Pasarela

Índice

Introducción

Este proyecto busca realizar una versión mejorada del funcionamiento actual del proxy en las clases de informática. Nos hemos basado en una infraestructura de red con 3 clases para la presente demostración.

Utilizando una APP web para poder gestionar los equipos que sean bloqueados, y además, permitiendo el acceso al CAMPUS con el fin de realizar por ejemplo exámenes, mientras se bloquea el uso de internet.

El funcionamiento está bastante fortificado contra errores, de modo que en caso de que algo salga mal, no se bloquee todo sin poder acceder.

Objetivos

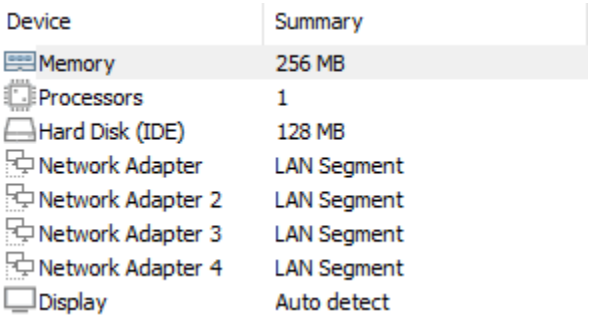
Como objetivo principal tenemos la creación de una APP que permita la gestión del proxy de manera muy sencilla e intuitiva, evitando así cualquier error humano que estropee las reglas de **iptables.**

Eso se va lograr interactuando con la APP, y esta a su vez, interactúa con las iptables mediante un script. Esto permitirá reducir considerablemente el fallo humano.

Herramientas

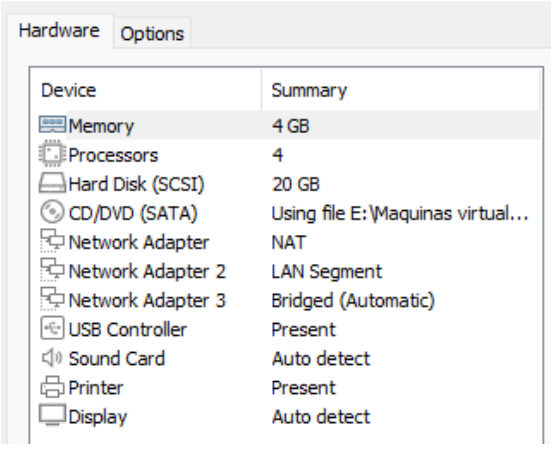
Durante el presente proyecto estaremos trabajando con 3 máquinas virtuales, cuyas propiedades son las siguientes:

- **Router Mikrotik ->** Este equipo tiene como propósito actuar de puente entre las redes de Clase 1-3 y nuestro Ubuntu Server. Dispone por tanto de 4 tarjetas de red, todas ellas Lan Segment. En términos de asignación de recursos, se está utilizando mediante CLI, dejando por tanto los recursos mínimos.



Device	Summary
Memory	256 MB
Processors	1
Hard Disk (IDE)	128 MB
Network Adapter	LAN Segment
Network Adapter 2	LAN Segment
Network Adapter 3	LAN Segment
Network Adapter 4	LAN Segment
Display	Auto detect

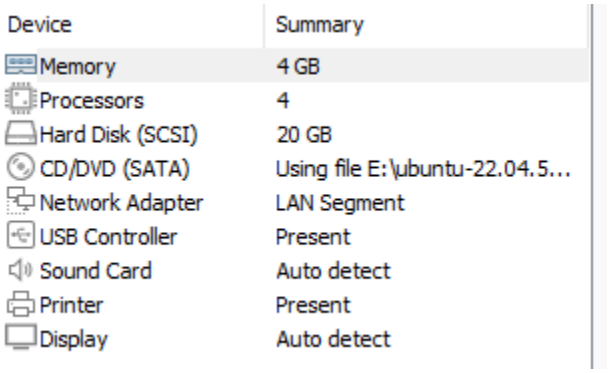
- **Ubuntu Server 22.04 ->** 3 adaptadores de red, siendo uno de ellos para el acceso al ISP, y el otro una interconexión con el router Mikrotik. Usará un disco de 20 GB, con 4 GB de memoria principal. Tendrá como propósito actual de salida a internet, y controlar las peticiones entrantes y salientes, es decir, cumplirá el papel de pasarela



Hardware | Options

Device	Summary
Memory	4 GB
Processors	4
Hard Disk (SCSI)	20 GB
CD/DVD (SATA)	Using file E:\Maquinas virtual...
Network Adapter	NAT
Network Adapter 2	LAN Segment
Network Adapter 3	Bridged (Automatic)
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

- **Cliente Ubuntu 24.04 ->** Esta máquina virtual será empleada para realizar pruebas, tales como acceso a internet, o reparto de direcciones IP por parte del servidor DHCP. Dispondrá de una única interfaz de red, LAN Segment, la cual cambiaremos entre clases para realizar pruebas con cada subred.



El software empleado es sencillo, pues, será solo lo previamente referenciado. Utilizaremos herramientas integradas en Ubuntu, cuya instalación separada no será necesaria, por ej. **iptables**.

Adicionalmente, se empleará Vscode para el desarrollo de la aplicación, empleando **JavaScript**

Desarrollo

---

1. Configuración Inicial R.Mikrotik

Acorde a lo mencionado en el apartado anterior, tendremos que configurar nuestro router Mikrotik para que disponga de Vlans, tenga las redes en la interfaz correspondiente, y haga reparto de direcciones IP.
Comenzaremos configurando las interfaces, asignando las VLANs.

```
[admin@MikroTik] > interface/vlan/print
Flags: R - RUNNING
Columns: NAME, MTU, ARP, VLAN-ID, INTERFACE
#    NAME      MTU  ARP      VLAN-ID  INTERFACE
0 R Clase 1   1500  enabled       10  ether7
1 R Clase 2   1500  enabled       20  ether6
2 R Clase 3   1500  enabled       30  ether5
3 R Trunk     1500  enabled       10  ether8
;;; Trunk
4 R vlan1     1500  enabled       20  ether8
;;; Trunk
5 R vlan2     1500  enabled       30  ether8
```

Evidentemente las Lan Segment de las interfaces se han comprobado previamente, corroborando con la MAC de cada una para que las VLANs sean las adecuadas.
Se dispone de 1 VLAN para cada clase, y 3 VLANs para ether8, que actuará de trunk.

