



# **MikroTik**

Trabalho Laboratorial nº 1/ (TL1)  
Laboratório de Tecnologias de Informação

EI 2022/23

Grupo G14

Rostyslav Romanyshyn nº 2200668

Rodrigo Ferreira nº 2201702

Leiria, abril de 2023



# **Resumo**

O presente relatório tem por objetivo apresentar e descrever de forma detalhada o trabalho que foi desenvolvido na criação de uma SDN (Software Defined Network), com recurso á linguagem Python, para a configuração de um router MikroTik.

# **Abstract**

In this report, we present and describe in detail the work that has been done in creating an SDN (Software Defined Network), using the Python language, for configuring MikroTik routers.

**Keywords:** MikroTik, SDN, Python

# Índice

<b>Resumo.....</b>	<b>iii</b>
<b>Abstract.....</b>	<b>iv</b>
<b>Lista de Figuras.....</b>	<b>viii</b>
<b>Lista de Tabelas.....</b>	<b>xii</b>
<b>1. Introdução .....</b>	<b>1</b>
<b>2. Fundamentação Teórica .....</b>	<b>2</b>
<b>3. Bibliotecas usadas no Python .....</b>	<b>3</b>
<b>4. Ações disponibilizadas no projeto .....</b>	<b>4</b>
<b>4.1. Controlar mais do que um dispositivo com a mesma aplicação.....</b>	<b>4</b>
<b>4.2. Listar todas as interfaces do dispositivo .....</b>	<b>5</b>
<b>4.3. Listar apenas as interfaces wireless .....</b>	<b>6</b>
<b>4.4. Listar/criar/editar/apagar interfaces bridge e respectivas portas associadas</b>	<b>7</b>
4.4.1. Listar interfaces bridge .....	7
4.4.2. Criar interfaces bridge .....	8
4.4.3. Editar interfaces bridge.....	9
4.4.4. Apagar interfaces bridge.....	10
4.4.5. Listar portas das interfaces bridge.....	11
4.4.6. Criar portas para as interfaces bridge .....	12
4.4.7. Editar portas bridge .....	13
4.4.8. Apagar portas bridge .....	14
<b>5. Criar/editar/apagar perfis de segurança para utilizar nas redes wireless ...</b>	<b>15</b>
5.1.1. Criar perfis de segurança .....	15
5.1.2. Editar perfis de segurança.....	16
5.1.3. Apagar perfis de segurança.....	17
<b>6. Ativar/desativar/configurar redes wireless .....</b>	<b>18</b>

6.1.	Ativar redes wireless .....	18
6.2.	Desativar redes wireless.....	19
6.3.	Configurar redes wireless .....	20
7.	Listar/criar/editar/apagar rotas estáticas .....	21
7.1.	Listar rotas estáticas .....	21
7.2.	Criar rotas estáticas .....	22
7.3.	Editar rotas estáticas.....	23
7.4.	Apagar rotas estáticas .....	24
8.	Listar/criar/editar/apagar endereços IP.....	25
8.1.	Listar endereços IP.....	25
8.2.	Criar endereços IP .....	26
8.3.	Editar endereços IP .....	27
8.4.	Apagar endereços IP .....	28
9.	Listar/ criar/editar/apagar servidores de DHCP.....	29
9.1.	Listar servidores de DHCP.....	29
9.2.	Criar servidores de DHCP .....	30
9.3.	Editar servidores de DHCP .....	31
9.4.	Apagar servidores de DHCP .....	32
10.	Ativar/desativar/configurar servidor DNS .....	33
10.1.1.	Ativar DNS.....	33
10.1.2.	Desativar DNS.....	34
10.1.1.	Configurar DNS .....	34
11.	Listar/criar/editar/apagar regras de firewall .....	35
11.1.1.	Listar Firewall .....	35

