



# REDES DE COMPUTADORAS

Trabajo Integrador

Grupo 09

RAFAEL DAVID NAZARENO RUIZ

ESTEBAN GASTON URANGA

KEVIN GABRIEL CONDORI CHAMBI

JOAQUIN ANTU BARCELLA

CRISTIAN MIGUEL OZIMO

## TRABAJO INTEGRADOR

### GRUPO 09

#### DESCRIPCIÓN DE LA RED

La topología representa una red de la facultad, donde se presentan distintos departamentos y oficinas, donde contamos con Ingeniería (VLAN 50), salud (VLAN 80), derecho (VLAN 70), sociales (VLAN 30), economía (VLAN 20), estudiantes (VLAN 40), CAU (VLAN 60) y gestión (VLAN 10) siendo este el encargado de la red mediante una PC de administración y el NETWORK CONTROLLER, para el cual se habilitó SSH y así poder controlar y monitorear la red.

Se implementaron los protocolos IPV4 y IPV6 de tipo classless por lo que la red es DUAL STACK. Implementamos ruteo dinámico mediante OSPFV2 y OSPFV3, también un servidor DNS y otro que cuenta con servicios de FTP, HTTP, y NTP.

Se tomó como base para el direccionamiento IPv4 para los hosts la red 172.16.0.0/16, mientras que para las redes punto a punto se eligió 10.1.0.0/16, como IP pública se tomó la 73.73.10.0/24, para el DNS se utilizó la red 8.8.8.0/24. Además, se utilizó la IP 100.100.100.10/27 para el Servidor HTTP, FTP, SMTP y POP.

Para IPv6 se tomó como base para los hosts la IP 2024:11:15::/48, mientras que para las redes punto a punto se tomó como base la IP 2024::/48. Como base para las redes del Servidor DNS se tomó la IP 3001:20:10:5::2/64 y la IP 3001:20:10:5::1/64 para el Servidor HTTP, FTP, SMTP y POP.

Contamos con conexiones redundantes para mejorar la seguridad en caso de que falle alguna.

Finalmente, para acceder a internet, la red emplea el router R1 como router de borde, estableciendo comunicación con el ISP a través de una ruta por defecto. Así, cualquier dirección que R1 no conozca es enviada a través de la interfaz conectada al ISP.

## TOPOLOGÍA:

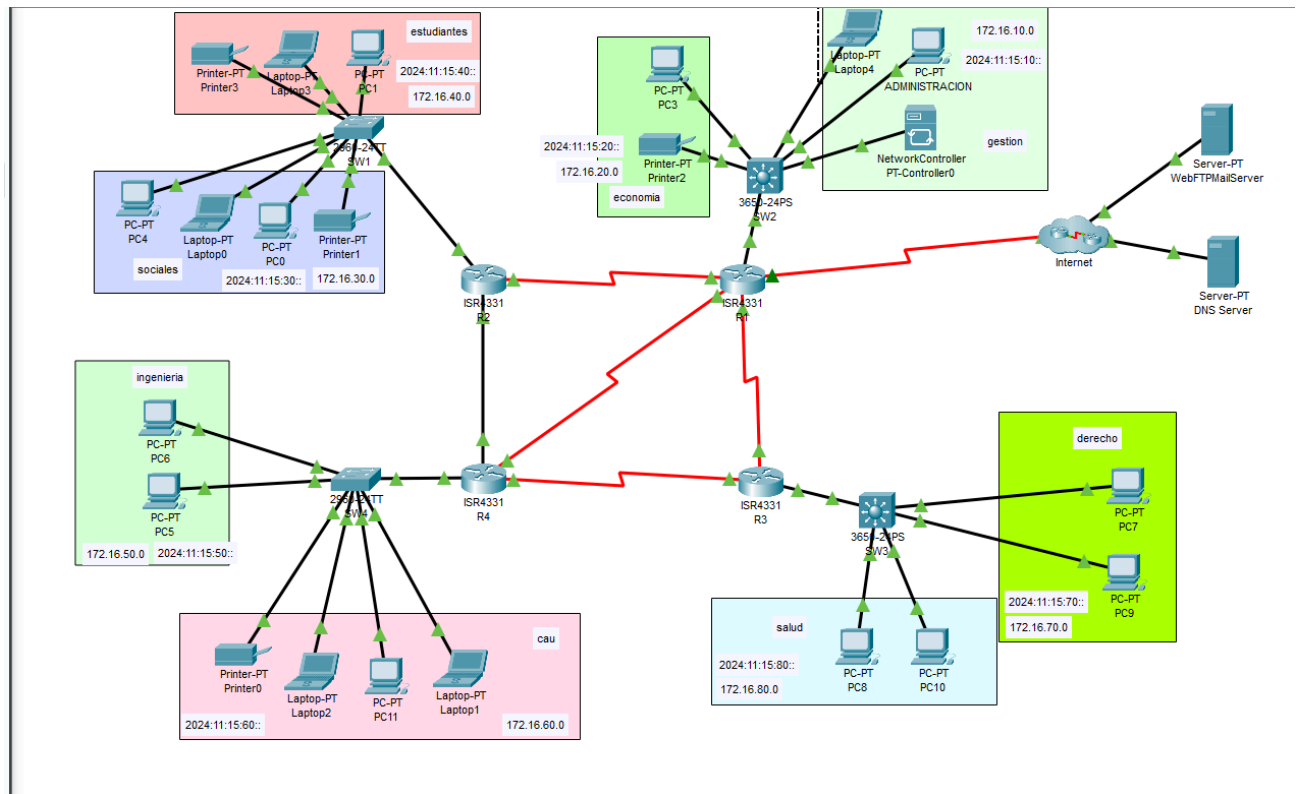


TABLA DE DIRECCIONES IPv4

Dispositivo	Interfaz	Dirección IP	Máscara de Subred	DG	DNS
R1	s0/1/1	73.73.10.2	/30		
R1	s0/1/0	10.1.1.5	/30		
R1	s0/2/0	10.1.1.9	/30		
R1	s0/2/1	10.1.1.13	/30		
R1	g0/0/0	10.1.1.2	/30		
R2	g0/0/1.30	172.16.30.254	/24		
R2	g0/0/1.40	172.16.40.254	/30		
R2	g0/0/0	10.1.1.17	/30		
R2	s0/1/0	10.1.1.6	/30		
R3	g0/0/0	10.1.1.25	/30		
R3	s0/1/0	10.1.1.14	/30		
R3	s0/1/1	10.1.1.22	/30		
R4	g0/0/0.50	172.16.50.254	/24		
R4	g0/0/0.60	172.16.60.254	/24		
R4	g0/0/1	10.1.1.18	/30		
R4	s0/1/0	10.1.1.10	/30		
R4	s0/1/1	10.1.1.21	/30		
SW2	vlan10	172.16.10.254	/24		
SW2	vlan20	172.16.20.254	/24		
SW2	g1/0/24	10.1.1.1	/30		
SW3	vlan70	172.16.70.254	/24		
SW3	vlan80	172.16.80.254	/24		
SW3	g1/0/24	10.1.1.26	/30		
PC0	Fa0	172.16.30.2	/24	172.16.30.254	8.8.8.8
PC1	Fa0	172.16.40.1	/24	172.16.40.254	8.8.8.8
PC3	Fa0	172.16.20.1	/24	172.16.20.254	8.8.8.8
PC4	Fa0	172.16.30.4	/24	172.16.30.254	8.8.8.8
PC5	Fa0	172.16.50.55	/24	172.16.50.254	8.8.8.8
PC6	Fa0	172.16.50.56	/24	172.16.50.254	8.8.8.8
PC7	Fa0	172.16.70.30	/24	172.16.70.254	8.8.8.8
PC8	Fa0	172.16.80.32	/24	172.16.80.254	8.8.8.8
PC9	Fa0	172.16.70.21	/24	172.16.70.254	8.8.8.8
PC10	Fa0	172.16.80.25	/24	172.16.80.254	8.8.8.8
Laptop0	Fa0	172.16.30.3	/24	172.16.30.254	8.8.8.8
Laptop1	Fa0	172.16.60.59	/24	172.16.60.254	8.8.8.8
Laptop2	Fa0	172.16.60.57	/24	172.16.60.254	8.8.8.8
Laptop3	Fa0	172.16.40.2	/24	172.16.40.254	8.8.8.8
Laptop4	Fa0	172.16.10.2	/24	172.16.10.254	8.8.8.8
PC-ADMIN	Fa0	172.16.10.1	/24	172.16.10.254	8.8.8.8
PC11	Fa0	172.16.60.58	/24	172.16.60.254	8.8.8.8
ISP	g0/0/0	100.100.100.1	/27		

ISP	g0/0/1	8.8.8.1	/24		
ISP	s0/1/0	73.73.10.1	/30		
WebFTPMailServer	Fa0	100.100.100.10	/27		
DNS Server	Fa0	8.8.8.8	/24		