A posteriori, asignamos IPs:

```
[admin@MikroTik] > ip/address/print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS           NETWORK     INTERFACE
0 10.69.0.254/24    10.69.0.0   ether8
1 10.69.10.1/24     10.69.10.0  ether7
2 10.69.20.1/24     10.69.20.0  ether6
3 10.69.30.1/24     10.69.30.0  ether5
```

Y, por último, configuramos las rutas pertinentes.

```
[admin@MikroTik] > ip/route/print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT, s - STATIC
Columns: DST-ADDRESS, GATEWAY, DISTANCE
#      DST-ADDRESS     GATEWAY     DISTANCE
0  As  0.0.0.0/0       10.69.0.1          1
   DAc 10.69.0.0/24    ether8             0
   DAc 10.69.10.0/24   ether7             0
   DAc 10.69.20.0/24   ether6             0
   DAc 10.69.30.0/24   ether5             0
```

2. Configuración Ubuntu Server

2.1 Netplan

```
network:
    ethernets:
        ens33:
            dhcp4: true
        ens34:
            dhcp4: false
            addresses:
              - 10.69.0.1/24
            routes:
             - to: 10.69.0.0/16
               via: 10.69.0.254
        ens35:
             dhcp4: false
             addresses:
               - 10.10.100.13/16
    version: 2
```

```
tfg_virtucan@virtucan:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:ab:10:fa brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.117.153/24 metric 100 brd 192.168.117.255 scope global dynamic ens33
       valid_lft 1088sec preferred_lft 1088sec
    inet6 fe80::20c:29ff:feab:10fa/64 scope link
       valid_lft forever preferred_lft forever
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:ab:10:04 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
    inet 10.69.0.1/24 brd 10.69.0.255 scope global ens34
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feab:1004/64 scope link
       valid_lft forever preferred_lft forever
4: ens35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:ab:10:0e brd ff:ff:ff:ff:ff:ff
    altname enp2s3
    inet 10.10.100.13/16 brd 10.10.255.255 scope global ens35
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feab:100e/64 scope link
       valid_lft forever preferred_lft forever
```

Y para comprobar que la configuración de red ha sido aplicada correctamente, hacemos ping del servidor a Internet, y del Mikrotik al servidor.

```
tfg_virtucan@virtucan:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=30.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=31.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=30.7 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=30.6 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 30.361/30.744/31.392/0.390 ms
tfg_virtucan@virtucan:~$ ping www.angelmelchor.pro
PING web.angelmelchor.pro (80.240.126.170) 56(84) bytes of data.
64 bytes from 80.240.126.170: icmp_seq=1 ttl=128 time=36.5 ms
64 bytes from 80.240.126.170: icmp_seq=2 ttl=128 time=36.3 ms
^C64 bytes from 80.240.126.170: icmp_seq=3 ttl=128 time=36.8 ms
```

```
[admin@MikroTik] > ping 10.69.0.1
  SEQ HOST                                     SIZE TTL TIME
    0 10.69.0.1                                  56  64 191us
    1 10.69.0.1                                  56  64 280us
    2 10.69.0.1                                  56  64 185us
    3 10.69.0.1                                  56  64 171us
    4 10.69.0.1                                  56  64 254us
```

Y como prueba adicional, trataremos de llegar desde el cliente ubuntu al servidor.

```
tfg_virtucan@tfg-virtucan:~$ ping 10.69.0.1
PING 10.69.0.1 (10.69.0.1) 56(84) bytes of data.
64 bytes from 10.69.0.1: icmp_seq=2 ttl=63 time=75.2 ms
64 bytes from 10.69.0.1: icmp_seq=3 ttl=63 time=55.2 ms
```

Hasta la configuración del DHCP el cliente Ubuntu dispone de una dirección estática asignada mediante el Netplan, concretamente la 10.69.10.69/24.

```
# Let NetworkManager manage all devices on this system
network:
  ethernets:
    ens33:
          dhcp4: false
          addresses:
            - 10.69.10.69/24
          routes:
           - to: default
             via: 10.69.10.1
          nameservers:
            addresses:
              - 8.8.8.8
              - 8.8.4.4
  version: 2
  renderer: NetworkManager
```

2.2 Iptables para acceso a internet

Tras confirmar que los parámetros de red son correctos, el siguiente paso es configurar iptables para que lleve a cabo el enmascaramiento, y tener así acceso a internet desde los clientes, así como el router Mikrotik.

Para ello necesitaremos tan solo 3 reglas. Una regla de **POSTROUTING**, y dos reglas **FORWARD**. Los comandos empleados son los siguientes:
- ***iptables -t nat -A POSTROUTING -s 10.69.0.0/16 -j MASQUERADE***
- ***iptables -A FORWARD -i ens33 -o ens34 -j ACCEPT***
- ***iptables -A FORWARD -o ens33 -i ens34 -j ACCEPT***

Comprobamos que se aplicaron correctamente:

```
tfg_virtucan@virtucan:~$ sudo iptables -t nat -L -v
Chain PREROUTING (policy ACCEPT 598 packets, 91910 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 462 packets, 80787 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 100 packets, 8569 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 100 packets, 8569 bytes)
 pkts bytes target     prot opt in     out     source               destination
  109  8115 MASQUERADE  all  --  any    any     10.69.0.0/16         anywhere
tfg_virtucan@virtucan:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 1299 packets, 1079K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 189 packets, 8316 bytes)
 pkts bytes target     prot opt in     out     source               destination
 248K 9958K ACCEPT     all  --  ens34  ens33   anywhere             anywhere
 128K 1827M ACCEPT     all  --  ens33  ens34   anywhere             anywhere

Chain OUTPUT (policy ACCEPT 521 packets, 44409 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Tras los cambios realizados, ya tenemos acceso a internet.

Cliente.

```
tfg_virtucan@tfg-virtucan:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=126 time=31.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=126 time=30.7 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 30.685/30.886/31.087/0.201 ms
tfg_virtucan@tfg-virtucan:~$ ping www.angelmelchor.pro
PING web.angelmelchor.pro (80.240.126.170) 56(84) bytes of data.
64 bytes from 764f93d8-2d3e-43a2-9577-4c427e4438ce.clouding.host (80.240.126
l=126 time=37.3 ms
64 bytes from 764f93d8-2d3e-43a2-9577-4c427e4438ce.clouding.host (80.240.126
l=126 time=37.0 ms
64 bytes from 764f93d8-2d3e-43a2-9577-4c427e4438ce.clouding.host (80.240.126
l=126 time=37.1 ms
```

Router.

```
[admin@MikroTik] > ip/dns/print
                   servers: 8.8.8.8,8.8.4.4
           dynamic-servers:
            use-doh-server:
```

```
[admin@MikroTik] > ping www.angelmelchor.pro
  SEQ HOST                                     SIZE TTL TIME       STATUS
    0 80.240.126.170                             56 127 36ms672us
    1 80.240.126.170                             56 127 35ms970us
    2 80.240.126.170                             56 127 36ms119us
    3 80.240.126.170                             56 127 62ms313us
    4 80.240.126.170                             56 127 36ms154us
    sent=5 received=5 packet-loss=0% min-rtt=35ms970us avg-rtt=41ms445us
   max-rtt=62ms313us

[admin@MikroTik] > ping 8.8.8.8
  SEQ HOST                                     SIZE TTL TIME       STATUS
    0 8.8.8.8                                    56 127 30ms376us
    1 8.8.8.8                                    56 127 30ms612us
    2 8.8.8.8                                    56 127 30ms793us
    sent=3 received=3 packet-loss=0% min-rtt=30ms376us avg-rtt=30ms593us
   max-rtt=30ms793us
```

3. DHCP en Router Mikrotik

Para este paso haremos uso de la opción "***setup***" que nos proporciona Mikrotik, facilitando en gran medida la creación de un servidor DHCP. Implementaremos un servidor DHCP en cada interfaz que conecta con una clase.

Ejemplo de setup con Clase 1.