11.1.2.	Criar Firewall.....	36
11.1.3.	Editar Firewall .....	37
11.1.4.	Editar Firewall .....	38
<b>12.</b>	<b>Ativar/desativar/configurar protocolos de encaminhamento.....</b>	<b>39</b>
12.1.1.	Criar protocolo de encaminhamento.....	39
12.1.2.	Ativar protocolo de encaminhamento.....	41
12.1.3.	Desativar protocolo de encaminhamento .....	42
<b>13.</b>	<b>VPN.....</b>	<b>44</b>
<b>14.</b>	<b>Análise crítica e proposta de melhorias.....</b>	<b>46</b>
<b>15.</b>	<b>Conclusão .....</b>	<b>47</b>

# Lista de Figuras

Figura 3.1 : main page.....	3
Figura 4.1 : Concatenação das variáveis para URL.....	4
Figura 4.2 : IP, User, Password .....	4
Figura 4.3 : interfaces.....	5
Figura 4.4 : método get das interfaces.....	5
Figura 4.5 : Interface Wireless .....	6
Figura 4.6 : método get das interfaces Wireless .....	6
Figura 4.7 : interface bridge .....	7
Figura 4.8 : método get para listar interfaces bridge .....	7
Figura 4.9 : Listar interfaces bridge .....	7
Figura 4.10 : método put da interface bridge .....	8
Figura 4.11 : Criar interface Bridge .....	8
Figura 4.12 : Editar interfaces bridge .....	9
Figura 4.13 : método patch para a interface bridge .....	9
Figura 4.14 : método delete para a interface bridge .....	10
Figura 4.15 : Apagar interfaces bridge .....	10
Figura 4.16 : : Listar portas das interfaces bridge .....	11
Figura 4.17 : método get para listar portas das interfaces bridge .....	11
Figura 4.18 : Criar porta para a interface Bridge.....	12
Figura 4.19 : método put das portas bridge .....	12
Figura 4.20 : Editar porta Bridge .....	13
Figura 4.21: método patch das portas bridge.....	13
Figura 4.22 : Apagar porta bridge .....	14
Figura 4.23 : método delete para portas bridge .....	14
Figura 5.1 : perfis de segurança main page .....	15
Figura 5.2 : criar perfis de segurança .....	15
Figura 5.3 : método put de perfis de segurança.....	15
Figura 5.4 : método patch de perfis de segurança .....	16



Figura 5.5 : editar perfis de segurança.....	16
Figura 5.6 : apagar perfis de segurança .....	17
Figura 5.7 : método delete de perfis de segurança.....	17
Figura 6.1 : : redes wireless main page.....	18
Figura 6.2 : método patch para a rede wireless.....	18
Figura 6.3 : Ativar rede wireless.....	18
Figura 6.4 : Desativar rede wireless .....	19
Figura 6.5 : método patch para desativar a rede wireless .....	19
Figura 6.6 : método patch para configurar a rede wireless .....	20
Figura 6.7 : Configurar rede wireless .....	20
Figura 7.1: rotas estáticas main page.....	21
Figura 7.2 : listar rotas estáticas .....	21
Figura 7.3 : método get das rotas estáticas .....	21
Figura 7.4 : Criar rotas estáticas .....	22
Figura 7.5 : método put para criar rotas estáticas .....	22
Figura 7.6 : Editar rotas estáticas.....	23
Figura 7.7 : método patch para editar rotas estáticas .....	23
Figura 7.8 : Apagar rotas estáticas.....	24
Figura 7.9 : método delete para apagar rotas estáticas.....	24
Figura 8.1 : Endereços Ip main page .....	25
Figura 8.2 : Listar endereços IP .....	25
Figura 8.3 : código endereços IP .....	25
Figura 8.4 :Criar endereços IP .....	26
Figura 8.5 : : método put para editar endereços IP .....	26
Figura 8.6 : método patch para editar endereços IP.....	27
Figura 8.7 : Editar endereços IP .....	27
Figura 8.8 : Apagar endereços IP .....	28
Figura 8.9 : método delete para editar endereços IP.....	28
Figura 9.1: servidores de DHCP main page .....	29
Figura 9.2 : Listar servidores de DHCP.....	29