TABLA DE DIRECCIONES IPv6

Dispositivo	Interfaz	Dirección	Link	Máscara	DG2
R1	s0/1/1	2024::14	fe80::	/127	
R1	s0/1/0	2024::2	fe80::	/127	
R1	s0/2/0	2024::4	fe80::	/127	
R1	s0/2/1	2024::6	fe80::	/127	
R1	g0/0/0	2024::1	fe80::1	/127	
R2	g0/0/1.30	2024:11:15:30::	fe80::	/64	
R2	g0/0/1.40	2024:11:15:40::	fe80::	/64	
R2	g0/0/0	2024::8	fe80::	/127	
R2	s0/1/0	2024::3	fe80::1	/127	
R3	g0/0/0	2024::12	fe80::	/127	
R3	s0/1/0	2024::7	fe80::1	/127	
R3	s0/1/1	2024::11	fe80::1	/127	
R4	g0/0/0.50	2024:11:15:50::	fe80::	/64	
R4	g0/0/0.60	2024:11:15:60::	fe80::	/64	
R4	g0/0/1	2024::9	fe80::1	/127	
R4	s0/1/0	2024::5	fe80::1	/127	
R4	s0/1/1	2024::10	fe80::	/127	
SW2	vlan10	2024:11:15:10::	fe80::	/64	
SW2	vlan20	2024:11:15:20::	fe80::	/64	
SW2	g1/0/24	2024::	fe80::	/127	
SW3	vlan70	2024:11:15:70::	fe80::	/64	
SW3	vlan80	2024:11:15:80::	fe80::	/64	
SW3	g1/0/24	2024::13	fe80::1	/127	
PC0	Fa0	2024:11:15:30::1			fe80::
PC1	Fa0	2024:11:15:40:2E0:A3FF:FED2:5ED	/64		fe80::
PC3	Fa0	2024:11:15:20:205:5EFF:FEB9:A082	/64		fe80::
PC4	Fa0	2024:11:15:30:205:5EFF:FEE7:E03A	/64		fe80::
PC5	Fa0	2024:11:15:50:230:A3FF:FE56:6AE3	/64		fe80::
PC6	Fa0	2024:11:15:50:2E0:F9FF:FEC9:6EBD	/64		fe80::
PC7	Fa0	2024:11:15:70::2	/64		fe80::
PC8	Fa0	2024:11:15:80::2	/64		fe80::
PC9	Fa0	2024:11:15:70::1	/64		fe80::
PC10	Fa0	2024:11:15:80::1	/64		fe80::
PC11	Fa0	2024:11:15:60:290:CFF:FE10:6BBD	/64		fe80::
Laptop0	Fa0	2024:11:15:30::3	/64		fe80::

<b>Laptop1</b>	Fa0	2024:11:15:60:2D0:97FF:FEAE:545C	/64		fe80::
<b>Laptop2</b>	Fa0	2024:11:15:60:290:21FF:FE14:7232	/64		fe80::
<b>Laptop3</b>	Fa0	2024:11:15:40:20B:BEFF:FEC7:154D	/64		fe80::
<b>Laptop4</b>	Fa0	2024:11:15:10:210:11FF:FE9B:4203	/64		fe80::
<b>PC-ADMIN</b>	Fa0	2024:11:15:10:2D0:BAFF:FEDE:26	/64		fe80::
<b>ISP</b>	g0/0/0	3001:20:10:5::	fe80::	/127	
<b>ISP</b>	g0/0/1	3001:20:10:5::3	fe80::	/127	
<b>ISP</b>	s0/1/0	2024::15	fe80::1	/127	
<b>WebFTPMailServer</b>	Fa0	3001:20:10:5::1		/127	fe80::
<b>DNS Server</b>	Fa0	3001:20:10:5::2		/127	fe80::

## ENRUTAMIENTO

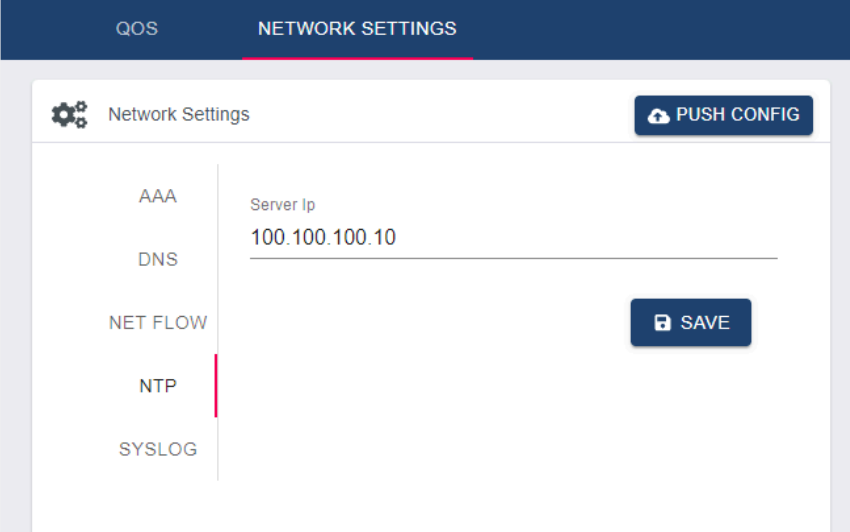
Se utilizó el protocolo de enrutamiento dinámico OSPF V2 y V3 para IPv4 e IPv6, respectivamente. Esto se debe a que OSPF es compatible con direcciones IP classless y ofrece una menor distancia administrativa en comparación con otros protocolos comunes como RIP, lo que garantiza mayor confiabilidad y eficiencia en la selección de rutas. Asimismo, a diferencia de RIPv1, admite el uso de VLSM y CIDR.

La configuración se llevó a cabo utilizando una única área 0 (backbone), recomendada para redes pequeñas como esta, compuesta por los dispositivos R1, R2, R3, R4, SW2 y SW3. Esto se realizó configurando en los routers y switches de capa tres, un router ospf, agregando las redes conectadas a cada nodo interno, pasivando las interfaces correspondientes y en particular en el router de borde R1, se utilizó el comando *default-information originate* para propagar la ruta por defecto.

# SERVICIOS

## NTP

NTP (Network Time Protocol) es un protocolo utilizado para sincronizar el reloj de los dispositivos en una red. Su objetivo principal es garantizar que todos los dispositivos de la red tengan la misma hora, con una precisión de milisegundos en redes locales y de decenas de milisegundos en redes más grandes.

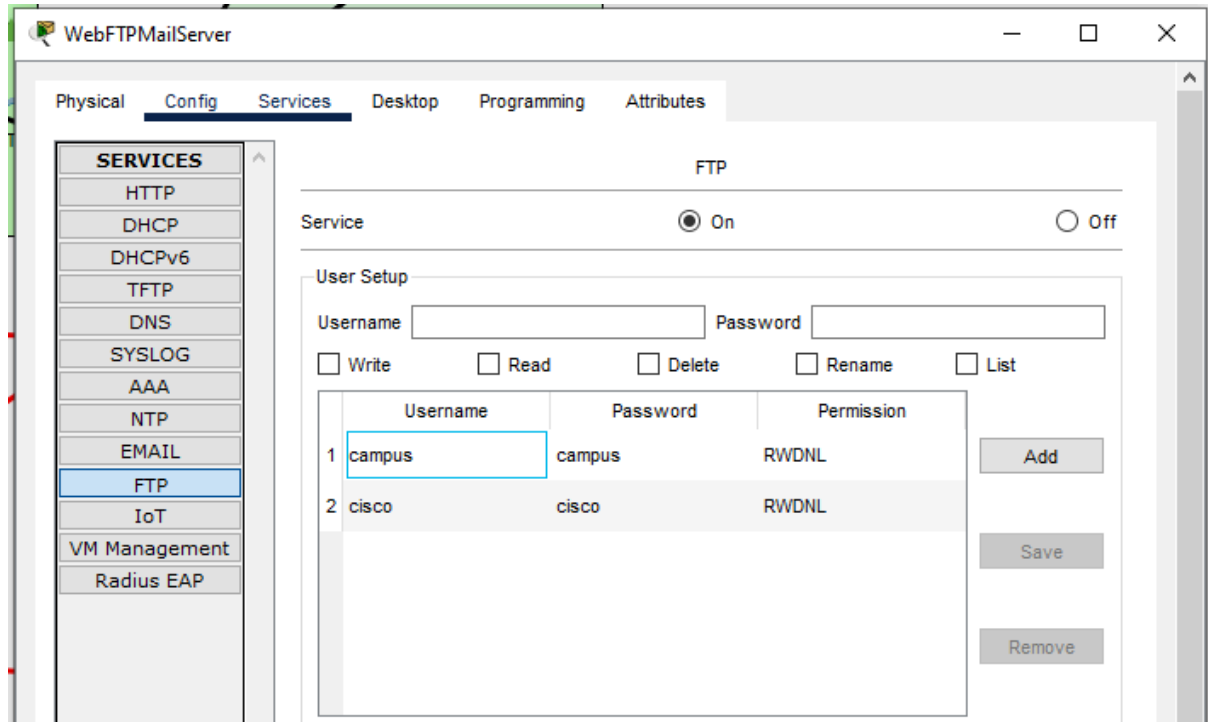


The screenshot displays a web interface for network configuration. At the top, there is a dark blue header with two tabs: 'QOS' and 'NETWORK SETTINGS', with the latter being the active tab. Below the header, the main content area is titled 'Network Settings' and includes a 'PUSH CONFIG' button. A left sidebar lists several configuration categories: AAA, DNS, NET FLOW, NTP (which is highlighted with a red vertical line), and SYSLOG. The main panel for the NTP section shows a 'Server Ip' field with the value '100.100.100.10' entered. A 'SAVE' button is located at the bottom right of this section.

Habilitamos NTP desde el network controller y agregamos la IP previamente configurada en el servidor con dicho servicio. Luego realizamos un push al resto de la red.

