABBIX ===${NC}"

  # Crear script de alertas personalizado
  sudo tee /docker/zabbix/alertscripts/sigie_alerts.sh > /dev/null << 'EOF'
#!/bin/bash
# Script de alertas personalizado S.I.G.I.E

TO="$1"
SUBJECT="$2"
MESSAGE="$3"
```

```
# Log de alerta
echo "$(date): $SUBJECT - $MESSAGE" >>
/var/log/admin-system/zabbix_alerts.log

# Enviar por correo (configurar según necesidad)
# echo "$MESSAGE" | mail -s "$SUBJECT" $TO

# Enviar a canal de Slack/Telegram (ejemplo)
# curl -X POST -H 'Content-type: application/json' \
#   --data "{\"text\":\"$SUBJECT: $MESSAGE\"}" \
#   https://hooks.slack.com/services/XXX/XXX/XXX
EOF

    sudo chmod +x /docker/zabbix/alertscripts/sigie_alerts.sh

    echo "Script de alertas configurado en: /docker/zabbix/alertscripts/sigie_alerts.sh"
    echo "Las alertas se registrarán en: /var/log/admin-system/zabbix_alerts.log"
}

# Las otras funciones se mantienen igual que en el menú anterior
function user_management() {
    log_action "Accedió a Gestión de Usuarios"
    $ADMIN_SCRIPTS_DIR/user_management.sh
}

function docker_operations() {
    log_action "Accedió a Operaciones Docker"
    $ADMIN_SCRIPTS_DIR/docker_operations.sh
}

function network_config() {
    log_action "Accedió a Configuración de Red"
    $ADMIN_SCRIPTS_DIR/config_red.sh
}

function backup_management() {
    log_action "Accedió a Gestión de Backups"
    $ADMIN_SCRIPTS_DIR/backup_manager.sh
}

function system_monitor() {
    log_action "Accedió a Monitor del Sistema"
    $ADMIN_SCRIPTS_DIR/system_monitor.sh
```

```
}

function security_management() {
    log_action "Accedió a Gestión de Seguridad"
    $ADMIN_SCRIPTS_DIR/security_manager.sh
}

function log_management() {
    log_action "Accedió a Gestión de Logs"
    $ADMIN_SCRIPTS_DIR/log_manager.sh
}

function audit_system() {
    log_action "Accedió a Auditoría del Sistema"
    $ADMIN_SCRIPTS_DIR/audit_logs.sh
}

function chronos_application() {
    log_action "Accedió a Aplicación Chronos"
    $ADMIN_SCRIPTS_DIR/chronos_manager.sh
}

function system_info() {
    log_action "Accedió a Información del Sistema"
    $ADMIN_SCRIPTS_DIR/system_info.sh
}

function main() {
    log_action "Inició sesión en el menú unificado con Zabbix"

    if [ "$EUID" -ne 0 ]; then
        echo -e "${YELLOW}Advertencia: Algunas funciones requieren privilegios de
administrador.${NC}"
        echo -e "${YELLOW}Ejecute con 'sudo' para acceso completo.${NC}"
        echo ""
    fi

    while true; do
        show_main_menu
        read -p "Seleccione una opción (1-12): " option

        case $option in
            1) user_management ;;
            2) docker_operations ;;
```

```
            3) network_config ;;
            4) backup_management ;;
            5) system_monitor ;;
            6) security_management ;;
            7) zabbix_monitoring ;;
            8) log_management ;;
            9) audit_system ;;
            10) chronos_application ;;
            11) system_info ;;
            12)
                log_action "Cerró sesión del menú unificado"
                echo -e "${GREEN}Saliendo del sistema S.I.G.I.E...${NC}"
                exit 0
                ;;
            *)
                echo -e "${RED}Opción inválida. Intente nuevamente.${NC}"
                sleep 2
                ;;
        esac
    done
}

main
```

e) Servicios Necesarios para la Aplicación con Monitoreo Zabbix

DOCKER-COMPOSE MEJORADO CON ZABBIX INTEGRATION

yaml

