



S.I.G.T.

Sistemas Operativos III

Katsu enterprise (勝つ企業)

| Rol | Apellido | Nombre | C.I | Email | Tel/Cel. |
|-----------------|----------|----------|-------------|------------------------------|-------------|
| Coordinador | Macedo | Fiorella | 5.503.612-7 | fiomacedoo@gmail.com | 093 646 109 |
| Sub-Coordinador | Dávila | Oriana | 5.074.874-1 | orianadavila99@gmail.com | 093 308 483 |
| Integrante 1 | Pérez | Lautaro | 5.468.712-7 | pabloramirez199221@gmail.com | 097 967 986 |

Docente: Domínguez, Walter.

Fecha de culminación

11/09/2023

SEGUNDA ENTREGA



Índice

| | |
|---------------------------------------------------------------------------------------------|----|
| Índice..... | 1 |
| Usuarios necesarios en el sistema operativo creados de acuerdo al estudio de roles..... | 2 |
| Menú para el Operador del Centro de Cómputos (Administrador del Sistema)..... | 3 |
| Script usuarios.sh..... | 4 |
| Script grupos.sh..... | 9 |
| Script de Redes..... | 13 |
| Script de Servicios..... | 15 |
| Script de Firewall..... | 16 |
| Script de respaldo total..... | 18 |
| Configuraciones de red en las terminales y el servidor..... | 22 |
| Configuración del servicio SSH en el cliente y el servidor..... | 23 |
| Archivos cuenta con rutinas de backup y sus correspondientes scripts para el administrador. | |
| 24 | |
| Configuración del firewall de Gnu/Linux..... | 24 |
| Filtrado de Ips mediante Firewall..... | 25 |



Usuarios necesarios en el sistema operativo creados de acuerdo al estudio de roles.

Operador de hardware.

Los usuarios operadores son aquellos quienes tienen control sobre el Hardware de los equipos que componen el sistema, por lo tanto, los trabajadores de Katsu Enterprise son considerados usuarios operadores.

Administrador (root)

El rol de usuario administrador es el encargado de gestionar la totalidad del servidor:

Debido a que en una empresa de informática cuando se trabaja en servidores, se cuentan con diferentes perfiles, se crearán grupos en los cuales se definirá los diferentes permisos dentro del archivo sudoers. Esta configuración se realizará mediante alias y reglas de acceso.

Los perfiles que se crearán serán los siguientes:

- Administrador de paquetes
- Administrador de base de datos
- Administrador de redes/firewall
- Otras configuraciones en el sistema.



Menú para el Operador del Centro de Cómputos (Administrador del Sistema)

Script de usuario

A continuación se detallarán los Script que se encuentran dentro de la máquina virtual CentOS 7.

A tener en cuenta que ambos archivos son bash, por ende, el principio de cada script es: `#!/bin/bash`.

También, ambos archivos fueron creados en una carpeta llamada “crud,” utilizando el comando “mkadir” para crearlo en cmd. Los archivos fueron creados y escritos dentro de el directorio crud (ingresando al mismo con el comando cd) y creados con nano.

Es de importancia aclarar que se hará este menú en formato Zenity para el día de la última entrega (13/11/2023), hasta el momento, nos enfocamos en hacer el menú de usuarios funcional, no obstante, el resto de menus se realizaron en zenity.



Script usuarios.sh

Funciones:

- Función menú: la función expone un menú, en el cual, el usuario elige qué acción quiere realizar ingresando el número correspondiente.

```
function menu(){  
    clear  
    echo "MENÚ DE GESTIÓN DE USUARIOS"  
    echo "1 - Agregar usuario"  
    echo "2 - Borrar usuario"  
    echo "3 - Listar usuarios del sistema"  
    echo "4 - Buscar un usuario en el sistema"  
    echo "5 - Cambiar contraseña de un usuario"  
    echo "6 - Bloquear usuario"  
    echo "7 - Desbloquear usuario"  
    echo "0 - Salir"  
}
```



- Función agregar_usuario: Esta función permite ingresar usuarios al sistema en un formato de apellidonombre, igualmente, en caso de agregarse un nombre con mayúsculas (ej.: MARTINEZRAUL), la función está programada para guardar todo en minúsculas.

```
function agregar_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    nomb=$(cat /etc/passwd | grep -c $usuario)
    if [ $nomb -eq 1 ]; then
        echo "El usuario ya existe"
        read pausa
    else
        echo "Ingrese el grupo: "
        read grupo
        user_group=$(echo $grupo | tr [:upper:] [:lower:])
        grup=$(cat /etc/group | grep -c $user_group)
        if [ $grup -eq 1 ]; then
            useradd -g $user_group -d /home/$usuario -s /bin/bash $usuario
            passwd -e -d $usuario
            echo "usuario dado de alta"
            read pausa
        else
            echo "El grupo no existe"
            read pausa
        fi
    fi
}
```