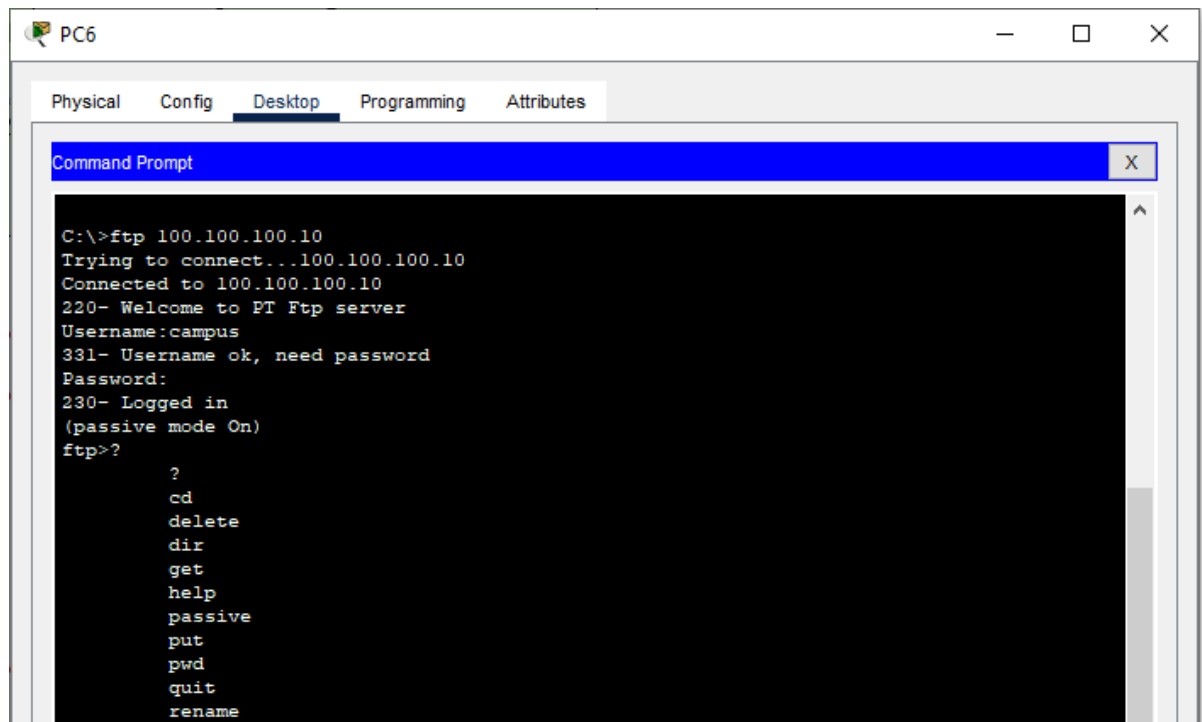
## FTP

Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor.



Dentro de la interfaz podremos generar usuarios y otorgarles los respectivos permisos como escritura, lectura, etc.





The image shows a screenshot of the PC6 software interface. At the top, there is a window titled 'PC6' with standard Windows window controls (minimize, maximize, close). Below the title bar, there are five tabs: 'Physical', 'Config', 'Desktop', 'Programming', and 'Attributes'. The 'Desktop' tab is currently selected. Within the 'Desktop' tab, a 'Command Prompt' window is open, displaying the following text:

```
C:\>ftp 100.100.100.10
Trying to connect...100.100.100.10
Connected to 100.100.100.10
220- Welcome to PT Ftp server
Username:campus
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>?
```

Below the 'ftp>?' prompt, a list of available FTP commands is displayed:

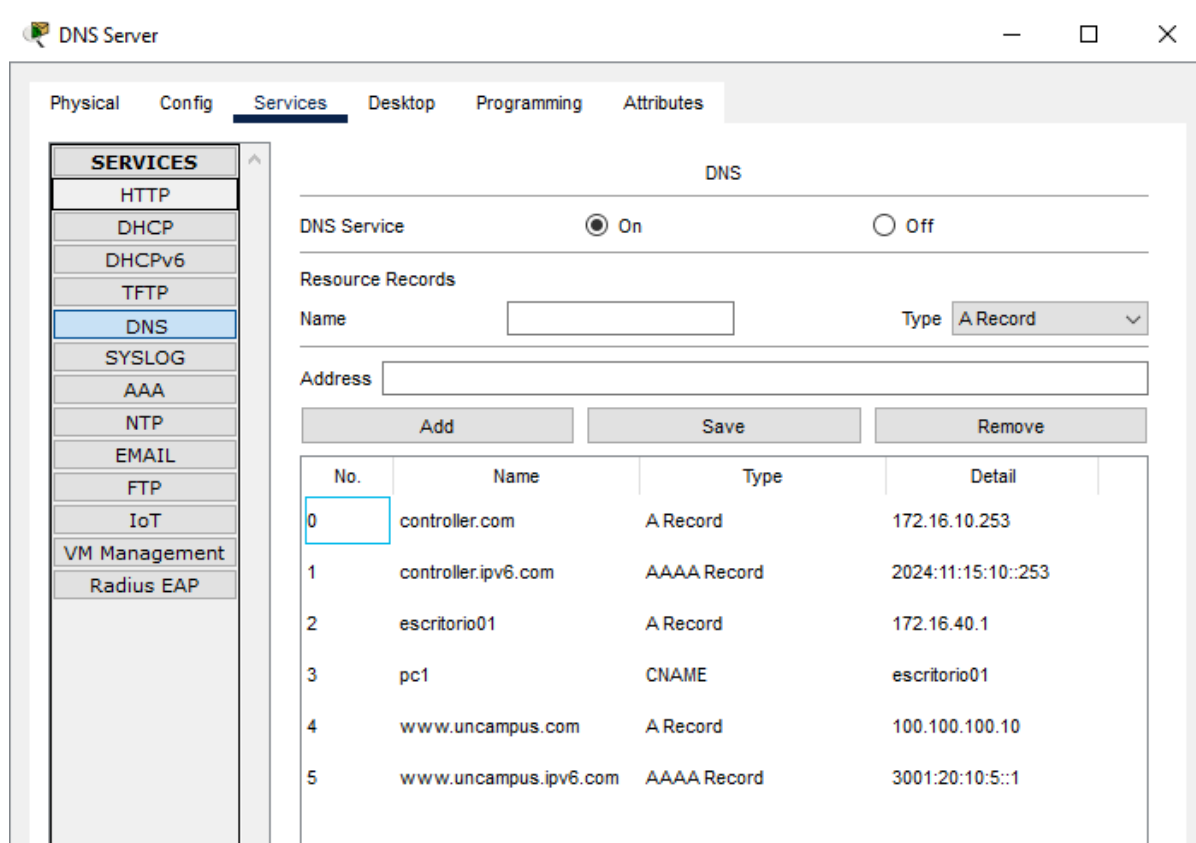
- ?
cd
delete
dir
get
help
passive
put
pwd
quit
rename

Ejemplo desde PC6 donde observamos los distintos comandos que podemos utilizar.

## DNS

DNS (Domain Name System) es un sistema que se utiliza para resolver nombres de dominio en direcciones IP, lo que facilita la navegación por Internet. En lugar de recordar direcciones IP numéricas de los sitios web, los usuarios pueden acceder a estos sitios utilizando nombres más fáciles de recordar.

En este caso contamos con un servidor dedicado a dicho servicio.



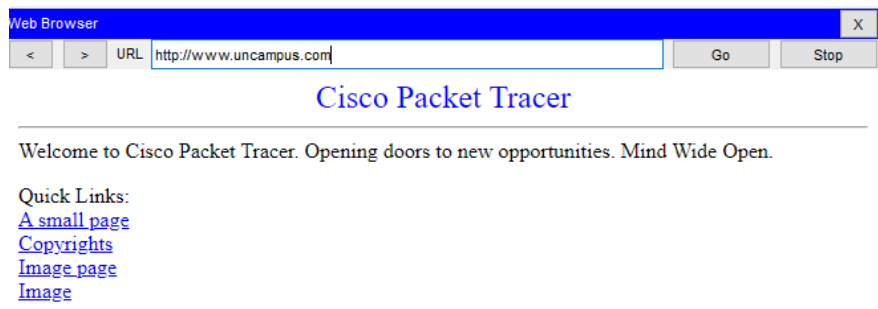
Se diferencian distintos tipos de registros, como lo son:

**A RECORD:** se utiliza para asociar un nombre de dominio con una dirección IPv4.

**AAAA RECORD:** utilizado para asociar un nombre de dominio con una dirección IPv6.

**CNAME:** se usa para alias de un dominio. Permite que un dominio apunte a otro dominio en lugar de a una dirección IP directa.

Prueba:



## DHCP

DHCP (Dynamic Host Configuration Protocol) es un protocolo de gestión de red que se configura para asignación dinámica de direcciones IP a hosts que se conecten a una red que emplee este protocolo. Permite configurar un rango de IPs para utilizar y permitiendo excluir aquellas que no puedan ser asignadas dentro del rango especificado. En nuestro caso, este servicio se configuró en el Router R2 y en el Switch de capa tres SW2, como se puede apreciar en las siguientes imágenes.

Configuración R2:

R2

Physical

Config

CLI

Attributes

IOS Command Line Interface

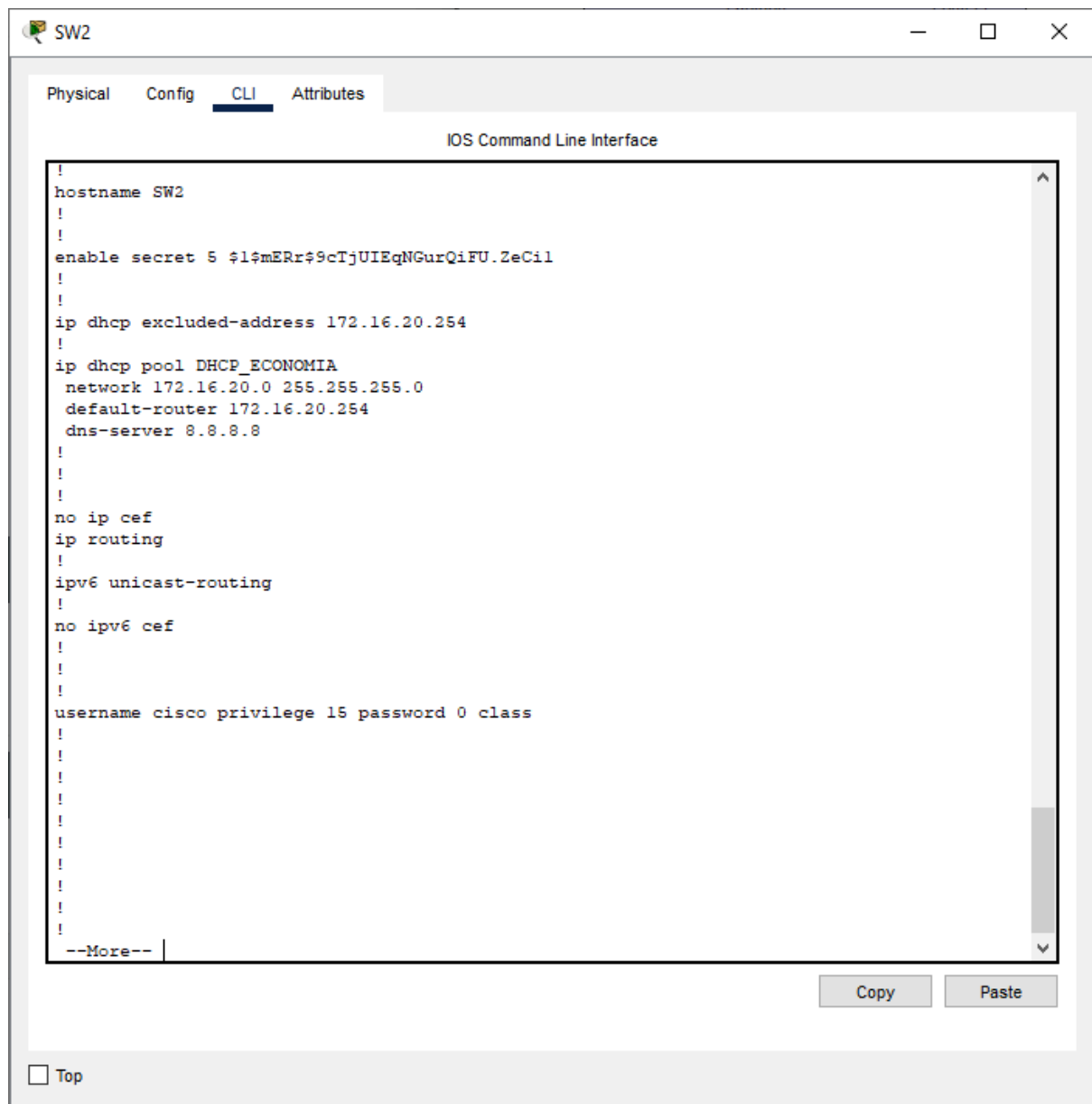
```
!
!
ip dhcp excluded-address 172.16.30.254
ip dhcp excluded-address 172.16.40.254
!
ip dhcp pool DHCP_SOCIALES
network 172.16.30.0 255.255.255.0
default-router 172.16.30.254
dns-server 8.8.8.8
ip dhcp pool DHCP_ESTUDIANTES
network 172.16.40.0 255.255.255.0
default-router 172.16.40.254
dns-server 8.8.8.8
!
!
!
no ip cef
ipv6 unicast-routing
!
no ipv6 cef
!
!
!
username cisco privilege 15 password 0 class
!
!
!
!
!
!
!
ip ssh version 2
no ip domain-lookup
ip domain-name uncampus.com
!
```

Copy

Paste

☐ Top

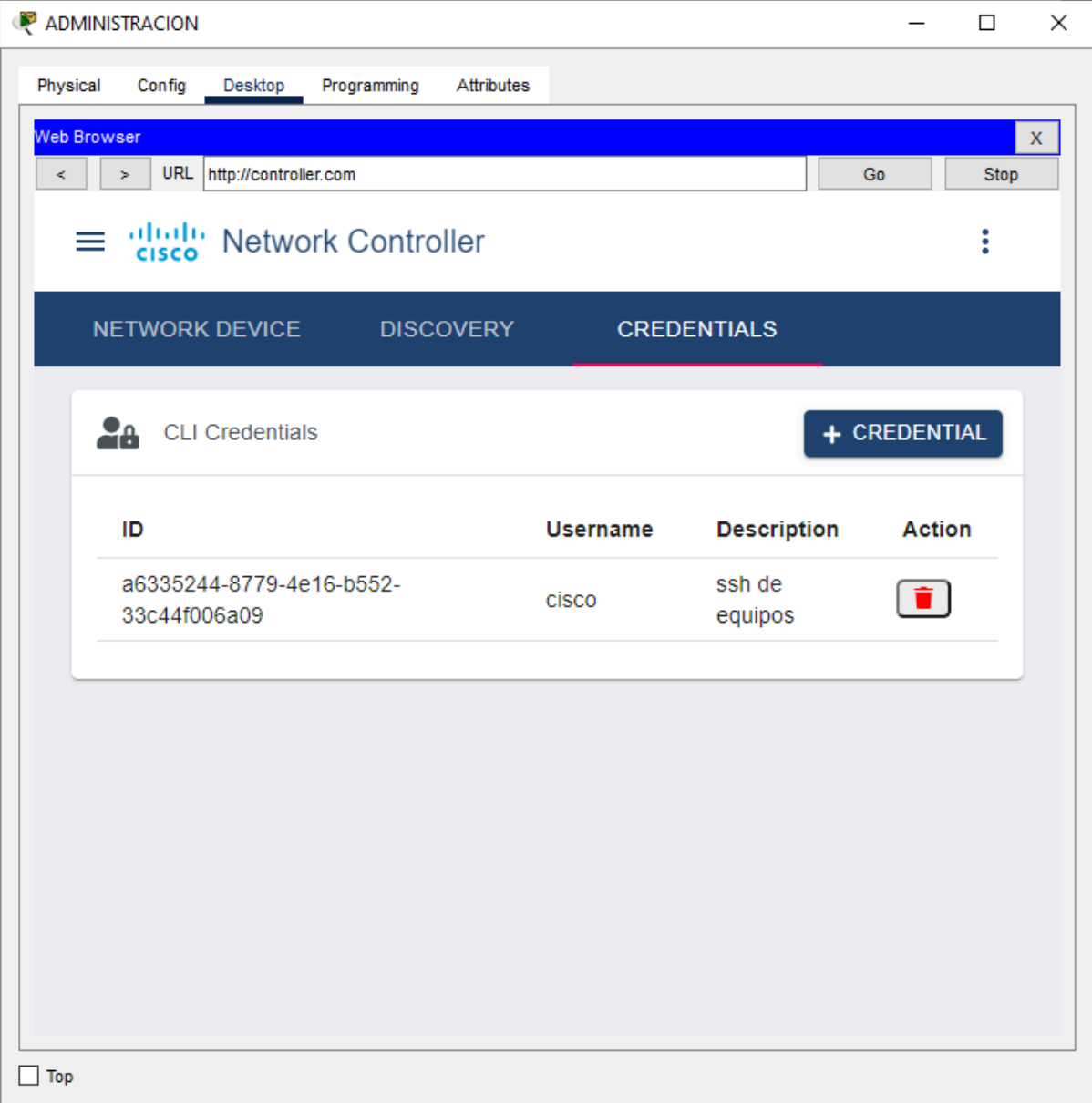
## Configuración SW2




## Network Controller

El Network Controller es un dispositivo que permite gestionar varios aspectos de la red, con tan solo configurar ssh en los dispositivos de la red y luego cargar las credenciales al propio controlador desde su interfaz. Esto se puede realizar desde una PC administrativa, o desde el exterior de Packet Tracer habilitándolo la opción desde el propio controller. En cualquiera de los casos, la configuración se realiza mediante una API o desde la aplicación web que se presenta accediendo desde un navegador al IP del controller.

Con respecto a la configuración del Network Controller, en nuestro caso utilizamos un solo set de credenciales, siendo el usuario *cisco* y la contraseña *class*.



The screenshot shows the Network Controller web interface. At the top, there's a navigation bar with tabs: Physical, Config, Desktop, Programming, and Attributes. The 'Desktop' tab is selected. Below this is a 'Web Browser' window showing the URL 'http://controller.com'. The main content area has a header with the Cisco logo and 'Network Controller'. Below the header is a navigation bar with three tabs: NETWORK DEVICE, DISCOVERY, and CREDENTIALS. The 'CREDENTIALS' tab is selected. Under this tab, there's a section titled 'CLI Credentials' with a '+ CREDENTIAL' button. Below this is a table with the following data:

ID	Username	Description	Action
a6335244-8779-4e16-b552-33c44f006a09	cisco	ssh de equipos	

At the bottom left, there's a 'Top' link.

A continuación se realizó un Discovery, donde se observaron todos los dispositivos de la red desde el Switch SW2 con IP 172.16.10.254.