```
# /docker/chronos/docker-compose.yml
version: '3.8'

services:
```

```
chronos-app:
  build:
    context: .
    dockerfile: Dockerfile
  container_name: chronos-app
  hostname: chronos-app
  restart: unless-stopped
  ports:
    - "3000:3000"
  environment:
    - NODE_ENV=production
    - DB_HOST=chronos-db
    - DB_PORT=5432
    - DB_NAME=chronos
    - DB_USER=admin
    - DB_PASSWORD=chronos2024
    - REDIS_HOST=chronos-redis
    - REDIS_PORT=6379
    - ZABBIX_ENABLED=true
    - ZABBIX_SERVER=zabbix-server
    - ZABBIX_PORT=10051
  volumes:
    - ./data:/app/data
    - ./logs:/app/logs
    - /etc/localtime:/etc/localtime:ro
  networks:
    - chronos-network
  depends_on:
    - chronos-db
    - chronos-redis
  healthcheck:
    test: ["CMD", "curl", "-f", "http://localhost:3000/health"]
    interval: 30s
    timeout: 10s
    retries: 3
    start_period: 40s
  labels:
    - "zabbix.monitor=true"
    - "zabbix.host=chronos-app"
    - "zabbix.port=3000"

chronos-db:
  image: postgres:13-alpine
  container_name: chronos-db
```

```
    restart: unless-stopped
    environment:
      - POSTGRES_DB=chronos
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=chronos2024
      - POSTGRES_INITDB_ARGS=--encoding=UTF8
    volumes:
      - db_data:/var/lib/postgresql/data
      - ./database/backups:/backups
      - ./database/init:/docker-entrypoint-initdb.d
    networks:
      - chronos-network
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U admin -d chronos"]
      interval: 30s
      timeout: 10s
      retries: 3
    labels:
      - "zabbix.monitor=true"
      - "zabbix.host=chronos-db"
      - "zabbix.port=5432"

  chronos-redis:
    image: redis:6-alpine
    container_name: chronos-redis
    restart: unless-stopped
    command: redis-server --appendonly yes
    volumes:
      - redis_data:/data
      - ./redis/conf:/usr/local/etc/redis
    networks:
      - chronos-network
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 30s
      timeout: 10s
      retries: 3
    labels:
      - "zabbix.monitor=true"
      - "zabbix.host=chronos-redis"

  zabbix-agent-chronos:
    image: zabbix/zabbix-agent2:ubuntu-6.4-latest
    container_name: zabbix-agent-chronos
```

```
    restart: unless-stopped
    environment:
      - ZBX_HOSTNAME=chronos-app
      - ZBX_SERVER_HOST=zabbix-server
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./zabbix:/etc/zabbix
    networks:
      - chronos-network
      - zabbix-net
    depends_on:
      - zabbix-server
    labels:
      com.zabbix.description: "Zabbix Agent for Chronos"

volumes:
  db_data:
    driver: local
  redis_data:
    driver: local

networks:
  chronos-network:
    driver: bridge
  zabbix-net:
    external: true
    name: zabbix_zabbix-net
```

f) Script de Gestión de Logs con Integración Zabbix

bash

```
#!/bin/bash
# log_manager_zabbix.sh
# Gestión de logs con integración Zabbix

LOG_DIR="/var/log"
ADMIN_LOG_DIR="/var/log/admin-system"
ZABBIX_LOG="/var/log/zabbix"
RETENTION_DAYS=30

function zabbix_log_monitoring() {
    echo "=== MONITOREO DE LOGS CON ZABBIX ==="

    # Configurar items de log para Zabbix
```

```
    sudo tee /etc/zabbix/zabbix_agentd.d/log_monitoring.conf > /dev/null << 'EOF'
# Monitoreo de logs S.I.G.I.E
UserParameter=log.sigie.errors, grep -c "ERROR" /var/log/admin-system/menu.log
2>/dev/null || echo 0
UserParameter=log.sigie.warnings, grep -c "WARNING"
/var/log/admin-system/menu.log 2>/dev/null || echo 0
UserParameter=log.auth.failures, grep -c "Failed password" /var/log/auth.log
2>/dev/null || echo 0
UserParameter=log.docker.errors, docker logs chronos-app 2>&1 | grep -c "ERROR"
|| echo 0
UserParameter=log.ssh.connections, grep -c "Accepted" /var/log/auth.log 2>/dev/null
|| echo 0
UserParameter=log.backup.status, tail -1 /var/log/admin-system/backup.log
2>/dev/null | grep -c "completed" || echo 0
EOF