- Función borrar_usuario: Como deja a imaginar el nombre de la función, ésta permite borrar usuarios. Tiene la capacidad de entender el nombre de usuario aunque se le escriba en mayúsculas.

```
function borrar_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "El usuario $usuario será eliminado del sistema, está seguro que desea eliminarlo S/N ?"
        read letra
        if [ $letra == 'S' -o == 's' ]; then
            echo "Usuario eliminado del sistema, presione enter para continuar"
            userdel $usuario
            read pausa
        else
            echo "Operación cancelada, presione enter para volver al menú principal"
            read pausa
        fi
    else
        echo "El usuario no existe en el sistema, presione enter para volver al menú principal"
        read pausa
    fi
}
```



- Función listar_usuarios: Posibilita verificar qué usuarios están ingresados..

```
function listar_usuarios(){
    echo "USUARIOS DEL SISTEMA"
    cut -d ":" -f 1 /etc/passwd | sort | more
    echo "Presione enter para volver al menú principal"
    read pausa
}
```

- Función buscar_usuario: Si se desea buscar un usuario en específico, podemos hacerlo con la función buscar usuario.

```
function buscar_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "El usuario: $usuario existe en el sistema"
        cat /etc/passwd | grep $usuario
        echo "presione enter para continuar"
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```

- Función cambiar_contra_usuario: Permite cambiar la contraseña del usuario.

```
function cambiar_contra_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "Se procede a cambiar la contraseña al usuario $usuario"
        passwd $usuario
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```



- Función bloquear_usuario: Autoriza el bloqueo de los usuarios.

```
function bloquear_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "Se procede a bloquear la cuenta del usuario $usuario"
        usermod -L $usuario
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```

- Función desbloquear_usuario: Ante usuarios bloqueados, podemos desbloquearlos con esta función.

```
function desbloquear_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "Se procede a desbloquear la cuenta del usuario $usuario"
        usermod -U $usuario
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```




- Main principal del script:

```
while [ $opc -ne 0 ]
do
    clear
    menu
    read -p "Ingrese la opción correspondiente: " opc
    case $opc in
        1)
            agregar_usuario;;
        2)
            borrar_usuario;;
        3)
            listar_usuarios;;
        4)
            buscar_usuario;;
        5)
            cambiar_contra_usuario;;
        6)
            bloquear_usuario;;
        7)
            desbloquear_usuario;;
        0)
            echo "Volviendo al menú principal"; break ;;
        *)
            echo "Seleccionó una opción incorrecta";;
    esac
done
```



Script grupos.sh

Funciones:

Las funciones de este script son prácticamente idénticas a las de usuarios.sh, con la diferencia de que éste script tiene sus propias variables.

```
function menu(){
    clear
    echo "MENÚ DE GESTIÓN DE GRUPOS"
    echo "1 - Agregar grupo"
    echo "2 - Borrar grupo"
    echo "3 - Listar grupo del sistema"
    echo "4 - Buscar un grupo en el sistema"
    echo "0 - Salir"
}
```

```
function agregar_grupo(){
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_grupo=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_grupo)
    if [ $existe -eq 0 ]; then
        groupadd $user_grupo
        echo "grupo creado, presione enter para continuar"
        read pausa
    else
        echo "El grupo ya existe"
        read pausa
    fi
fi
}
```



```
function borrar_grupo(){
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 1 ]; then
        groupdel $user_group
        echo "grupo borrado, presione enter para continuar"
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
fi
}
```

```
function buscar_grupo() {
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 1 ]; then
        echo "el grupo existe"
        cat /etc/group | grep $user_group
        echo "presione enter para continuar."
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
fi
}
```



```
function buscar_grupo() {
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_grupo=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_grupo)
    if [ $existe -eq 1 ]; then
        echo "el grupo existe"
        cat /etc/group | grep $user_grupo
        echo "presione enter para continuar."
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
fi
}
```

```
while [ $opc -ne 0 ]
do
    clear
    menu
    read -p "Ingrese la opción correspondiente: " opc
    case $opc in
        1)
            agregar_grupo;;
        2)
            borrar_grupo;;
        3)
            listar_grupos;;
        4)
            buscar_grupo;;
        0)
            echo "Volviendo al menú principal"; break ;;
        *)
            echo "Seleccionó una opción incorrecta";;
    esac
done
```



```
function buscar_grupo() {
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 1 ]; then
        echo "el grupo existe"
        cat /etc/group | grep $user_group
        echo "presione enter para continuar."
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
fi
}
```

```
while [ $opc -ne 0 ]
do
    clear
    menu
    read -p "Ingrese la opción correspondiente: " opc
    case $opc in
        1)
            agregar_grupo;;
        2)
            borrar_grupo;;
        3)
            listar_grupos;;
        4)
            buscar_grupo;;
        0)
            echo "Volviendo al menú principal"; break ;;
        *)
            echo "Seleccionó una opción incorrecta";;
    esac
done
```



