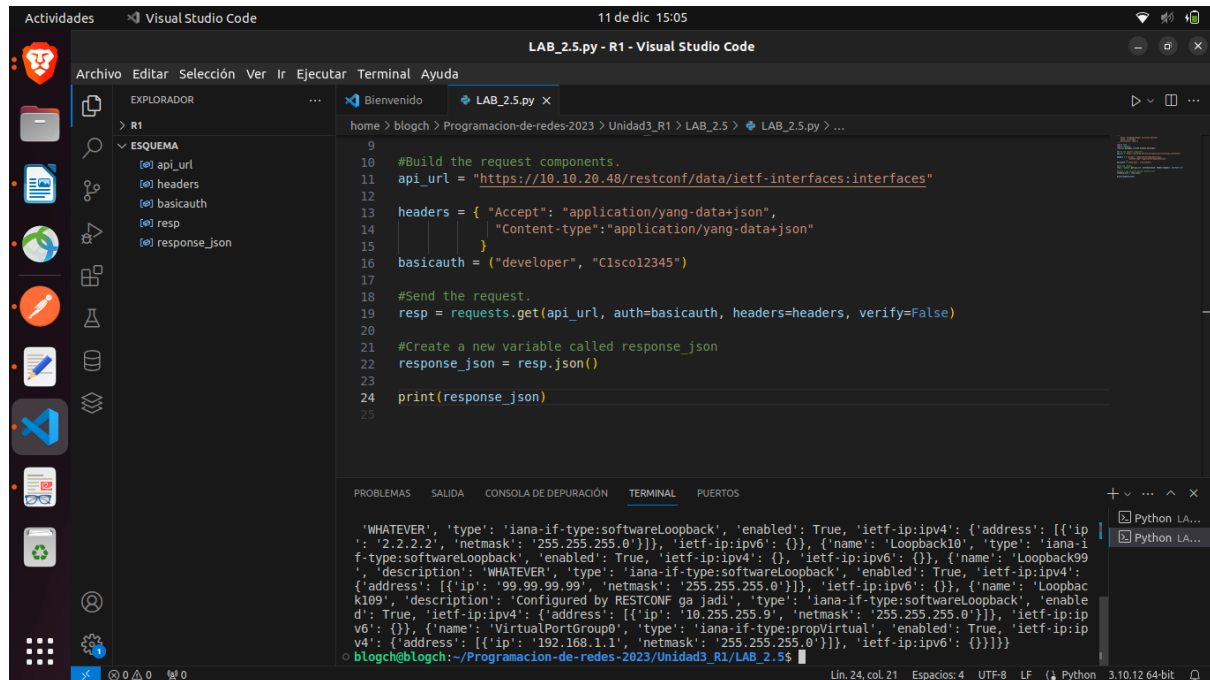


Christian Giovanni Arredondo Rangel  
1222100691  
Part 1: RESTCONF basics in Python