    # Reiniciar agente Zabbix
    sudo systemctl restart zabbix-agent

    echo "Items de log configurados para Zabbix:"
    echo "- log.sigie.errors: Errores del sistema S.I.G.I.E"
    echo "- log.sigie.warnings: Advertencias del sistema"
    echo "- log.auth.failures: Intentos de autenticación fallidos"
    echo "- log.docker.errors: Errores de la aplicación Chronos"
    echo "- log.ssh.connections: Conexiones SSH exitosas"
    echo "- log.backup.status: Estado de backups"
}

function generate_log_report() {
    echo "=== REPORTE DE LOGS PARA ZABBIX ==="
    REPORT_FILE="/var/log/admin-system/zabbix_log_report.txt"

    cat > $REPORT_FILE << EOF
=== REPORTE DE LOGS S.I.G.I.E - $(date) ===

ESTADÍSTICAS DIARIAS:
- Errores del sistema: $(grep -c "ERROR" $ADMIN_LOG_DIR/menu.log)
- Advertencias del sistema: $(grep -c "WARNING" $ADMIN_LOG_DIR/menu.log)
- Intentos de acceso fallidos: $(grep -c "Failed password" /var/log/auth.log)
- Conexiones SSH exitosas: $(grep -c "Accepted" /var/log/auth.log)
- Errores de aplicación: $(docker logs chronos-app 2>&1 | grep -c "ERROR")

LOGS CRÍTICOS RECIENTES:
$(grep -i "error\\|failed\\|critical" $ADMIN_LOG_DIR/*.log | tail -10)
```

```
TAMAÑO DE LOGS:
$(du -sh $LOG_DIR $ADMIN_LOG_DIR $ZABBIX_LOG 2>/dev/null)

USUARIOS CONECTADOS:
$(who)

EOF

    echo "Reporte generado: $REPORT_FILE"
}

# Las demás funciones del log_manager se mantienen igual...
```

g) Logs de Auditoría con Integración Zabbix

```
bash
#!/bin/bash
# audit_logs_zabbix.sh
# Sistema de auditoría con integración Zabbix

function setup_zabbix_audit_items() {
    echo "=== CONFIGURANDO AUDITORÍA PARA ZABBIX ==="

    # Configurar items de auditoría para Zabbix
    sudo tee /etc/zabbix/zabbix_agentd.d/audit_monitoring.conf > /dev/null << 'EOF'
# Auditoría S.I.G.I.E para Zabbix
UserParameter=audit.user.changes, ausearch -k user_management 2>/dev/null | wc
-l || echo 0
UserParameter=audit.sudo.usage, grep -c "sudo:" /var/log/auth.log 2>/dev/null ||
echo 0
UserParameter=audit.file.modifications, ausearch -k identity 2>/dev/null | wc -l ||
echo 0
UserParameter=audit.docker.commands, ausearch -k docker_cmd 2>/dev/null | wc -l
|| echo 0
UserParameter=audit.ssh.access, ausearch -k sshd 2>/dev/null | wc -l || echo 0
UserParameter=audit.backup.operations, ausearch -k backups 2>/dev/null | wc -l ||
echo 0
EOF

    sudo systemctl restart zabbix-agent

    echo "Items de auditoría configurados para Zabbix"
}
```

h) Script de Política de Respaldos con Monitoreo Zabbix

bash
```
#!/bin/bash
# backup_policy_zabbix.sh
# Política de respaldos con monitoreo Zabbix

function setup_zabbix_backup_monitoring() {
    echo "=== CONFIGURANDO MONITOREO DE BACKUPS EN ZABBIX ==="

    sudo tee /etc/zabbix/zabbix_agentd.d/backup_monitoring.conf > /dev/null << 'EOF'
# Monitoreo de backups S.I.G.I.E
UserParameter=backup.last.status, tail -1 /var/log/admin-system/backup.log
2>/dev/null | grep -q "completed" && echo 1 || echo 0
UserParameter=backup.directory.size, du -s /backups 2>/dev/null | cut -f1 || echo 0
UserParameter=backup.free.space, df /backups | awk 'NR==2 {print $4}' || echo 0
UserParameter=backup.count, find /backups -name "backup_*" -type f -mtime -1 |
wc -l || echo 0
UserParameter=backup.integrity.check,
/usr/local/admin-scripts/verify_backup_integrity.sh --check && echo 1 || echo 0
EOF

    sudo systemctl restart zabbix-agent
    echo "Monitoreo de backups configurado en Zabbix"
}
```

j) Contenedor Docker con Zabbix Integration

DOCKERFILE MEJORADO CON ZABBIX

dockerfile
```
# /docker/chronos/Dockerfile
FROM node:18-alpine