ADMINISTRACION

Physical Config **Desktop** Programming Attributes

Web Browser

URL http://controller.com Go Stop

Network Controller

NETWORK DEVICE **DISCOVERY** CREDENTIALS

Discoveries + DISCOVERY

Status	Name
✓ Complete	a
✓ Complete	b
✓ Complete	c
✓ Complete	SW2
✓ Complete	internal_inventory_management
✓ Complete	internal_health_check
🔄 In Progress	internal_health_check

SW2 EDIT START

Top

En la siguiente imagen se pueden apreciar los nodos de la red encontrados por el controlador, seguido por la topología de la red:

ADMINISTRACION

PhysicalConfigDesktopProgrammingAttributes

Web Browser

<>URLhttp://controller.comGoStop

Network Controller

cisco

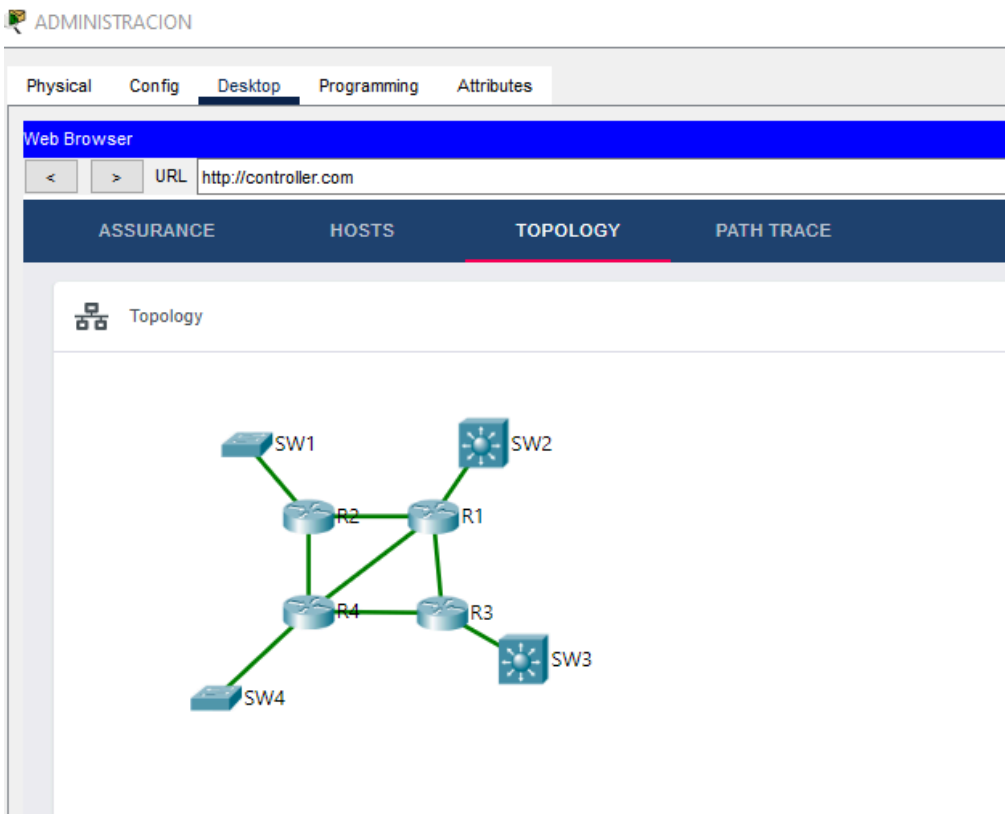
NETWORK DEVICEDISCOVERYCREDENTIALS

Network Device

+ DEVICE

	Hostname	Type	IP	Up Time	Last Updated	Collection Status
⚙	SW2	MultiLayerSwitch	172.16.20.254	59 minutes, 6 seconds	2024-11-17 22:17:09	Managed
⚙			172.16.10.253		2024-11-17 22:17:09	Unsupported
⚙	R2	Router	172.16.40.254	59 minutes, 6 seconds	2024-11-17 22:17:09	Managed
⚙	R1	Router	10.1.1.13	59 minutes, 6 seconds	2024-11-17 22:17:09	Managed
⚙	R2	Router	172.16.1.2	2 hours, 21 minutes, 49 seconds	2024-11-17 22:16:59	Unreachable
⚙	R4	Router	172.16.60.254	59 minutes, 7 seconds	2024-11-17 22:17:10	Managed
⚙	R3	Router	10.1.1.22	59 minutes, 7 seconds	2024-11-17 22:17:10	Managed
⚙	SW3	MultiLayerSwitch	172.16.80.254	59 minutes, 7 seconds	2024-11-17 22:17:10	Managed
⚙			100.100.100.1		2024-11-17 22:17:10	Wrong credential
⚙	SW1	Switch	172.16.1.1	59 minutes, 8 seconds	2024-11-17 22:17:11	Managed
⚙	SW4	Switch	172.16.4.1	59 minutes, 8 seconds	2024-11-17 22:17:11	Managed

☐ Top



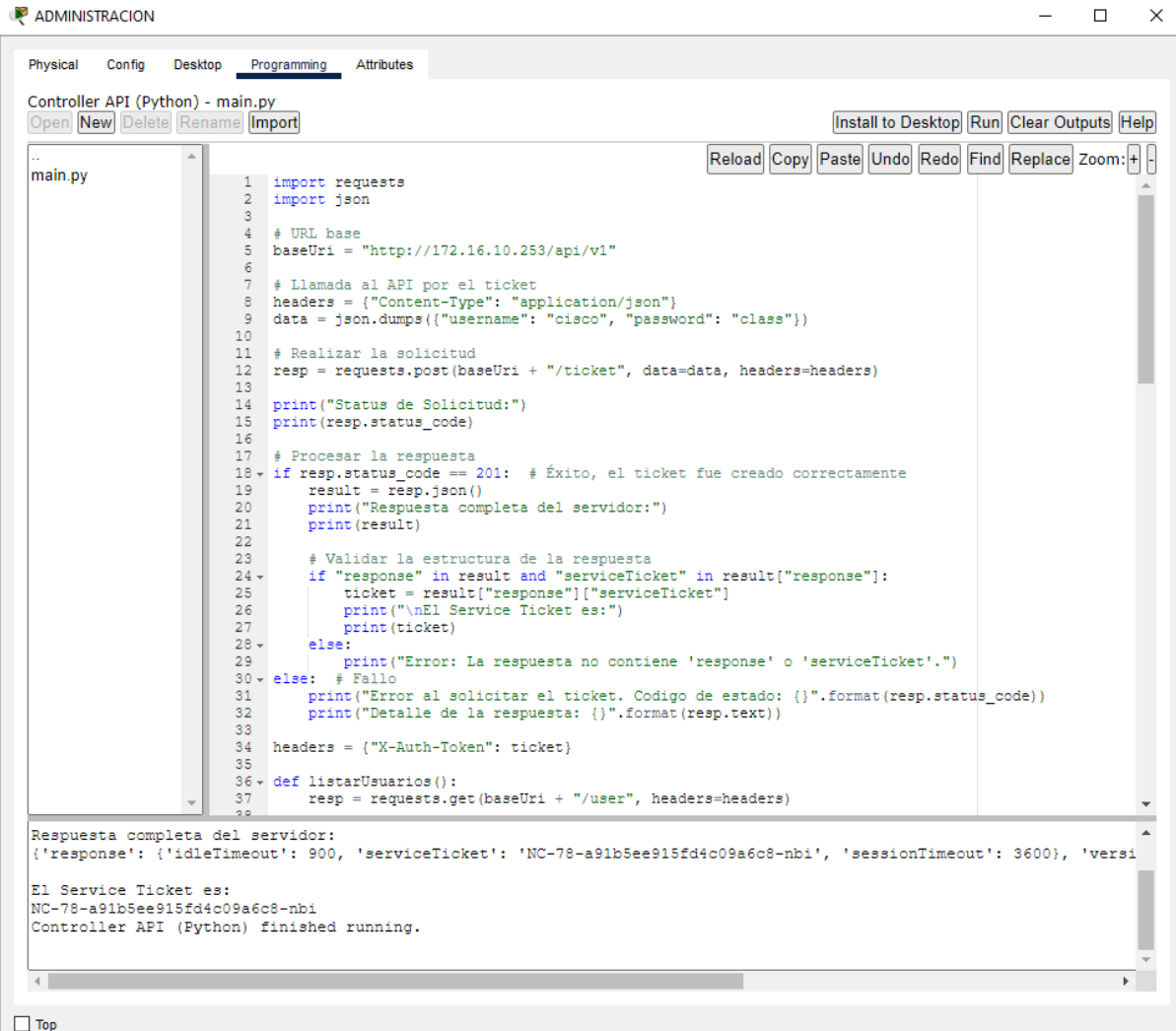


# APIs

Todo el script se encuentra en la PC PT-ADMINISTRACION, y se adjuntara como un **.txt**.

Primero obtenemos un ticket (credenciales) para realizar posteriores consultas a endpoints de la API de CISCO, se maneja con el formato JSON. Para esto se realiza una petición HTTP de tipo GET al endpoint */ticket* como se aprecia en la imagen.

TICKET:



The screenshot shows a web-based IDE window titled "ADMINISTRACION". The "Programming" tab is active, displaying a Python script named "main.py". The script uses the 'requests' and 'json' libraries to send a POST request to the endpoint "http://172.16.10.253/api/v1/ticket" with a JSON body containing "username": "cisco" and "password": "class". The script prints the status code and the full response. It then checks if the status code is 201 (success) and if the response contains a "serviceTicket". If successful, it prints the ticket and sets it as an "X-Auth-Token" header for subsequent requests. The output console at the bottom shows the successful execution, displaying the ticket and the session timeout.

```
1 import requests
2 import json
3
4 # URL base
5 baseUrl = "http://172.16.10.253/api/v1"
6
7 # Llamada al API por el ticket
8 headers = {"Content-Type": "application/json"}
9 data = json.dumps({"username": "cisco", "password": "class"})
10
11 # Realizar la solicitud
12 resp = requests.post(baseUrl + "/ticket", data=data, headers=headers)
13
14 print("Status de Solicitud:")
15 print(resp.status_code)
16
17 # Procesar la respuesta
18 if resp.status_code == 201: # Éxito, el ticket fue creado correctamente
19     result = resp.json()
20     print("Respuesta completa del servidor:")
21     print(result)
22
23     # Validar la estructura de la respuesta
24     if "response" in result and "serviceTicket" in result["response"]:
25         ticket = result["response"]["serviceTicket"]
26         print("\nEl Service Ticket es:")
27         print(ticket)
28     else:
29         print("Error: La respuesta no contiene 'response' o 'serviceTicket'.")
30 else: # Fallo
31     print("Error al solicitar el ticket. Código de estado: {}".format(resp.status_code))
32     print("Detalle de la respuesta: {}".format(resp.text))
33
34 headers = {"X-Auth-Token": ticket}
35
36 def listarUsuarios():
37     resp = requests.get(baseUrl + "/user", headers=headers)
```