```
[admin@MikroTik] > ip/dhcp-server/setup
Select interface to run DHCP server on

dhcp server interface: ether7
Select network for DHCP addresses

dhcp address space: 10.69.10.0/24
Select gateway for given network

gateway for dhcp network: 10.69.10.1
Select pool of ip addresses given out by DHCP server

addresses to give out: 10.69.10.10-10.69.10.69
Select DNS servers

dns servers: 8.8.8.8,8.8.4.4
Select lease time

lease time: 10000
```

Los parámetros finales del DHCP serán los siguientes:

```
[admin@MikroTik] > ip/dhcp-server/print
Columns: NAME, INTERFACE, ADDRESS-POOL, LEASE-TIME
# NAME   INTERFACE  ADDRESS-POOL  LEASE-TIME
0 dhcp1  ether7     dhcp_pool0    2h46m40s
1 dhcp2  ether6     dhcp_pool1    2h46m40s
2 dhcp3  ether5     dhcp_pool2    2h46m40s
[admin@MikroTik] > ip/dhcp-server/network/print
Columns: ADDRESS, GATEWAY, DNS-SERVER
# ADDRESS         GATEWAY     DNS-SERVER
0 10.69.10.0/24   10.69.10.1  8.8.8.8
                              8.8.4.4
1 10.69.20.0/24   10.69.20.1  8.8.8.8
                              8.8.4.4
2 10.69.30.0/24   10.69.30.1  8.8.8.8
                              8.8.4.4
```

```
[admin@MikroTik] /ip/dhcp-server> /ip pool print
Columns: NAME, RANGES
#  NAME         RANGES
0  dhcp_pool0   10.69.10.10-10.69.10.69
1  dhcp_pool1   10.69.20.10-10.69.20.69
2  dhcp_pool2   10.69.30.10-10.69.30.69
```

3.1 Pruebas de reparto de direcciones

Clase 1.

```
tfg_virtucan@tfg-virtucan:~$ sudo dhclient
tfg_virtucan@tfg-virtucan:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:f8:f6:dd brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 10.69.10.68/24 brd 10.69.10.255 scope global dynamic ens33
       valid_lft 9543sec preferred_lft 9543sec
    inet6 fe80::20c:29ff:fef8:f6dd/64 scope link
       valid_lft forever preferred_lft forever
```

```
    inet 10.69.10.68/24 brd 10.69.10.255 scope global dynamic noprefixr
3
       valid_lft 9993sec preferred_lft 9993sec
    inet6 fe80::20c:29ff:fef8:f6dd/64 scope link
       valid_lft forever preferred_lft forever
tfg_virtucan@tfg-virtucan:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=126 time=30.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=126 time=30.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=126 time=30.8 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 30.772/30.802/30.827/0.022 ms
```

```
[admin@MikroTik] /ip/dhcp-server> lease/print
Flags: D - DYNAMIC
Columns: ADDRESS, MAC-ADDRESS, HOST-NAME, SERVER, STATUS, LAST-SEEN
#   ADDRESS       MAC-ADDRESS        HOST-NAME     SERVER  STATUS    LAST-SEEN
0 D 10.69.10.69                                    dhcp1   conflict  2m36s
1 D 10.69.10.68   00:0C:29:F8:F6:DD  tfg-virtucan  dhcp1   bound     2m25s
```

Clase 2.

```
tfg_virtucan@tfg-virtucan:~$ sudo dhclient
tfg_virtucan@tfg-virtucan:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:f8:f6:dd brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 10.69.20.69/24 brd 10.69.20.255 scope global dynamic ens33
```

```
    inet 10.69.20.69/24 brd 10.69.20.255 scope global dynamic ens33
       valid_lft 9931sec preferred_lft 9931sec
    inet6 fe80::20c:29ff:fef8:f6dd/64 scope link
       valid_lft forever preferred_lft forever
tfg_virtucan@tfg-virtucan:~$ ping www.google.com
PING www.google.com (142.250.185.4) 56(84) bytes of data.
64 bytes from mad41s11-in-f4.1e100.net (142.250.185.4): icmp_seq=1 ttl=126 time=28.6 ms
64 bytes from mad41s11-in-f4.1e100.net (142.250.185.4): icmp_seq=2 ttl=126 time=28.1 ms
^C
--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
```

```
[admin@MikroTik] /ip/dhcp-server> lease/print
Flags: D - DYNAMIC
Columns: ADDRESS, MAC-ADDRESS, HOST-NAME, SERVER, STATUS, LAST-SEEN
#    ADDRESS       MAC-ADDRESS         HOST-NAME     SERVER  STATUS  LAST-SEEN
0 D  10.69.30.69   00:0C:29:F8:F6:DD   tfg-virtucan  dhcp3   bound   2m11s
1 D  10.69.20.69   00:0C:29:F8:F6:DD   tfg-virtucan  dhcp2   bound   10s
```

Clase 3.

```
tfg_virtucan@tfg-virtucan:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:f8:f6:dd brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 10.69.30.69/24 brd 10.69.30.255 scope global dynamic ens33
       valid_lft 9999sec preferred_lft 9999sec
    inet6 fe80::20c:29ff:fef8:f6dd/64 scope link
       valid_lft forever preferred_lft forever
tfg_virtucan@tfg-virtucan:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=126 time=31.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=126 time=31.8 ms
```