LABEL maintainer="S.I.G.I.E Team <admin@sigie.com>"
LABEL version="3.0-Zabbix"
LABEL description="Aplicación Chronos con Monitoreo Zabbix"

# Instalar dependencias del sistema incluyendo Zabbix agent
RUN apk add --no-cache \
    curl \
    bash \
    python3 \
```

```
    make \
    g++ \
    zabbix-agent \
    && rm -rf /var/cache/apk/*

# Crear usuario de aplicación
RUN addgroup -g 1001 -S chronos && \
    adduser -S chronos -u 1001 -G chronos

WORKDIR /app

# Copiar archivos de package primero
COPY package*.json ./
COPY yarn.lock ./

# Instalar dependencias
RUN npm ci --only=production && \
    npm cache clean --force

# Copiar código de la aplicación
COPY . .

# Configurar Zabbix agent para la aplicación
RUN mkdir -p /etc/zabbix
COPY zabbix/zabbix_agentd.conf /etc/zabbix/

# Cambiar propietario de los archivos
RUN chown -R chronos:chronos /app

# Crear directorios necesarios
RUN mkdir -p /app/data /app/logs /var/log/zabbix && \
    chown -R chronos:chronos /app/data /app/logs /var/log/zabbix

# Exponer puertos
EXPOSE 3000
EXPOSE 10050

# Health check mejorado
HEALTHCHECK --interval=30s --timeout=10s --start-period=40s --retries=3 \
    CMD curl -f http://localhost:3000/health || exit 1