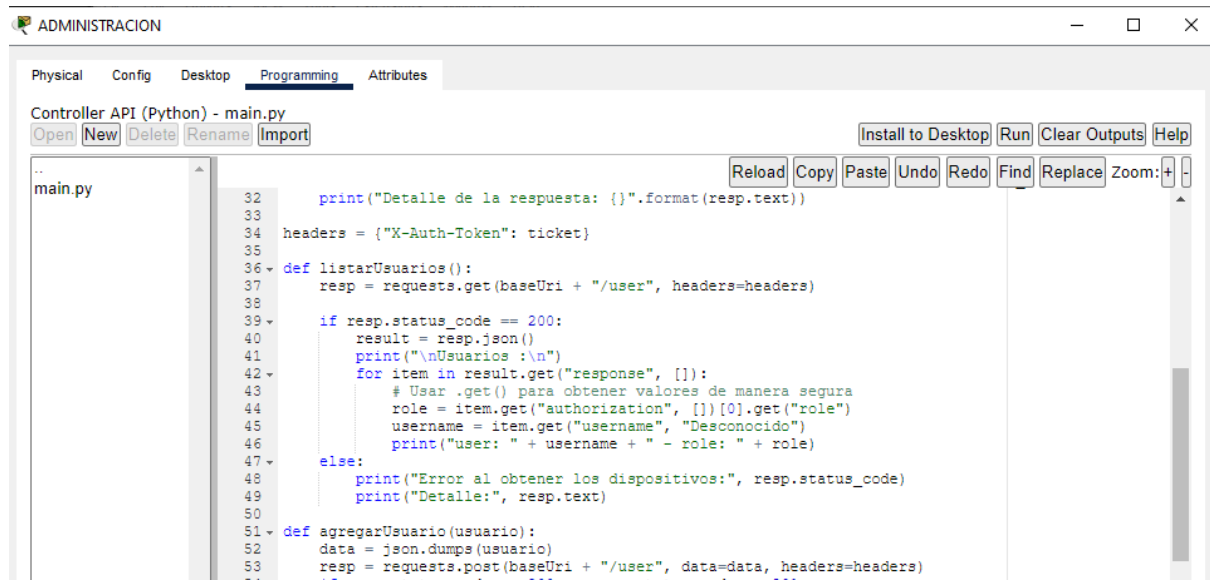
Respuesta completa del servidor:  
{'response': {'idleTimeout': 900, 'serviceTicket': 'NC-78-a91b5ee915fd4c09a6c8-nbi', 'sessionTimeout': 3600}, 'versi

El Service Ticket es:  
NC-78-a91b5ee915fd4c09a6c8-nbi  
Controller API (Python) finished running.

Listar Usuarios:

Se utilizó el endpoint `/user` con el método HTTP GET para obtener los usuarios disponibles:

Funcion:



The screenshot shows a code editor window titled "ADMINISTRACION" with tabs for Physical, Config, Desktop, Programming, and Attributes. The Programming tab is active, showing a file named "main.py". The code in the editor is as follows:

```
32     print("Detalle de la respuesta: {}".format(resp.text))
33
34     headers = {"X-Auth-Token": ticket}
35
36     def listarUsuarios():
37         resp = requests.get(baseUrl + "/user", headers=headers)
38
39         if resp.status_code == 200:
40             result = resp.json()
41             print("\nUsuarios :\n")
42             for item in result.get("response", []):
43                 # Usar .get() para obtener valores de manera segura
44                 role = item.get("authorization", [])[0].get("role")
45                 username = item.get("username", "Desconocido")
46                 print("user: " + username + " - role: " + role)
47         else:
48             print("Error al obtener los dispositivos:", resp.status_code)
49             print("Detalle:", resp.text)
50
51     def agregarUsuario(usuario):
52         data = json.dumps(usuario)
53         resp = requests.post(baseUrl + "/user", data=data, headers=headers)
54         if resp.status_code == 200 or resp.status_code == 201:
```

Ejecución:



The screenshot shows a terminal window with the following output:

```
110
111 listarUsuarios()

El Service Ticket es:
NC-79-3fa9cfd65e6d4ad798cb-nbi

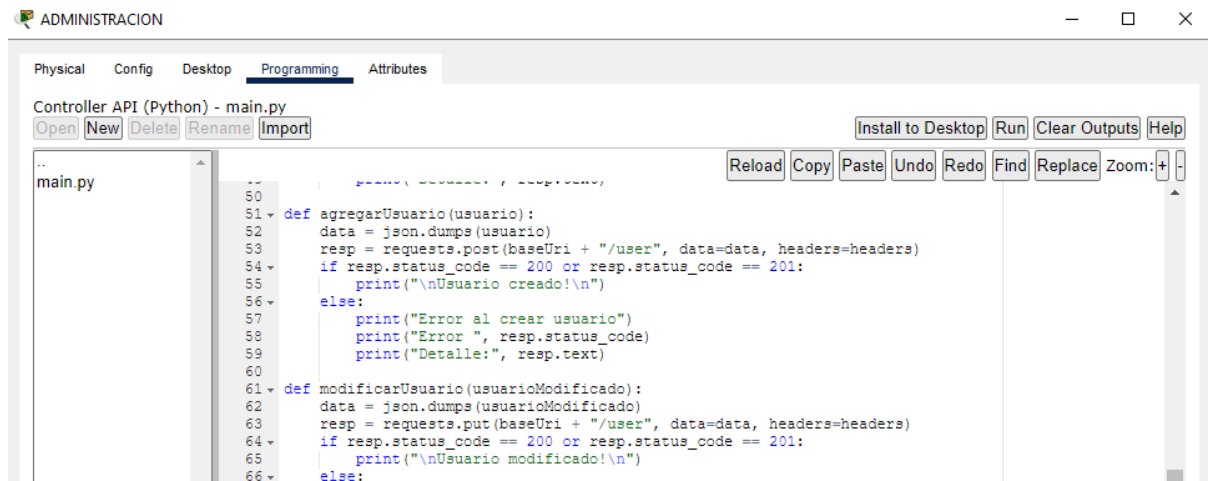
Usuarios :

user: cisco - role: ROLE_ADMIN
Controller API (Python) finished running.
```

## Agregar Usuario:

Se utilizó el endpoint `/user` con el método HTTP POST para agregar un nuevo usuario:

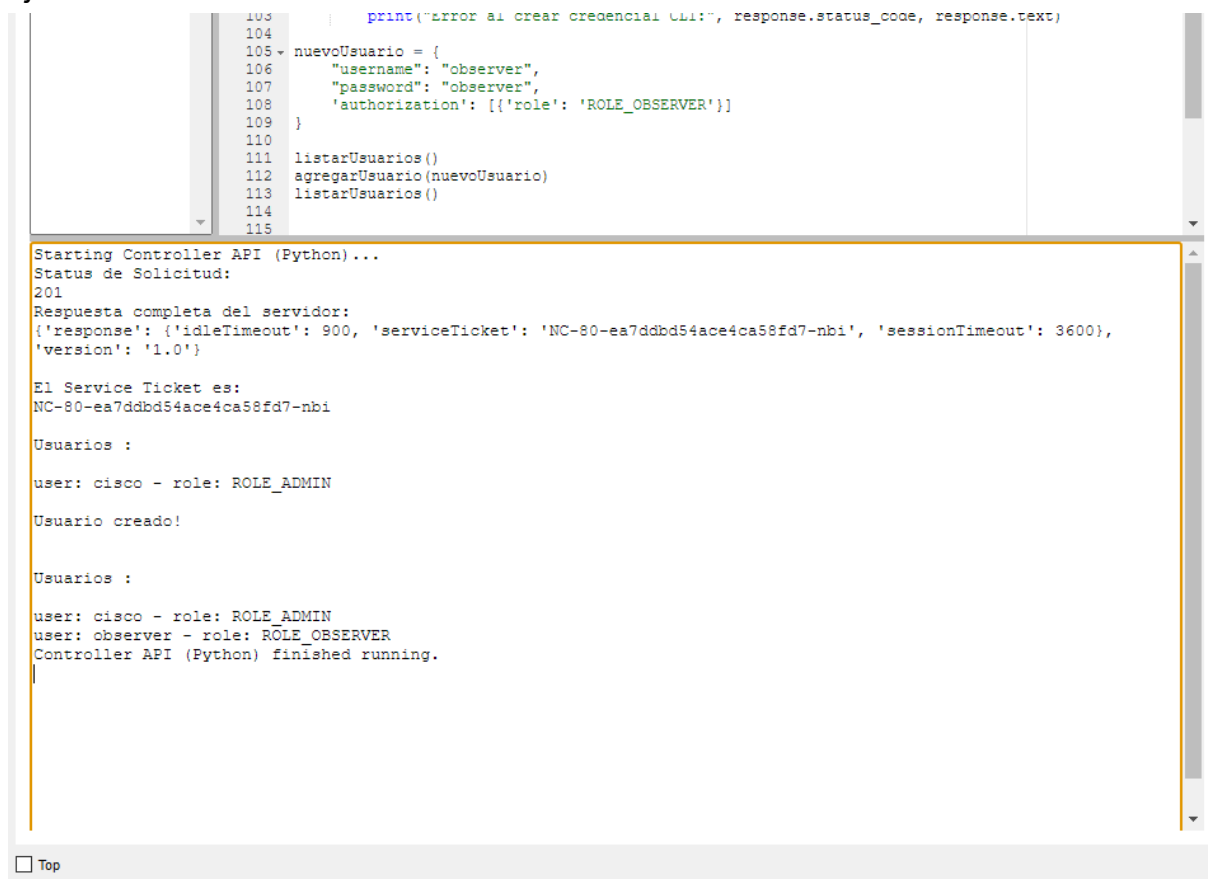
## Function:



The screenshot shows a code editor window titled "ADMINISTRACION" with tabs for Physical, Config, Desktop, Programming, and Attributes. The Programming tab is active, showing a file named "main.py". The code defines two functions: `agregarUsuario` and `modificarUsuario`. Both functions use the `requests` library to interact with an API endpoint `/user`. `agregarUsuario` sends a POST request, and `modificarUsuario` sends a PUT request. Both functions check the response status code and print messages accordingly.

```
..
main.py
50
51 def agregarUsuario(usuario):
52     data = json.dumps(usuario)
53     resp = requests.post(baseUrl + "/user", data=data, headers=headers)
54     if resp.status_code == 200 or resp.status_code == 201:
55         print("\nUsuario creado!\n")
56     else:
57         print("Error al crear usuario")
58         print("Error ", resp.status_code)
59         print("Detalle:", resp.text)
60
61 def modificarUsuario(usuarioModificado):
62     data = json.dumps(usuarioModificado)
63     resp = requests.put(baseUrl + "/user", data=data, headers=headers)
64     if resp.status_code == 200 or resp.status_code == 201:
65         print("\nUsuario modificado!\n")
66     else:
```

## Ejecución:



The screenshot shows the same code editor window with the code from the previous block. Below the code, there is a terminal output window showing the execution of the script. The output includes the status of the request, the response from the server, the service ticket, and the list of users.

```
103         print("Error al crear credencial UI:", response.status_code, response.text)
104
105 nuevoUsuario = {
106     "username": "observer",
107     "password": "observer",
108     'authorization': [{'role': 'ROLE_OBSERVER'}]
109 }
110
111 listarUsuarios()
112 agregarUsuario(nuevoUsuario)
113 listarUsuarios()
114
115

Starting Controller API (Python)...
Status de Solicitud:
201
Respuesta completa del servidor:
{'response': {'idleTimeout': 900, 'serviceTicket': 'NC-80-ea7ddb54ace4ca58fd7-nbi', 'sessionTimeout': 3600},
'version': '1.0'}

El Service Ticket es:
NC-80-ea7ddb54ace4ca58fd7-nbi

Usuarios :

user: cisco - role: ROLE_ADMIN

Usuario creado!

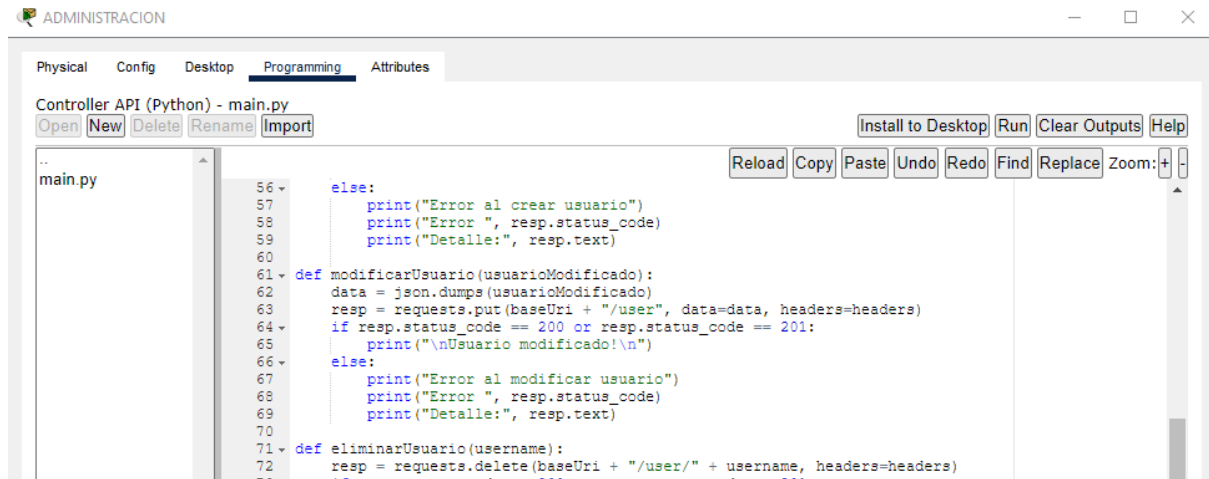
Usuarios :

user: cisco - role: ROLE_ADMIN
user: observer - role: ROLE_OBSERVER
Controller API (Python) finished running.
```

## Modificar Usuario:

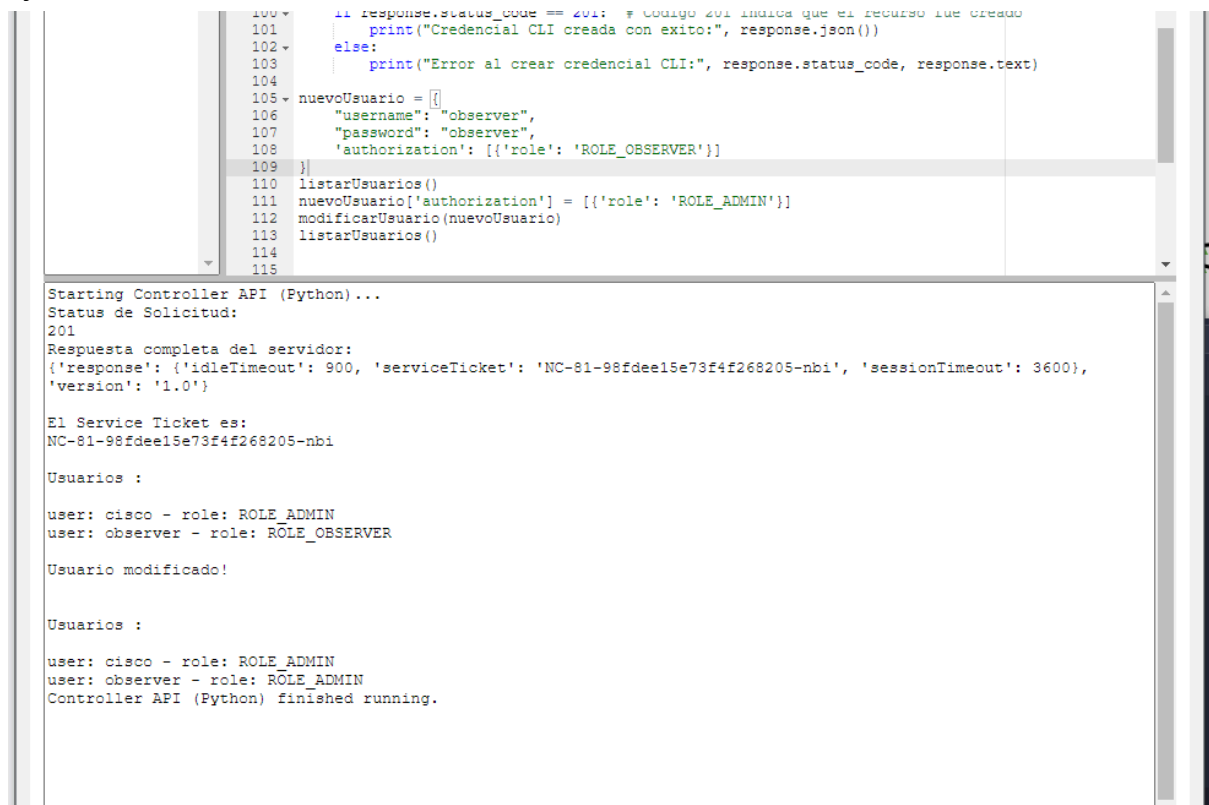
Se utilizó el endpoint `/user` con el método HTTP PUT para modificar un usuario existente:

## Funcion:



```
..
main.py
56 else:
57     print("Error al crear usuario")
58     print("Error ", resp.status_code)
59     print("Detalle:", resp.text)
60
61 def modificarUsuario(usuarioModificado):
62     data = json.dumps(usuarioModificado)
63     resp = requests.put(baseUrl + "/user", data=data, headers=headers)
64     if resp.status_code == 200 or resp.status_code == 201:
65         print("\nUsuario modificado!\n")
66     else:
67         print("Error al modificar usuario")
68         print("Error ", resp.status_code)
69         print("Detalle:", resp.text)
70
71 def eliminarUsuario(username):
72     resp = requests.delete(baseUrl + "/user/" + username, headers=headers)
73     if resp.status_code == 200 or resp.status_code == 201:
```

## Ejecución



```
100 if response.status_code == 201: # Código 201 indica que el recurso fue creado
101     print("Credencial CLI creada con éxito:", response.json())
102 else:
103     print("Error al crear credencial CLI:", response.status_code, response.text)
104
105 nuevoUsuario = [
106     "username": "observer",
107     "password": "observer",
108     'authorization': [{'role': 'ROLE_OBSERVER'}]
109 ]
110 listarUsuarios()
111 nuevoUsuario['authorization'] = [{'role': 'ROLE_ADMIN'}]
112 modificarUsuario(nuevoUsuario)
113 listarUsuarios()
114
115
```

Starting Controller API (Python)...

Status de Solicitud:  
201

Respuesta completa del servidor:  
{'response': {'idleTimeout': 900, 'serviceTicket': 'NC-81-98fdee15e73f4f268205-nbi', 'sessionTimeout': 3600}, 'version': '1.0'}

El Service Ticket es:  
NC-81-98fdee15e73f4f268205-nbi

Usuarios :

user: cisco - role: ROLE\_ADMIN  
user: observer - role: ROLE\_OBSERVER

Usuario modificado!