```
[admin@MikroTik] /ip/dhcp-server> lease/print
Flags: D - DYNAMIC
Columns: ADDRESS, MAC-ADDRESS, HOST-NAME, SERVER, STATUS, LAST-SEEN
#    ADDRESS       MAC-ADDRESS         HOST-NAME     SERVER  STATUS  LAST-SEEN
0 D  10.69.30.69   00:0C:29:F8:F6:DD   tfg-virtucan  dhcp3   bound   1m27s
```

4. Script bloqueo y desbloqueo de conexión

Para el presente proyecto se ha elaborado un script que cumple 5 funciones principales. Estas son:
- **block:** bloquea todas las salidas del cliente, sea a internet u otros hosts, dejando habilitado solo el protocolo icmp para así poder comprobar si el cliente sigue activo.

- **unblock:** elimina las restricciones anteriores.

- **campus-block:** con esta opción se realiza lo mismo que en el block normal, siendo la diferencia que se deja habilitado el acceso al campus, realizado mediante la obtención de las direcciones del dominio con dig, y su posterior ACCEPT en iptables. Se deja habilitado también el puerto udp para DNS, para que así el cliente no tenga que realizar la búsqueda mediante IP.

- **campus-unblock:** elimina las restricciones aplicadas por *campus-block.*

- **status y campus-status:** nos indica el estado del cliente, es decir, si está bloqueado o no.

Para emplear el script se invoca de la siguiente manera: **./sepia.sh "acción" "dirección IP"**

4.1 Comprobaciones de funciones

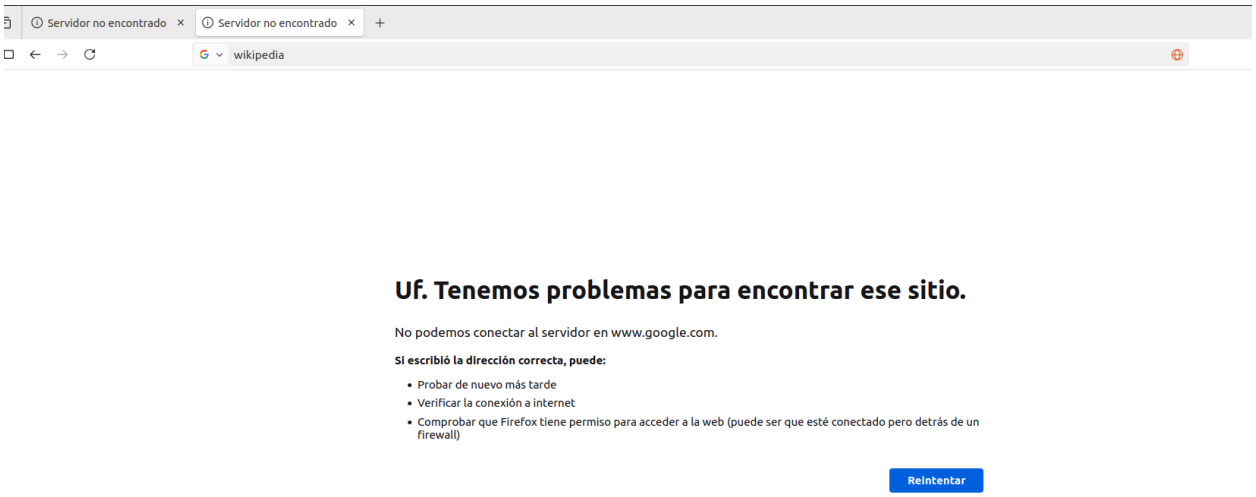
4.1.1 Block

```
root@virtucan:/home/tfg_virtucan# ./sepia.sh block 10.69.20.69
Bloqueo activado para 10.69.20.69
Reglas guardadas
root@virtucan:/home/tfg_virtucan# ./sepia.sh status 10.69.20.69
DROP  all opt -- in * out *  10.69.20.69  -> 0.0.0.0/0
Estado: 10.69.20.69 está bloqueada
root@virtucan:/home/tfg_virtucan# iptables -L -v
Chain INPUT (policy ACCEPT 4926 packets, 18M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     icmp --  any    any     10.69.20.69          anywhere
    0     0 DROP       all  --  any    any     10.69.20.69          anywhere
   75 13198 ACCEPT     all  --  ens34  ens33   anywhere             anywhere
   78 60732 ACCEPT     all  --  ens33  ens34   anywhere             anywhere

Chain OUTPUT (policy ACCEPT 2993 packets, 209K bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Vista del cliente.

```
tfg_virtucan@tfg-virtucan:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=126 time=31.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=126 time=31.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=126 time=30.5 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 30.542/30.927/31.181/0.277 ms
tfg_virtucan@tfg-virtucan:~$ curl www.google.com
^C
tfg_virtucan@tfg-virtucan:~$ ping 10.69.0.254
PING 10.69.0.254 (10.69.0.254) 56(84) bytes of data.
64 bytes from 10.69.0.254: icmp_seq=1 ttl=64 time=0.219 ms
64 bytes from 10.69.0.254: icmp_seq=2 ttl=64 time=0.723 ms
^C
--- 10.69.0.254 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.219/0.471/0.723/0.252 ms
```

Podemos observar que el cliente sigue pudiendo hacer ping, mientras que acciones como curl no funcionan, al no poder acceder a la página especificada. Además, si intentamos por ejemplo acceder a wikipedia mediante un navegador, este se queda cargando, mostrando al rato un **timeout.**