Script de Redes

```
#!/bin/bash

# Función para mostrar el menú principal
show_menu() {
    choice=$(zenity --list --title="Administrador de Redes" --column="Acción" "Crear
Red" "Eliminar Red" "Editar Red" "Salir")

    case "$choice" in
        "Crear Red")
            create_network
            ;;
        "Eliminar Red")
            delete_network
            ;;
        "Editar Red")
            edit_network
            ;;
        "Salir")
            exit 0
            ;;
        *)
            zenity --error --text="Opción no válida."
            show_menu
            ;;
    esac
}
```



Función para crear una nueva red

```
create_network() {  
    network_name=$(zenity --entry --title="Crear Red" --text="Introduce el nombre de  
la nueva red:")  
    if [ -n "$network_name" ]; then  
        sudo nmcli connection add con-name "$network_name" ifname "*" type  
ethernet  
        zenity --info --text="Red '$network_name' creada con éxito."  
    fi  
    show_menu  
}
```

Función para eliminar una red existente

```
delete_network() {  
    network_name=$(zenity --entry --title="Eliminar Red" --text="Introduce el nombre  
de la red que deseas eliminar:")  
    if [ -n "$network_name" ]; then  
        sudo nmcli connection delete "$network_name"  
        zenity --info --text="Red '$network_name' eliminada con éxito."  
    fi  
    show_menu  
}
```

Función para editar una red existente

```
edit_network() {  
    network_name=$(zenity --entry --title="Editar Red" --text="Introduce el nombre de  
la red que deseas editar:")  
    if [ -n "$network_name" ]; then  
        nmcli connection edit "$network_name"  
        zenity --info --text="Editando la red '$network_name'. "  
    fi  
    show_menu}
```



Script de Servicios

```
#!/bin/bash

# Función para mostrar el menú principal
show_menu() {
    choice=$(zenity --list --title="Administrador de Servicios" --column="Servicio" "PHP"
"MySQL" "SSH" "Salir")

    case "$choice" in
        "PHP")
            toggle_service "php7.0-fpm" # Cambia php7.0-fpm al nombre del servicio de
PHP en tu sistema
            ;;
        "MySQL")
            toggle_service "mysql" # Cambia mysql al nombre del servicio de MySQL en
tu sistema
            ;;
        "SSH")
            toggle_service "ssh" # Cambia ssh al nombre del servicio de SSH en tu
sistema
            ;;
        "Salir")
            exit 0
            ;;
        *)
            zenity --error --text="Servicio no válido."
            show_menu
            ;;
    esac
}
```




Función para activar o desactivar un servicio

```
toggle_service() {  
    service_name="$1"  
    if systemctl is-active --quiet "$service_name"; then  
        systemctl stop "$service_name"  
        zenity --info --text="Servicio $service_name desactivado con éxito."  
    else  
        systemctl start "$service_name"  
        zenity --info --text="Servicio $service_name activado con éxito."  
    fi  
    show_menu  
}
```

Script de Firewall

```
#!/bin/bash
```

Función para mostrar el menú principal

```
show_menu() {  
    choice=$(zenity --list --title="Administrador de Firewall" --column="Acción" "Activar  
Firewall" "Desactivar Firewall" "Agregar Regla" "Eliminar Regla" "Salir")
```

```
    case "$choice" in  
        "Activar Firewall")  
            sudo iptables -F # Limpia todas las reglas existentes  
            sudo iptables -P INPUT DROP # Establece la política de entrada en DROP  
(denegar todo)  
            sudo iptables -P FORWARD DROP # Establece la política de reenvío en  
DROP  
            sudo iptables -P OUTPUT ACCEPT # Establece la política de salida en
```



ACCEPT (permitir todo)

```
zenity --info --text="Firewall activado con éxito."
```

```
show_menu
```

```
::
```

```
"Desactivar Firewall")
```

```
sudo iptables -F # Limpia todas las reglas existentes
```

```
sudo iptables -P INPUT ACCEPT # Establece la política de entrada en
```