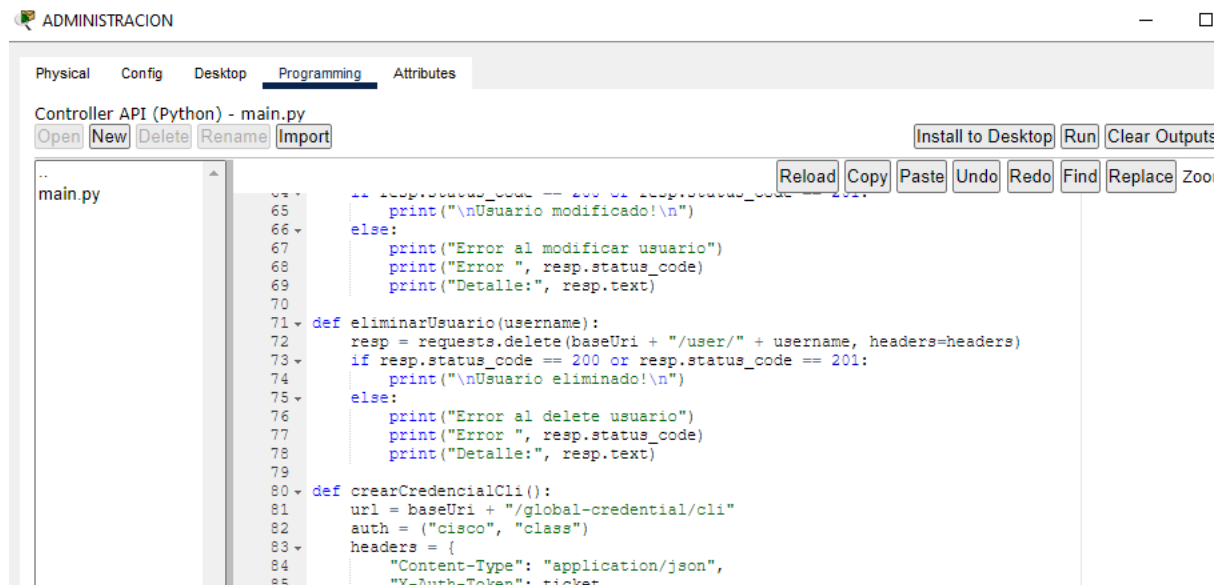
Usuarios :

user: cisco - role: ROLE\_ADMIN  
user: observer - role: ROLE\_ADMIN  
Controller API (Python) finished running.

## Delete Usuario:

Se utilizó el endpoint `/user/{username}` con el método HTTP DELETE para modificar un usuario existente:

## Funcion:



The screenshot shows a code editor window titled "ADMINISTRACION" with tabs for Physical, Config, Desktop, Programming, and Attributes. The Programming tab is active, showing the file "Controller API (Python) - main.py". The code defines a function `eliminarUsuario(username)` that sends a DELETE request to `baseUri + "/user/" + username`. It checks the response status code: 200 or 201 indicates success, while 400 or 401 indicates an error. The function also includes a `crearCredencialCli()` function that sets headers for a CLI request.

```
..
main.py
65     resp.status_code == 400 or resp.status_code == 401:
66         print("\nUsuario modificado!\n")
67     else:
68         print("Error al modificar usuario")
69         print("Error ", resp.status_code)
70         print("Detalle:", resp.text)
71
72 def eliminarUsuario(username):
73     resp = requests.delete(baseUri + "/user/" + username, headers=headers)
74     if resp.status_code == 200 or resp.status_code == 201:
75         print("\nUsuario eliminado!\n")
76     else:
77         print("Error al delete usuario")
78         print("Error ", resp.status_code)
79         print("Detalle:", resp.text)
80
81 def crearCredencialCli():
82     url = baseUri + "/global-credential/cli"
83     auth = ("cisco", "class")
84     headers = {
85         "Content-Type": "application/json",
86         "X-Auth-Token": ticket
```

## Ejecución:



The screenshot shows a terminal window with the following output:

```
Starting Controller API (Python)...
Status de Solicitud:
201
Respuesta completa del servidor:
{'response': {'idleTimeout': 900, 'serviceTicket': 'NC-82-ac50c7daaef5405fbbb5-nbi', 'sessionTimeout': 3600},
'version': '1.0'}

El Service Ticket es:
NC-82-ac50c7daaef5405fbbb5-nbi

Usuarios :

user: cisco - role: ROLE_ADMIN
user: observer - role: ROLE_ADMIN

Usuario eliminado!

Usuarios :

user: cisco - role: ROLE_ADMIN
Controller API (Python) finished running.
```

## Crear Credencial CLI:

Se utilizó el endpoint `/global-credential/cli` con el método HTTP POST para crear credenciales de cli:

## Funcion:

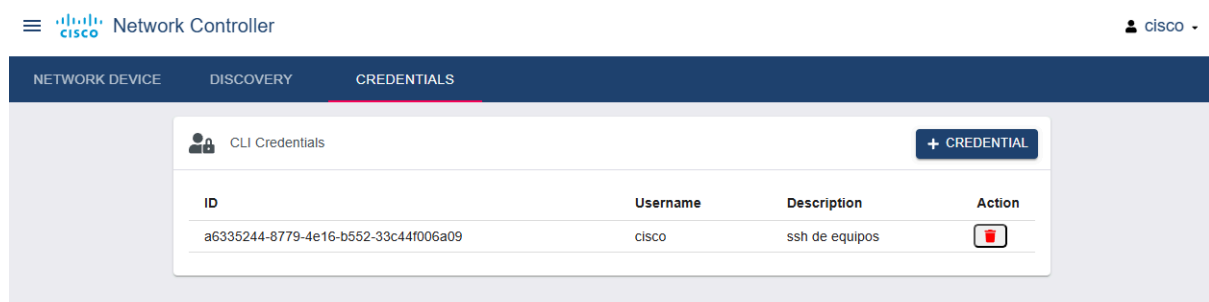


The screenshot shows a code editor window titled "ADMINISTRACION" with a tab for "Controller API (Python) - main.py". The code defines a function `crearCredencialCli()` that sends a POST request to `/global-credential/cli` with a JSON payload. The payload includes fields for `credentialType`, `username`, `password`, `enablePassword`, `comments`, and `description`. After the request, it checks the `status_code` and prints a success or error message. Finally, it calls `crearCredencialCli()`.


```
..
80 def crearCredencialCli():
81     url = baseUrl + "/global-credential/cli"
82     auth = ("cisco", "class")
83     headers = {
84         "Content-Type": "application/json",
85         "X-Auth-Token": ticket
86     }
87
88     # datos de la credencial CLI
89     payload = {
90         "credentialType": "CLI",
91         "username": "cisco", # Usuario de la CLI
92         "password": "class", # Contraseña del usuario
93         "enablePassword": "enablepassword123", # Contraseña para modo privilegiado
94         "comments": "Credencial para acceso CLI",
95         "description": "ssh - API"
96     }
97
98     response = requests.post(url, auth=auth, json=payload, headers=headers)
99
100     if response.status_code == 201: # Código 201 indica que el recurso fue creado
101         print("Credencial CLI creada con éxito:", response.json())
102     else:
103         print("Error al crear credencial CLI:", response.status_code, response.text)
104
105     nuevoUsuario = {
106         "username": "observer",
107         "password": "observer",
108         'authorization': [{'role': 'ROLE_OBSERVER'}]
109     }
110
111     crearCredencialCli()
112
113 ..
```

## Ejecucion:

- Antes



The screenshot shows the Cisco Network Controller interface with the "CREDENTIALS" tab selected. It displays a table of CLI Credentials. The table has columns for ID, Username, Description, and Action. There is one entry with ID `a6335244-8779-4e16-b552-33c44f006a09`, Username `cisco`, and Description `ssh de equipos`. A red square icon with a white 'X' is in the Action column. A "+ CREDENTIAL" button is in the top right corner.

ID	Username	Description	Action
a6335244-8779-4e16-b552-33c44f006a09	cisco	ssh de equipos	

Physical Config Desktop **Programming** Attributes

Controller API (Python) - main.py

Open New Delete Rename Import

Install to Desktop Run Clear Outputs Help

Reload Copy Paste Undo Redo Find Replace Zoom: + -

```

80 def crearCredencialCli():
81     url = baseUrl + "/global-credential/cli"
82     auth = ("cisco", "class")
83     headers = {
84         "Content-Type": "application/json",
85         "X-Auth-Token": ticket
86     }
87
88     # datos de la credencial CLI
89     payload = {
90         "credentialType": "CLI",
91         "username": "cisco", # Usuario de la CLI
92         "password": "class", # Contraseña del usuario
93         "enablePassword": "enablepassword123", # Contraseña para modo privilegiado
94         "comments": "Credencial para acceso CLI",
95         "description": "ssh - API"
96     }
97
98     response = requests.post(url, auth=auth, json=payload, headers=headers)
99
100     if response.status_code == 201: # Código 201 indica que el recurso fue creado
101         print("Credencial CLI creada con éxito:", response.json())
102     else:
103         print("Error al crear credencial CLI:", response.status_code, response.text)
104
105     nuevoUsuario = {
106         "username": "observer",
107         "password": "observer",
108         'authorization': [{'role': 'ROLE_OBSERVER'}]
109     }
110
111     crearCredencialCli()
112
113

```

Starting Controller API (Python)...

Status de Solicitud:  
201

Respuesta completa del servidor:  
{'response': {'idleTimeout': 900, 'serviceTicket': 'NC-85-92b5495d46c04cac9860-nbi', 'sessionTimeout': 3600}, 'version': '1.0'}

El Service Ticket es:  
NC-85-92b5495d46c04cac9860-nbi

{'Credencial CLI creada con éxito:', {'username': 'cisco', 'password': 'class', 'credentialType': 'CLI', 'enablePassword': 'enablepassword123', 'comments': 'Credencial para acceso CLI', 'description': 'ssh - API', 'id': '1d68bfa9-0a6c-42d1-b470-8b110432e460', 'instanceUuid': '1d68bfa9-0a6c-42d1-b470-8b110432e460'}}

Controller API (Python) finished running

- Después

Network Controller

NETWORK DEVICE DISCOVERY **CREDENTIALS**

CLI Credentials + CREDENTIAL

ID	Username	Description	Action
1d68bfa9-0a6c-42d1-b470-8b110432e460	cisco	ssh - API	
a6335244-8779-4e16-b552-33c44f006a09	cisco	ssh de equipos	