## Part 2: Modify interface configuration with RESTCONF in Python

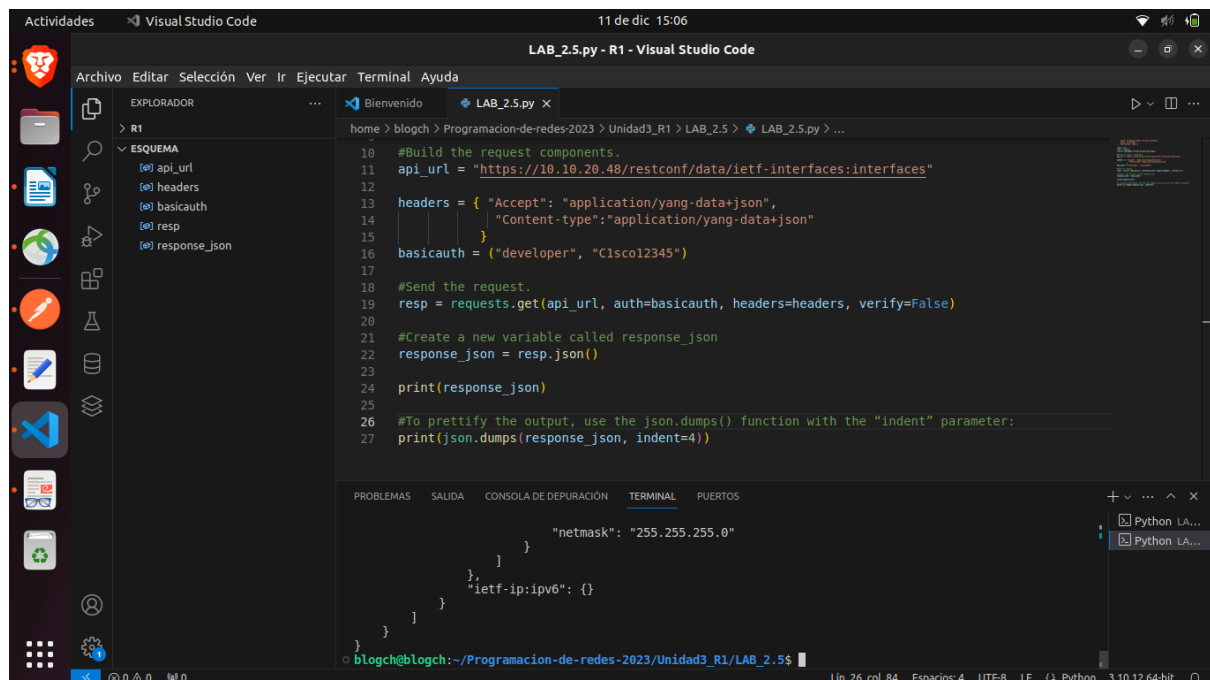


```
LAB_2.5.py - R1 - Visual Studio Code

9
10 #Build the request components.
11 api_url = "https://10.10.20.48/restconf/data/ietf-interfaces:interfaces"
12
13 headers = { "Accept": "application/yang-data+json",
14             "Content-type": "application/yang-data+json"
15           }
16 basicauth = ("developer", "C1sco12345")
17
18 #Send the request.
19 resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
20
21 #Create a new variable called response_json
22 response_json = resp.json()
23
24 print(response_json)
25
```

```
'WHATEVER', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip': '2.2.2.2', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}, {'name': 'Loopback10', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {}, 'ietf-ip:ipv6': {}, {'name': 'Loopback99', 'description': 'WHATEVER', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip': '99.99.99.99', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}, {'name': 'Loopback109', 'description': 'Configured by RESTCONF ga jadi', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip': '10.255.255.9', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}, {'name': 'VirtualPortGroup0', 'type': 'iana-if-type:propVirtual', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip': '192.168.1.1', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}}}}]
```

#To prettify the output, use the json.dumps() function with the "indent" parameter:



```
LAB_2.5.py - R1 - Visual Studio Code

10 #Build the request components.
11 api_url = "https://10.10.20.48/restconf/data/ietf-interfaces:interfaces"
12
13 headers = { "Accept": "application/yang-data+json",
14             "Content-type": "application/yang-data+json"
15           }
16 basicauth = ("developer", "C1sco12345")
17
18 #Send the request.
19 resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
20
21 #Create a new variable called response_json
22 response_json = resp.json()
23
24 print(response_json)
25
26 #To prettify the output, use the json.dumps() function with the "indent" parameter:
27 print(json.dumps(response_json, indent=4))
```

```
    "netmask": "255.255.255.0"
  }
},
"ietf-ip:ipv6": {}
}
}
```

Parte 2:

Modifique la loopback por la 100 porque ya tenia una loopback99

```

35     "ietf-ip:ipv6": {}
36 }
37 }
38
39 #Send the PUT request.
40
41 resp = requests.put(api_url, data=json.dumps(yangConfig), auth=basicauth, headers=headers, verify=True)
42 if (resp.status_code >= 200 and resp.status_code <= 299):
43     print("STATUS OK: {}".format(resp.status_code))
44 else:
45     print("Error code {}, reply: {}".format(resp.status_code, resp.json()))

```

```

• blogch@blogch:~/Programacion-de-redes-2023/Unidad3_R1/LAB_2.5$ python3 LAB_2.5_part2.py
STATUS OK: 204
• blogch@blogch:~/Programacion-de-redes-2023/Unidad3_R1/LAB_2.5$ python3 LAB_2.5_part2.py
Error code 400, reply: {'ietf-restconf:errors': {'error': [{'error-type': 'application', 'error-tag': 'malformed-message', 'error-path': '/ietf-interfaces:interfaces/interface[name=Loopback99]', 'error-message': 'Mismatched keypaths: /if:interfaces/if:interface[if:name=Loopback100] , /if:interfaces/if:interface[if:name=Loopback99]']}}}
• blogch@blogch:~/Programacion-de-redes-2023/Unidad3_R1/LAB_2.5$ python3 LAB_2.5_part2.py
STATUS OK: 201
• blogch@blogch:~/Programacion-de-redes-2023/Unidad3_R1/LAB_2.5$

```

w. Verifique usando la CLI de IOS que se haya creado la nueva interfaz Loopback100 (sh ip int brief).