ACCEPT (permitir todo)

```
sudo iptables -P FORWARD ACCEPT # Establece la política de reenvío en
```

ACCEPT

```
sudo iptables -P OUTPUT ACCEPT # Establece la política de salida en
```

ACCEPT (permitir todo)

```
zenity --info --text="Firewall desactivado con éxito."
```

```
show_menu
```

```
::
```

```
"Agregar Regla")
```

```
rule=$(zenity --entry --title="Agregar Regla" --text="Introduce una regla
```

iptables:")

```
if [ -n "$rule" ]; then
```

```
sudo iptables -A INPUT $rule
```

```
zenity --info --text="Regla agregada con éxito: $rule"
```

```
fi
```

```
show_menu
```

```
::
```

```
"Eliminar Regla")
```

```
rule=$(zenity --entry --title="Eliminar Regla" --text="Introduce la regla iptables
```

que deseas eliminar:")

```
if [ -n "$rule" ]; then
```

```
sudo iptables -D INPUT -j $rule
```

```
zenity --info --text="Regla eliminada con éxito: $rule"
```

```
fi
```

```
show_menu
```



```
;;  
;;  
"Salir")  
exit 0  
;;  
*)  
zenity --error --text="Opción no válida."  
show_menu  
;;  
esac  
}
```

Script de respaldo total

```
#!/bin/bash  
opc=0  
fecha=$(date +%Y%m%d-%H-%M-%S-backup_bd)  
  
function menu() {  
    echo "MENÚ DE GESTIÓN DE RESPALDOS"  
    echo "1 - Realizar un respaldo completo"  
    echo "2 - Respalda la estructura de la Base de datos"  
    echo "3 - Restaurar base de datos"  
    echo "4 - Realizar consulta personalizada"  
    echo "5 - Salir"  
}  
  
function realizar_respaldo_completo(){  
    echo "Ingrese el nombre de la base de datos"  
    read nomb  
    nombre=$(echo $nomb | tr '[:upper:]' '[:lower:]')
```



```
echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then
    mysqldump --opt --events --routines --triggers --default-character-set=utf8 -u root
-p $nombre > $nombre.sql
    tar -czvf $fecha.tar.gz $nombre.sql
    mv $fecha.tar.gz /root/respaldos/bd/completa/
    rm $nombre.sql
    rm bd.sql
else
    echo "no existe la base de datos, presione enter para continuar.."
    read enter
    rm bd.sql
fi
}
```

```
function realizar_respaldo_estructura(){
    echo "Ingrese el nombre de la base de datos"
    read nomb
    nombre=$(echo $nomb | tr '[:upper:]' '[:lower:]')
    echo "show databases" | mysql -u root -p > bd.sql
    existe=$(cat bd.sql | grep -c $nombre)
    if [ $existe -ge 1 ]
    then
        mysqldump -v --opt --no-data --default-character-set=utf8 -u root -p $nombre >
$nombre.sql
        tar -czvf $fecha.tar.gz $nombre.sql
        mv $fecha.tar.gz /root/respaldos/bd/estructura
        rm $nombre.sql
        rm bd.sql
    else
```



```
    echo "no existe la base de datos, presione enter para continuar.."
    read enter
    rm bd.sql
fi
}

function restaurar_basedatos(){

    echo "Ingrese el nombre de la base de datos"
    read nomb
    nombre=$(echo $nomb | tr '[:upper:]' '[:lower:]')
    echo "show databases" | mysql -u root -p > bd.sql
    existe=$(cat bd.sql | grep -c $nombre)
    if [ $existe -ge 1 ]
    then
        mysqldump -v --opt --no-data --default-character-set=utf8 -u root -p $nombre >
$nombre.sql
        tar -czvf $fecha.tar.gz $nombre.sql
        mv $fecha.tar.gz /root/respaldos/bd/estructura
        rm $nombre.sql
        rm bd.sql
    else
        echo "no existe la base de datos, presione enter para continuar.."
        read enter
        rm bd.sql
    fi
}

function pregunta_personalizada(){
    echo "Ingrese el nombre de la base de datos"
    read nomb
    nombre=$(echo $nomb | tr '[:upper:]' '[:lower:]')
```



```
echo "Ingrese la sintaxis correspondiente"
read $sentencia
echo "$sentencia" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then $sentencia
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

while [ $opc -ne 6 ]
do
menu
echo "Ingrese una opción: "
read opc
case $opc in
1) echo "realizar_respaldo_completo" ;;
2) echo "realizar_respaldo_estructura" ;;
3) echo "restaurar_basedatos" ;;
4) echo "realizar_respaldo_estructura" ;;
5) echo "Adios" ; break ;;
*) echo "Opción no valida" ; read pausa ;;
esac
done
```



Configuraciones de red en las terminales y el servidor.