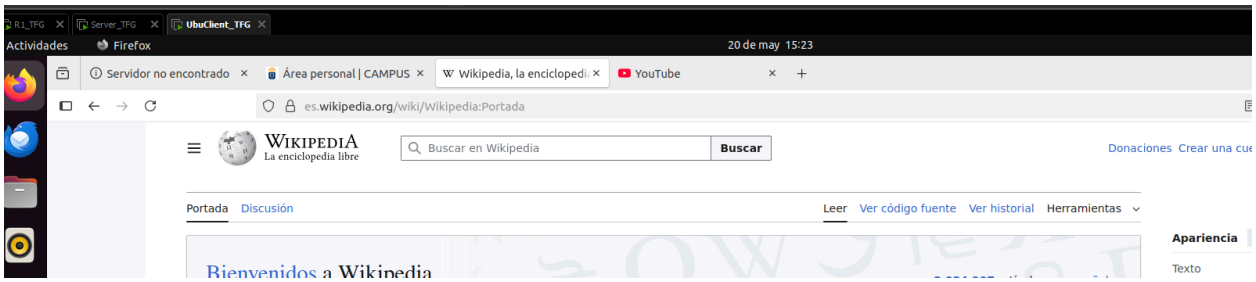
4.1.2 Unblock

```
root@virtucan:/home/tfg_virtucan# ./sepia.sh unblock 10.69.20.69
Bloqueo eliminado para 10.69.20.69
Reglas guardadas
root@virtucan:/home/tfg_virtucan# ./sepia.sh status 10.69.20.69
Estado: 10.69.20.69 no está bloqueada
root@virtucan:/home/tfg_virtucan# iptables -L -v
Chain INPUT (policy ACCEPT 5039 packets, 18M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  113 18280 ACCEPT     all  --  ens34  ens33   anywhere             anywhere
  119 68970 ACCEPT     all  --  ens33  ens34   anywhere             anywhere

Chain OUTPUT (policy ACCEPT 3057 packets, 216K bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Y ahora las páginas que antes no cargaban:

4.1.3 Campus-block

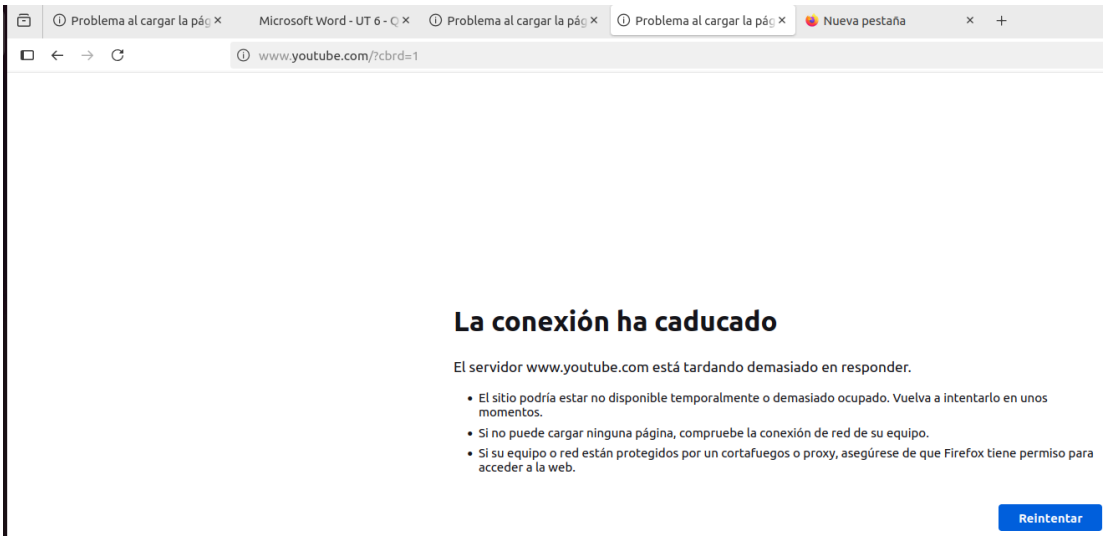
Antes de aplicar bloqueo en el cliente:



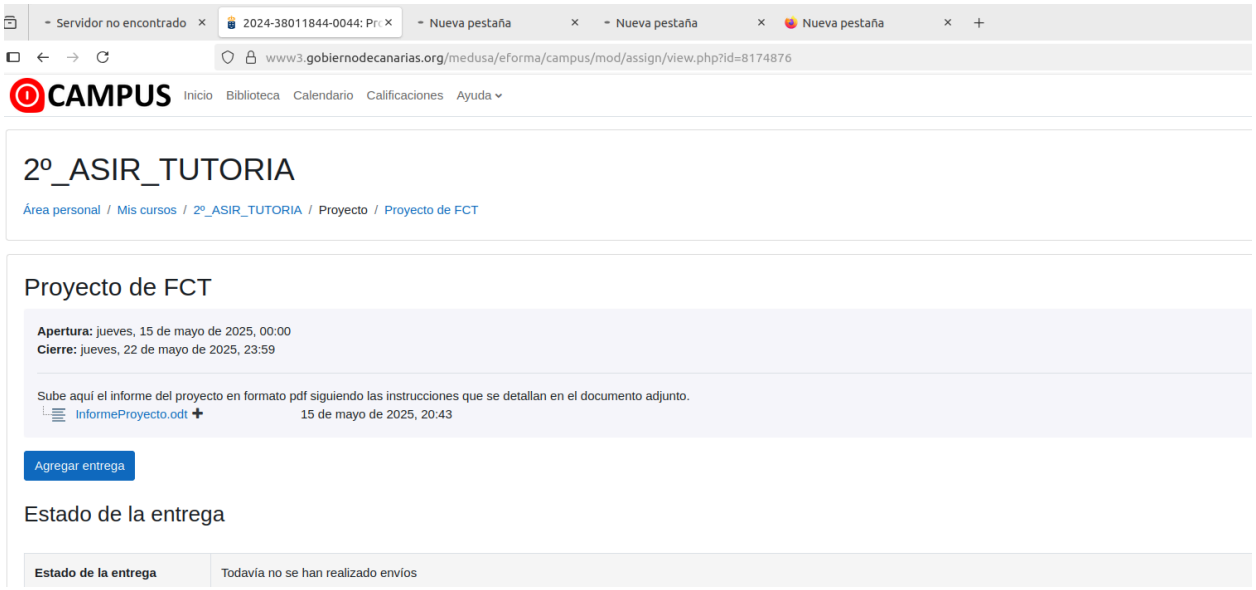
Bloqueo desde el servidor:

```
root@virtucan:/home/tfg_virtucan# ./sepia.sh campus-block 10.69.20.69
Resolviendo IPs para www3.gobiernodecanarias.org...
Modo campus activado para 10.69.20.69
Reglas guardadas
root@virtucan:/home/tfg_virtucan# ./sepia.sh campus-status 10.69.20.69
DROP  all opt -- in * out *  10.69.20.69  -> 0.0.0.0/0
Modo campus activado para 10.69.20.69
93.188.136.126/32
root@virtucan:/home/tfg_virtucan# iptables -L -v
Chain INPUT (policy ACCEPT 5429 packets, 18M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  any    any     10.69.20.69          93.188.136.126
    0     0 ACCEPT     udp  --  any    any     10.69.20.69          anywhere             udp dpt:domain
    0     0 ACCEPT     icmp --  any    any     10.69.20.69          anywhere
   25 16915 DROP       all  --  any    any     10.69.20.69          anywhere
10836 1178K ACCEPT     all  --  ens34  ens33   anywhere             anywhere
10026   66M ACCEPT     all  --  ens33  ens34   anywhere             anywhere
```

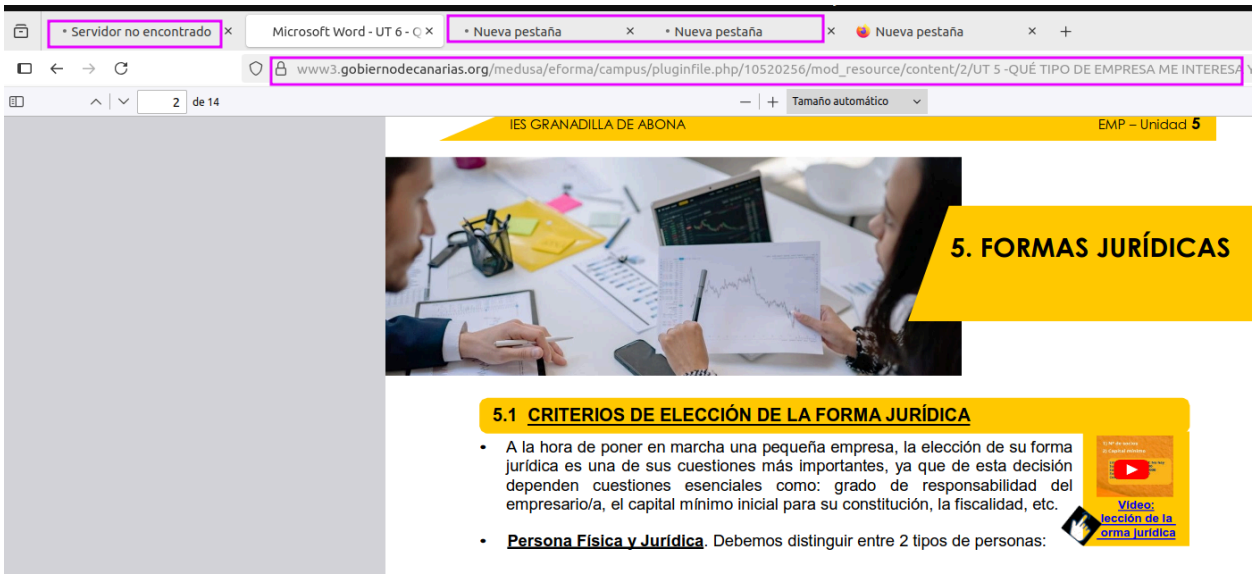
Resultado:



Mientras que el campus se ve de la siguiente manera:



Cargando incluso documentos con formato pdf, mientras que no se abandone el dominio del gobierno de canarias se puede navegar libremente. Otras webs, por su parte, no cargan de ninguna manera.



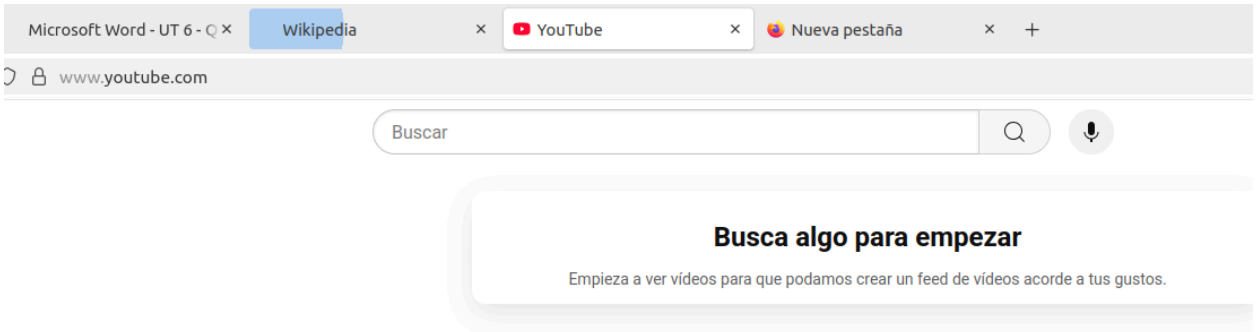
Ping y DNS.