Utilice el archivo LAB2.2.py para ver las interfaces y sus ip's

```

40
41 resp = requests.put(api_url, data=json.dumps(yangConfig), auth=basicauth, headers=headers, verify=True)
42 if (resp.status_code >= 200 and resp.status_code <= 299):
43     print("STATUS OK: {}".format(resp.status_code))
44 else:
45     print("Error code {}, reply: {}".format(resp.status_code, resp.json()))
46

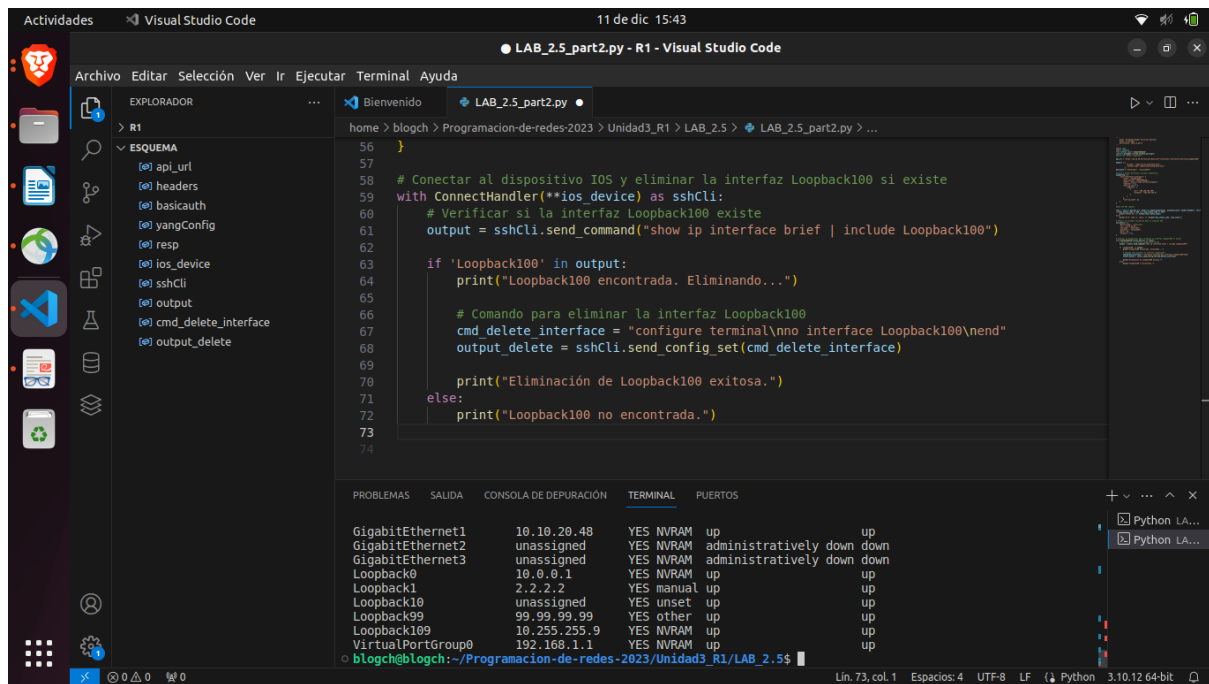
```

```

GigabitEthernet2      unassigned    YES NVRAM   administratively down down
GigabitEthernet3      unassigned    YES NVRAM   administratively down down
Loopback0             10.0.0.1      YES NVRAM   up            up
Loopback1             2.2.2.2       YES manual  up            up
Loopback10            unassigned    YES unset   up            up
Loopback99            99.99.99.99   YES other   up            up
Loopback100           100.100.100.100 YES other   up            up
Loopback109           10.255.255.9  YES NVRAM   up            up
VirtualPortGroup0     192.168.1.1   YES NVRAM   up            up
• blogch@blogch:~/Programacion-de-redes-2023/Unidad3_R1/LAB_2.5$

```

X. Modifique el código para eliminar la interfaz Loopback100.



y ¿Qué cambios se aplicaron al código para eliminar la interfaz Loopback100?

Esto consta de dos partes

La primera fue agregar una sección al código para decirle a Python cómo conectara al router con la ip 10.10.20.48

```

ios_device = {
    'device_type': 'cisco_ios',
    'ip': '10.10.20.48',
    'username': 'developer',
    'password': 'C1sc0l2345',
    'port': 22,
    'verbose': True,
}

```

y la segunda fue crear un bloque que buscara una interfaz llamada loopback100

```

# Conectar al dispositivo IOS y eliminar la interfaz Loopback100 si existe
with ConnectHandler(**ios_device) as sshCli:
    # Verificar si la interfaz Loopback100 existe
    output = sshCli.send_command("show ip interface brief | include Loopback100")

    if 'Loopback100' in output:
        print("Loopback100 encontrada. Eliminando...")

        # Comando para eliminar la interfaz Loopback100
        cmd_delete_interface = "configure terminal\nno interface Loopback100\nend"
        output_delete = sshCli.send_config_set(cmd_delete_interface)

        print("Eliminación de Loopback100 exitosa.")
    else:
        print("Loopback100 no encontrada.")

```