Verificar la configuración actual

```
ip addr show
```

Editar el archivo de configuración de red para tu interfaz (por ejemplo, eth0)

```
sudo nano /etc/sysconfig/network-scripts/ifcfg-eth0
```

Dentro del archivo, configura la dirección IP estática, la máscara de subred, la puerta de enlace y los servidores DNS:

```
DEVICE=eth0
```

```
BOOTPROTO=static
```

```
ONBOOT=yes
```

```
IPADDR=192.168.1.2
```

```
NETMASK=255.255.255.0
```

```
GATEWAY=192.168.1.1
```

```
DNS1=8.8.8.8
```

Reiniciar el servicio de red

```
sudo systemctl restart network
```

Verificar la nueva configuración

```
ip addr show
```



Configuración del servicio SSH en el cliente y el servidor.

A continuación se mostrarán los comandos necesarios para realizar la configuración de un servidor SSH en Centos7

Paso 1) Instalar en el repositorio yum SSH

```
# yum install openssh-server
```

Paso 2) Habilitar y arrancar servicio SSH

```
# systemctl enable sshd  
# systemctl start sshd
```

Paso 3) Permitir el acceso en el firewall

```
# firewall-cmd --zone=public --permanent --add-port=22/tcp
```




```
[root@10 ~]# rpm -q openssh-server
openssh-server-7.4p1-22.el7_9.x86_64

[root@10 ~]# sudo firewall-cmd --permanent --add-service=ssh
Warning: ALREADY_ENABLED: ssh
success
```

[Archivos cuenta con rutinas de backup y sus correspondientes scripts para el administrador.](#)

```
#!/bin/bash
fecha=$(date +%Y%d%H%M%S)

mysqldump -u root -rkatsuisbo1234 katsuenterprise >
/home/kenterprise/ScriptCrontab/Respaldo_`$fecha`.sql
```

[Configuración del firewall de Gnu/Linux.](#)

```
# Verificar el estado del firewall
sudo firewall-cmd --state

# Habilitar el firewall (si no está activo)
sudo systemctl start firewalld
sudo systemctl enable firewalld

# Verificar las zonas de red activas
sudo firewall-cmd --get-active-zones

# Configurar reglas de firewall (ejemplo para SSH)
```



```
sudo firewall-cmd --zone=public --add-service=ssh --permanent
```

Recargar el firewall para aplicar cambios

```
sudo firewall-cmd --reload
```

Verificar reglas configuradas para una zona específica (ejemplo para la zona public)

```
sudo firewall-cmd --zone=public --list-all
```

Filtrado de Ips mediante Firewall.

```
#!/bin/bash
```

Función para mostrar el menú principal

```
show_menu() {  
    choice=$(zenity --list --title="Filtrar direcciones IP" --column="Acción" "Bloquear IP"  
"Permitir IP" "Listar Reglas" "Salir")
```

```
    case "$choice" in  
        "Bloquear IP")  
            block_ip  
            ;;  
        "Permitir IP")  
            allow_ip  
            ;;  
        "Listar Reglas")  
            list_rules  
            ;;  
        "Salir")  
            exit 0  
            ;;  
        *)  
            zenity --error --text="Opción no válida."  
            show_menu  
            ;;  
    esac
```

```
}
```

Función para bloquear una IP

```
block_ip() {
```



```
ip_to_block=$(zenity --entry --title="Bloquear IP" --text="Introduce la dirección IP que
deseas bloquear:")
if [ -n "$ip_to_block" ]; then
    sudo firewall-cmd --permanent --add-rich-rule='rule source ipset="$ip_to_block"
drop'
    sudo firewall-cmd --reload
    zenity --info --text="IP $ip_to_block bloqueada con éxito."
fi
show_menu
}

# Función para permitir una IP
allow_ip() {
    ip_to_allow=$(zenity --entry --title="Permitir IP" --text="Introduce la dirección IP que deseas
permitir:")
    if [ -n "$ip_to_allow" ]; then
        sudo firewall-cmd --permanent --add-rich-rule='rule source ipset="$ip_to_allow"
accept'
        sudo firewall-cmd --reload
        zenity --info --text="IP $ip_to_allow permitida con éxito."
    fi
    show_menu
}

# Función para listar las reglas de firewall
list_rules() {
    firewall-cmd --list-all
    zenity --info --text="Reglas de firewall listadas en la terminal."
    show_menu
}
```