```
tfg_virtucan@tfg-virtucan:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=126 time=31.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=126 time=30.7 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 30.687/30.938/31.190/0.251 ms
tfg_virtucan@tfg-virtucan:~$ ping www.angelmelchor.pro
PING web.angelmelchor.pro (80.240.126.170) 56(84) bytes of data.
64 bytes from 764f93d8-2d3e-43a2-9577-4c427e4438ce.clouding.host (80.240.126.170): icmp_seq=1 ttl=126 time=38.7 ms
64 bytes from 764f93d8-2d3e-43a2-9577-4c427e4438ce.clouding.host (80.240.126.170): icmp_seq=2 ttl=126 time=45.7 ms
^C
--- web.angelmelchor.pro ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 38.723/42.223/45.723/3.500 ms
tfg_virtucan@tfg-virtucan:~$ curl www.wikipedia.org
```

4.1.4 Campus-unblock

```
root@virtucan:/home/tfg_virtucan# ./sepia.sh campus-unblock 10.69.20.69
Modo campus desactivado para 10.69.20.69
Reglas guardadas
root@virtucan:/home/tfg_virtucan# ./sepia.sh campus-status 10.69.20.69
Modo campus no activado para 10.69.20.69
root@virtucan:/home/tfg_virtucan# iptables -L -v
Chain INPUT (policy ACCEPT 5581 packets, 18M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
10836 1178K ACCEPT     all  --  ens34  ens33   anywhere             anywhere
10606   67M ACCEPT     all  --  ens33  ens34   anywhere             anywhere