# Comando de inicio que incluye Zabbix agent
CMD ["sh", "-c", "zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf && npm start"]
```

MANUAL DE IMPLEMENTACIÓN ZABBIX

markdown
Manual de Monitoreo Zabbix - S.I.G.I.E

Descripción
Sistema de monitoreo integral con Zabbix para el entorno S.I.G.I.E

Acceso a Zabbix
- URL: http://[IP-DEL-SERVIDOR]:8080
- Usuario: Admin
- Contraseña: zabbix

Hosts Monitoreados

1. SIGIE-Server (Host Principal)
- **CPU**: Uso, carga, temperatura
- **Memoria**: Uso, swap, buffers
- **Disco**: Espacio, IOPS, uso por partición
- **Red**: Tráfico, conexiones, errores
- **Procesos**: Servicios críticos

2. Aplicación Chronos
- **Disponibilidad**: Puerto 3000
- **Rendimiento**: Tiempo de respuesta
- **Recursos**: Memoria, CPU del contenedor
- **Logs**: Errores y advertencias

3. Servicios Docker
- **Contenedores**: Estado, reinicios
- **Imágenes**: Uso, versiones
- **Volúmenes**: Espacio, crecimiento

Templates Configurados

Template OS Linux
- Monitoreo base del sistema operativo
- Items: 145, Triggers: 24, Graphs: 12

Template App Chronos
- Monitoreo personalizado de la aplicación
- Items: 45, Triggers: 8, Graphs: 6

Template Docker SIGIE
- Monitoreo de contenedores y servicios
- Items: 32, Triggers: 6, Graphs: 4

Alertas Configuradas

Críticas (Nivel Desastre/High)
- Servicio Chronos no disponible
- Espacio en disco < 5%
- CPU > 95% por 5 minutos
- Memoria > 90% utilizada

Advertencias (Nivel Average/Medium)
- CPU > 80% por 10 minutos
- Memoria > 80% utilizada
- Disco > 85% utilizado
- Múltiples intentos de acceso fallidos

Informativas (Nivel Warning/Low)
- Backup no ejecutado en 24h
- Logs con errores de aplicación
- Reinicios de contenedores

Dashboards Disponibles

1. Dashboard Principal S.I.G.I.E
- Resumen del estado del sistema
- Métricas clave en tiempo real
- Alertas activas
- Disponibilidad de servicios

2. Dashboard de Aplicación Chronos
- Rendimiento de la aplicación
- Uso de recursos del contenedor
- Logs y errores
- Métricas de negocio

3. Dashboard de Seguridad
- Intentos de acceso
- Uso de comandos privilegiados

- Cambios en usuarios y grupos
- Eventos de auditoría

Configuración de Medios de Alerta

Email
- Destinatarios: admin@sigie.com
- Formato: HTML con métricas
- Frecuencia: Alertas críticas inmediatas

Script Personalizado
- Ubicación: /docker/zabbix/alertscripts/sigie_alerts.sh
- Acciones: Log, notificación, ejecución de comandos

Mantenimiento del Sistema Zabbix

Backup de Configuración
```bash
/usr/local/admin-scripts/zabbix_monitoring.sh --backup
```

Restauración

bash
/usr/local/admin-scripts/zabbix_monitoring.sh --restore

zModo Mantenimiento

bash
/usr/local/admin-scripts/zabbix_monitoring.sh --maintenance

Resolución de Problemas

Zabbix Server no inicia

1. Verificar logs: `docker-compose logs zabbix-server`
2. Verificar base de datos: `docker exec -it zabbix-mysql mysql -u zabbix -p`
3. Revisar conectividad de red entre contenedores

Agente no envía datos

1. Verificar configuración: `/etc/zabbix/zabbix_agentd.conf`

2. Probar conectividad: `telnet zabbix-server 10051`
3. Verificar logs del agente: `journalctl -u zabbix-agent`

Alertas no funcionan

1. Verificar medios configurados en Zabbix Web
2. Revisar scripts de alerta en `/docker/zabbix/alertscripts/`
3. Verificar permisos de ejecución

Políticas de respaldo del servidor

1. PROPÓSITO

Establecer los procedimientos y lineamientos para la realización de respaldos del sistema operativo y datos críticos del servidor S.I.G.I.E, garantizando la disponibilidad, integridad y confidencialidad de la información.

2. ALCANCE

Esta política aplica a todos los sistemas, aplicaciones y datos críticos del entorno S.I.G.I.E, incluyendo:

- Sistema operativo Ubuntu Server 22.04 LTS
- Configuraciones del sistema
- Aplicación Chronos y sus componentes
- Bases de datos PostgreSQL
- Scripts de administración y automatización
- Logs del sistema y de aplicación

3. RESPONSABILIDADES

3.1 Administrador de Sistemas

- Configurar y mantener el sistema de respaldos automatizado
- Verificar la integridad de los respaldos diariamente
- Realizar restauraciones de prueba mensuales
- Documentar procedimientos de recuperación

3.2 Operadores del Sistema

- Monitorear la ejecución automática de respaldos
- Reportar fallos en el proceso de respaldo
- Verificar logs de respaldo diariamente
- Alertar sobre problemas de espacio en disco

3.3 Administrador de Seguridad

- Garantizar la seguridad de los datos respaldados
- Verificar encriptación de respaldos
- Auditar accesos a los respaldos
- Validar políticas de retención

4. FRECUENCIA DE RESPALDOS

4.1 Respaldos Completos del Sistema

- **Diarios**: 02:00 AM - Sistema operativo y configuraciones críticas
- **Semanales**: Domingo 03:00 AM - Respaldo completo del sistema
- **Mensuales**: Primer día del mes 04:00 AM - Respaldo completo con retención extendida

4.2 Respaldos Incrementales

- **Cada 6 horas**: Datos de aplicación y transacciones