Chain OUTPUT (policy ACCEPT 3257 packets, 235K bytes)
 pkts bytes target     prot opt in     out     source               destination
root@virtucan:/home/tfg_virtucan#
```

Una vez eliminadas las restricciones, las páginas que previamente no cargaban se ven de la siguiente manera:



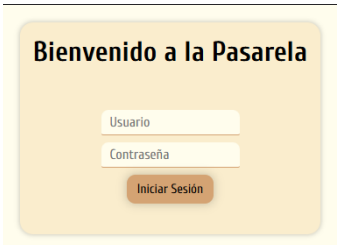
Y el curl de la siguiente:

En las anteriores instancias no hemos utilizado parámetros con el curl ya que era simplemente una demostración, sabiendo con certeza que no obtendríamos ningún dato, siendo por tanto irrelevante, ya que se queda cargando y pensando, lanzando un error independientemente de los parámetros empleados.

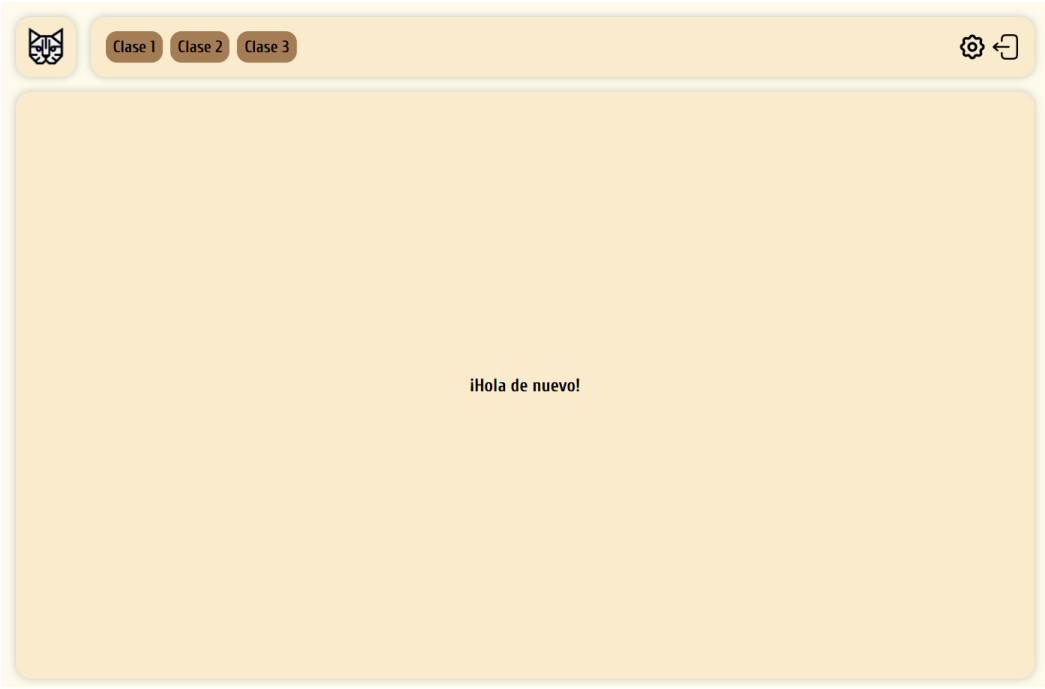
5. Aplicación Web

Para evitar un mal uso del script que permite modificar el proxy, todos los cambios se realizan desde una página web expuesta en el servidor Ubuntu. Se puede acceder desde cualquier clase permitiendo una gestión centralizada.

Además el funcionamiento de la web está protegido mediante el acceso por credenciales. Se utiliza MySQL para almacenar los datos de los usuarios y una caché con los equipos ya conocidos para agilizar el uso de la APP.



Una vez podamos acceder con nuestras credenciales correctas. Tendremos acceso al panel de control donde podremos gestionar el proxy.



La barra superior contiene botones que nos permiten acceder a las diferentes clases. Junto a dos botones de acción en la parte derecha. Uno nos permite acceder al menú de configuración de usuarios, que explicaremos más adelante; y el último nos permite cerrar sesión.

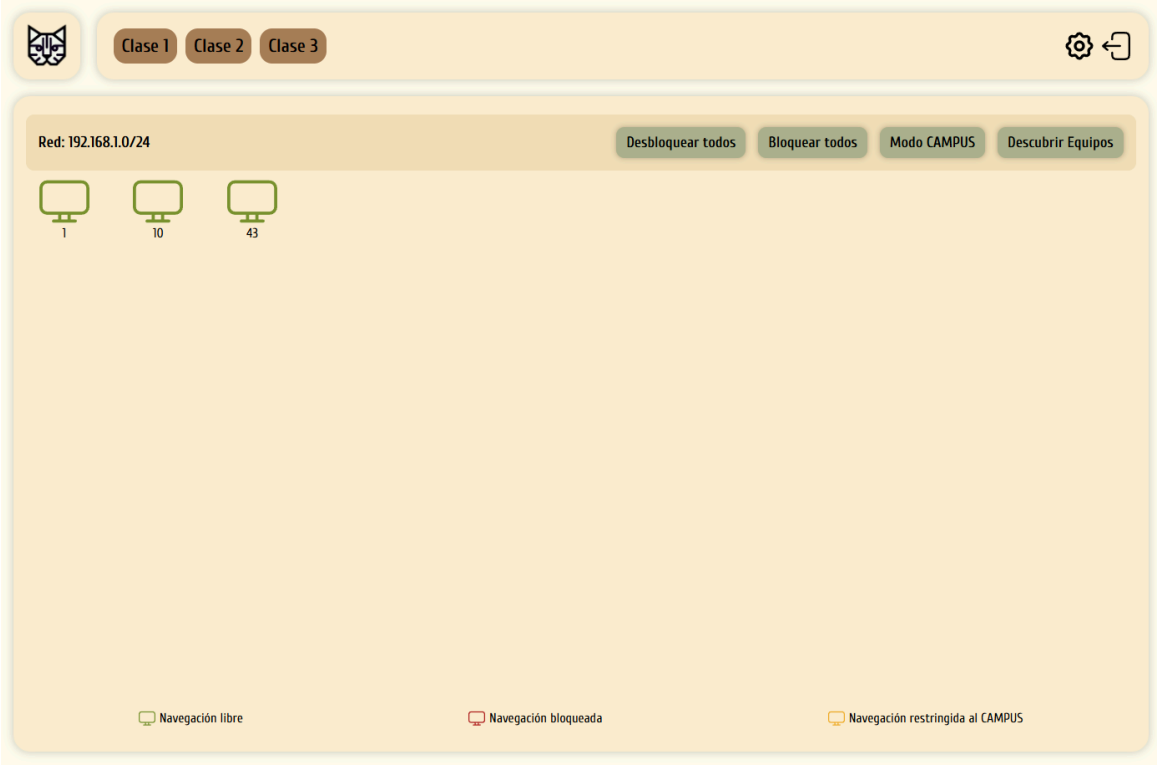
5.1 Visor de Equipos

Si hacemos click en una clase por ejemplo, **Clase 1**. Cambiará las vista, a una cuadrícula que contiene los equipos que están registrados en la base de datos.

En la nueva barra de acción que nos aparece, nos indica la Red IP sobre la que estamos trabajando. Y 4 botones de acción con las siguientes funciones:

- **Desbloquear todos y Bloquear todos:** Son bastante autodescriptivos, permiten bloquear o desbloquear a todos los equipos en la red.
- **Modo CAMPUS:** Bloquea a todos los equipos para que solo puedan navegar por el CAMPUS.
- **Descubrir equipos:** Este botón es el más importante ya que escanea la RED en búsqueda de equipos. Además también comprueba las normas que tiene el proxy aplicadas y corrige la información en la base datos si contiene incongruencias. Es importante tener en cuenta que escanear una red con máscara /24 tarda aproximadamente 9,67 segundos.

Además aparecen los equipos con un identificador haciendo alusión al último octeto de la dirección IP completa. En caso de no haber equipos registrados, sencillamente no aparece ninguno.

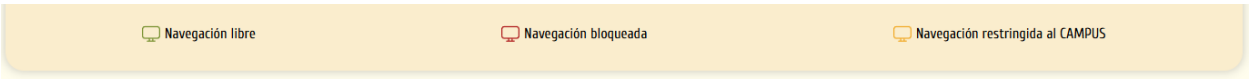


Al hacer clic sobre uno de los equipos se nos abre un menú desplegable. Con las opciones anteriormente explicadas pero afectando sólo a ese equipo. Además aparece la opción de **eliminar**, lo que permite eliminar el registro de la base de datos. Por ejemplo si es un equipo que ya no está en uso.

Los colores de los equipos simbolizan el estado en el que se encuentra dentro de las normas del proxy:

- **Verde:** Conexión libre.
- **Naranja:** Conexión exclusiva con el CAMPUS.
- **Rojo:** Conexión bloqueada de forma completa.

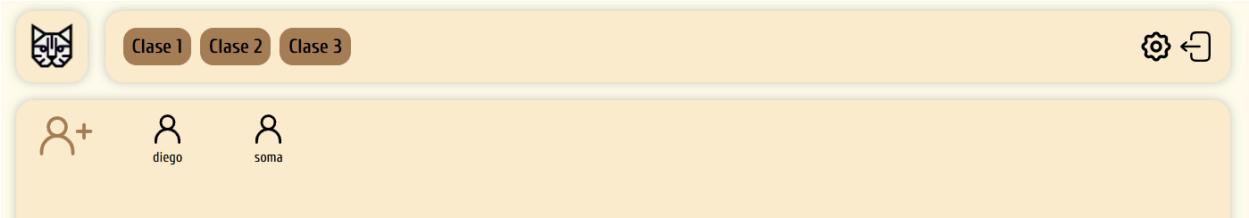
Esta leyenda se encuentra en la parte inferior de la cuadrícula.



5.2 Gestión de usuarios

La gestión de usuarios es sencilla puesto que la APP no requiere mucho en este aspecto más allá de garantizar el inicio de sesión correcto. El visor es similar al de los equipos.

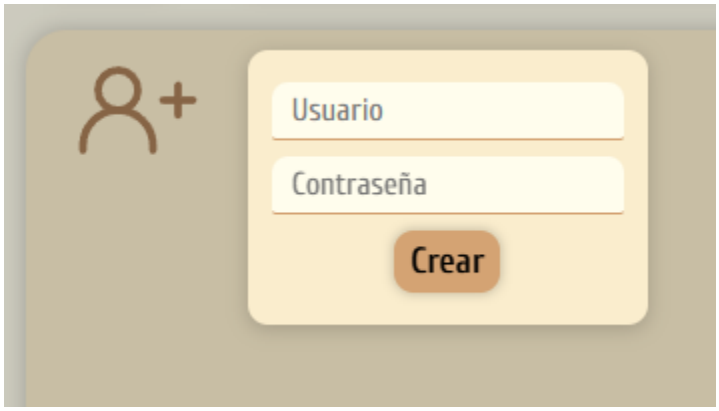
Todos los usuarios se representan mediante un icono y su nombre de usuario, el mismo con el que se inicia sesión.



5.2.1 Crear Usuarios

Para poder crear un nuevo usuario, debemos usar el botón que aparece primero y se distingue por tener un color diferente. Lo que nos desplegará un formulario donde rellenar los datos.

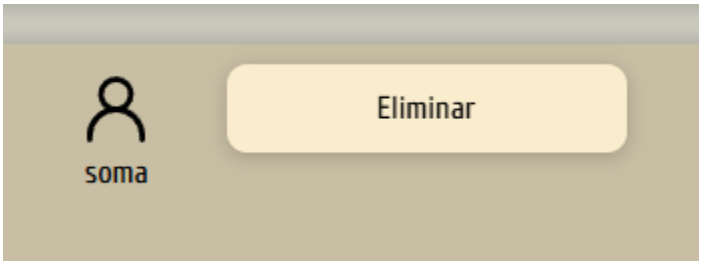
La única limitación que tenemos es que la contraseña tenga al menos, 5 caracteres.



5.2.2 Borrar un usuario

Si queremos borrar un usuario, debemos hacer clic en su icono y se desplegará un menú con la opción de eliminarlo. La aplicación tiene una medida de protección extra, ya que cuando queda un solo usuario no permitirá que sea borrado.

Aunque si por cualquier motivo la base de datos se queda sin usuarios la propia aplicación crea uno al arrancar siempre y cuando esté vacía. Los datos de esa cuenta por defecto se sacan del fichero **.env**.



5.3 Funcionamiento Interno de la APP

La aplicación está programada en javascript usando Nodejs como motor (versión 22.12.0). Además se utiliza el paquete pm2 que se instala usando npm. Para gestionar la app permitiendo que se ejecute en el arranque entre otras.

Las variables de entorno necesarias son las siguientes:

```
# APP
INT="0.0.0.0"
PORT="42068"
SALT="12"

# Database MySQL
DB_HOST="127.0.0.1"
DB_PORT="42069"
DB_USER="tfg"
DB_PASSWD="Passw0rd1"
DB_DB="pasarela"

# User
USERNAME="diego"
PASSWD="diego"

# JWT
JWT_SECRET="Buenas tardes a todos y todas,
```

5.3.1 Autenticación

La autenticación está hecha mediante JWT (Json Web Token) lo que libera al servidor de tener que almacenar los datos de sesión del cliente y consultarlos desde la base de datos cada vez permitiendo una autenticación sin estado.

Todos los endpoints de la APP están protegidos bajo sesión excepto aquellos que permiten iniciarla.

5.3.2 Base de Datos

Como base de datos estamos usando MySQL desde un contenedor de docker mediante **docker-compose**. Lo que permite tenerla aislada y levantarla sin tener que realizar instalaciones. La APP debería ir dentro de un contenedor pero como necesita interactuar con las **iptables** del servidor hemos decidido mantenerla en el propio servidor.

La estructura de la base de datos se maneja desde un archivo ubicado en el punto de entrada del contenedor de MySQL. Esta es una vista previa de ese fichero.

```sql
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id BINARY(16) PRIMARY KEY,
  username VARCHAR(30) UNIQUE,
  passwd VARCHAR(100)
);

DROP TABLE IF EXISTS classes;
CREATE TABLE classes (
  id INT PRIMARY KEY AUTO_INCREMENT,
  class VARCHAR(30),
  network VARCHAR(40) UNIQUE
);

DROP TABLE IF EXISTS hosts;
CREATE TABLE hosts (
  ip VARCHAR(24) PRIMARY KEY,
  class_id INT REFERENCES classes(id) ON DELETE CASCADE,
  blocked TINYINT(1) NOT NULL DEFAULT 0
);

INSERT INTO classes (class, network) VALUES
  ('Clase 1', '192.168.1.0/24'),
  ('Clase 2', '10.69.20.0/24'),
  ('Clase 3', '10.69.30.0/24');
```

6. Conclusión

Durante el desarrollo del proyecto nos hemos encontrado con múltiples problemas. La idea original era utilizar el proxy **Squid**, pero sin establecerlo en los clientes, es decir, que los alumnos no puedan saltarse la restricción de ninguna forma, y sin necesidad de configurar el proxy en el navegador, haciendo que este actuase como proxy transparente. Para lograrlo teníamos que redirigir el tráfico HTTP y HTTPS saliente hacia el proxy mediante iptables, sin embargo al realizar esto el tráfico HTTPS no estaba afectado ya que no se podía acceder a los encabezados ya que estaban encriptados.

La primera solución que barajamos fue el uso de **Sniproxy,** el cual aprovecha una parte de la petición HTTPS que viaja sin encriptar ya que contiene el dominio de destino para filtrar. Desde el principio no nos convencía puesto que era una especie de proxy inverso que redireccionaba el tráfico hacia un puerto cerrado del servidor para lograr un timeout. Además este sistema no es del todo fiable porque no siempre se incluye la cabecera **sni**.

La siguiente solución fue volver a **Squid**. Hay una opción adicional que se llama "***SSL Bump"***, que consiste en que el cliente forme una conexión segura con nuestro servidor y luego este último con el servidor destino. Básicamente consiste en replicar un ataque MITM para poder filtrar las conexiones. Comenzando por el hecho de que no es una práctica ética sin el consentimiento del usuario. En muchos equipos se requiere instalar el certificado SSL autofirmado de nuestro servidor en el cliente, lo que no nos sirve ya que nuestra idea principal es que el proxy debe actuar de forma transparente, y el usuario final no sepa que está pasando su tráfico a través de él.

Descartamos esta idea porque había que compilar de nuevo squid para activar esta función, además de no haber conseguido restablecer las conexiones SS, ya que el tráfico no salía correctamente hacia internet. Cabe destacar que si usabas el proxy de manera normal todo funcionaba bien. Así que en última instancia optamos por el uso de iptables.

Creamos un script que permitiese la gestión sencilla de las reglas de iptables para bloquear completamente el tráfico, solo permitiendo las conexión al CAMPUS o sin restricciones. Además el usuario final nunca utilizaría el script así que no corremos el riesgo de un mal uso, ni de tener usuarios en Linux con permisos de root para modificar las iptables.

Finalmente, la APP trabaja tanto con el script como con paquetes de escaneo de red como **fping** lo que permite que los tiempos de espera sean lo más cortos posibles, permitiendo las conexiones concurrentes y el multi-usuario, ya que varios usuarios pueden gestionar las clases al mismo tiempo.