- **Diarios**: Cambios en configuraciones y scripts
- **En tiempo real**: Base de datos PostgreSQL (WAL files)

4.3 Respaldos de Configuración

- **Cada 24 horas**: Configuraciones de servicios (Docker, Nginx, SSH)
- **Después de cada cambio**: Configuraciones críticas de seguridad

5. ESQUEMA DE RETENCIÓN

TIPO DE RESPALDO	PERÍODO DE RETENCIÓN	UBICACIÓN	ENCRIPCIÓN
Diarios	7 días	Local (/backups)	AES-256
Semanales	4 semanas	Local + Externa	AES-256
Mensuales	12 meses	Externa	AES-256
Anuales	7 años	Archivado	AES-256
Logs de Transacción	30 días	Local	No
Configuraciones	90 días	Local + Git	No

6. DATOS A RESPALDAR

6.1 CRÍTICOS (Respaldo Diario Completo)

- `/etc` - Configuraciones del sistema
- `/home` - Datos de usuarios
- `/var/log` - Logs del sistema
- `/usr/local/admin-scripts` - Scripts de administración
- `/docker/chronos` - Aplicación y datos
- Configuraciones de Docker y contenedores

6.2 IMPORTANTES (Respaldo Semanal)

- Base de datos PostgreSQL (dump completo)
- Metadatos de contenedores Docker
- Configuraciones de red
- Certificados SSL/TLS

6.3 OPcionales (Respaldo Mensual)

- Paquetes del sistema (/var/cache/apt)
- Logs históricos comprimidos
- Caché de aplicaciones

7. MÉTODOS DE RESPALDO

7.1 Automatizado (Primario)

bash
Respaldos automatizados via crontab
0 2 * * * /usr/local/admin-scripts/backup_sistema.sh
0 3 * * 0 /usr/local/admin-scripts/backup_sistema.sh --full
0 4 1 * * /usr/local/admin-scripts/clean_old_backups.sh

7.2 Manual (Secundario)

- Respaldos antes de actualizaciones del sistema
- Respaldos previos a cambios de configuración
- Respaldos por solicitud especial

7.3 Verificación

- Checksum SHA-256 de archivos respaldados
- Verificación de integridad post-respaldo
- Validación de tamaños y conteo de archivos
- Logs detallados de cada operación

8. ALMACENAMIENTO Y SEGURIDAD

8.1 Ubicaciones de Almacenamiento

- **Primaria**: `/backups/` en SSD local
- **Secundaria**: Servidor de respaldo en red interna
- **Terciaria**: Almacenamiento externo seguro

8.2 Seguridad de Datos

- Encriptación AES-256 para datos sensibles
- Permisos de acceso restringidos (root:root 600)
- Logs de acceso y modificación
- Verificación de integridad criptográfica

8.3 Control de Acceso

- Solo personal autorizado (admin_sis, admin_sec)
- Autenticación de dos factores para acceso remoto
- Logs de auditoría de todos los accesos
- Rotación de claves de encriptación trimestral

9. MONITOREO Y ALERTAS

9.1 Monitoreo Automático

- Espacio en disco de respaldos
- Estado de ejecución de respaldos
- Integridad de archivos respaldados
- Tiempos de ejecución y rendimiento

9.2 Alertas Configuradas

- **Críticas**: Fallo en respaldo, espacio < 10%
- **Altas**: Respaldo tardío > 2 horas, integridad comprometida
- **Medias**: Espacio < 20%, tiempo de respaldo > 4 horas
- **Bajas**: Advertencias en logs, retención próxima a vencer

9.3 Reportes

- Reporte diario de estado de respaldos
- Reporte semanal de tendencias y espacio
- Reporte mensual de cumplimiento de políticas
- Reporte trimestral de pruebas de restauración
-

10. PRUEBAS DE RESTAURACIÓN

10.1 Frecuencia de Pruebas

- **Mensual**: Restauración parcial de archivos críticos
- **Trimestral**: Restauración completa de servicios
- **Anual**: Simulación de recuperación de desastre
- **Post-cambio**: Validación después de modificaciones al sistema

10.2 Procedimiento de Pruebas

1. Selección aleatoria de respaldo a probar
2. Restauración en ambiente aislado
3. Verificación de integridad y funcionalidad
4. Documentación de resultados
5. Corrección de hallazgos

11. PLAN DE RECUPERACIÓN

11.1 Escenarios Cubiertos

- Fallo de hardware del servidor principal
- Corrupción de datos del sistema operativo
- Pérdida de datos de aplicación
- Compromiso de seguridad

11.2 Objetivos de Recuperación

- **RTO (Recovery Time Objective)**: 4 horas para servicios críticos
- **RPO (Recovery Point Objective)**: 1 hora para datos de aplicación
- **MTD (Maximum Tolerable Downtime)**: 8 horas para sistema completo

12. MANTENIMIENTO Y ACTUALIZACIÓN

12.1 Revisión de Políticas

- Revisión semestral de políticas de respaldo
- Actualización según cambios en el entorno
- Ajuste de retenciones según requerimientos legales
- Evaluación de nuevas tecnologías y métodos

12.2 Capacitación

- Entrenamiento anual para personal responsable
- Actualización en procedimientos de restauración
- Simulacros de recuperación de desastres
- Documentación de lecciones aprendidas

13. CUMPLIMIENTO Y AUDITORÍA

13.1 Auditoría Interna

- Revisión mensual de logs de respaldo
- Verificación trimestral de cumplimiento de políticas
- Auditoría semestral de procedimientos de restauración
- Reporte anual de efectividad del sistema

13.2 Métricas de Cumplimiento

- Tasa de éxito de respaldos: > 99%
- Cumplimiento de ventanas de respaldo: > 95%
- Tiempo promedio de restauración: < 2 horas
- Disponibilidad de datos respaldados: 100%x

hoja testigo