Figura 9.3 : : método get para servidores de DHCP .....	29
Figura 9.4 : Criar servidores de DHCP .....	30
Figura 9.5 : método put para servidores de DHCP.....	30
Figura 9.6 : método patch para servidores de DHCP .....	31
Figura 9.7 : editar servidores de DHCP .....	31
Figura 9.8 : Apagar servidores de DHCP.....	32
Figura 9.9 : método delete para servidores de DHCP .....	32
Figura 11.1 : Página DNS .....	33
Figura 11.2 : Ativar DNS .....	33
Figura 11.3 : Desativar DNS .....	34
Figura 11.4 : Configurar DNS.....	34
Figura 12.1 : Página de Firewall .....	35
Figura 12.2 : Listar firewall.....	35
Figura 12.3 : código Listar Firewall .....	36
Figura 12.4 : Criar Regra de Firewall.....	36
Figura 12.5 : Código criar regra Firewall .....	37
Figura 12.6 : Editar regra firewall .....	37
Figura 12.7 : Apagar regra firewall .....	38
Figura 13.1 : Página de protocolos de encaminhamento .....	39
Figura 13.2 : Criar Instância.....	39
Figura 13.3 : código para criar instância .....	40
Figura 13.4 : Criar área .....	40
Figura 13.5 : código para criar área.....	40
Figura 13.6 : Criar Interface Template .....	41
Figura 13.7 : código para criar Interface template .....	41
Figura 13.8 : Ativar Instância.....	41
Figura 13.9 : código para ativar instância .....	42
Figura 13.10 : desativar Instância.....	42
Figura 13.11 : código para desativar Instância .....	43
Figura 14.1 : criar VPN .....	44

Figura 14.2 : código para criar VPN.....	45
--	----

# Lista de Tabelas

Elemento a figurar, **quando aplicável**.

Tabela 2.1 - Texto ilustrativo da tabela 1..... **Error! Bookmark not defined.**

Tabela 4.1 - Texto ilustrativo da tabela 2..... **Error! Bookmark not defined.**

# 1. Introdução

O presente trabalho foi realizado no âmbito da unidade curricular de Laboratório de Tecnologias de Informação, do 3º ano do curso de Engenharia Informática, na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Para a implementação, foi necessário o desenvolvimento de uma SDN com a ajuda do Python para comunicar com o dispositivo através da sua API REST.

Uma SDN é uma forma de conceber a arquitetura de redes entre computadores, como o próprio nome sugere (Software Defined Network), ela utiliza software em vez de dispositivos especializados para gerenciar serviços de redes e aplicativos. Além de conferir maior mobilidade aos sistemas, viabiliza o fornecimento de aplicativos expansíveis, feitos sob demanda.

## 2. Fundamentação Teórica

Para a obtenção de conhecimento das técnicas de desenvolvimento referentes às tecnologias abordadas neste projeto, foram utilizados meios de pesquisa, com o objetivo de reunir conhecimento sobre as tecnologias nas áreas de programação e desenvolvimento.

As tecnologias pesquisadas foram:



- **Python:** Python é uma linguagem de programação que, segundo Van Rossum (2003), é multi-paradigma e gerenciada por uma organização sem fins lucrativos. O Python foi lançado em 1991 e a sua estrutura permite uma comunicação sólida com o equipamento, além de ótimos atributos para aplicações no ramo da matemática, eletrônica e física. Esta linguagem foi utilizada neste projeto para programar os componentes ligados ao Raspberry PI.



- **Postman:** Postman é uma ferramenta que dá suporte à documentação das requisições feitas pela API. Ele possui ambiente para a documentação, execução de testes de APIs e requisições em geral.

### 3. Bibliotecas usadas no Python

Foi utilizada a biblioteca tkinter, que permite a construção de uma aplicação labels, botões, etc..., para construir uma aplicação com interface gráfica.

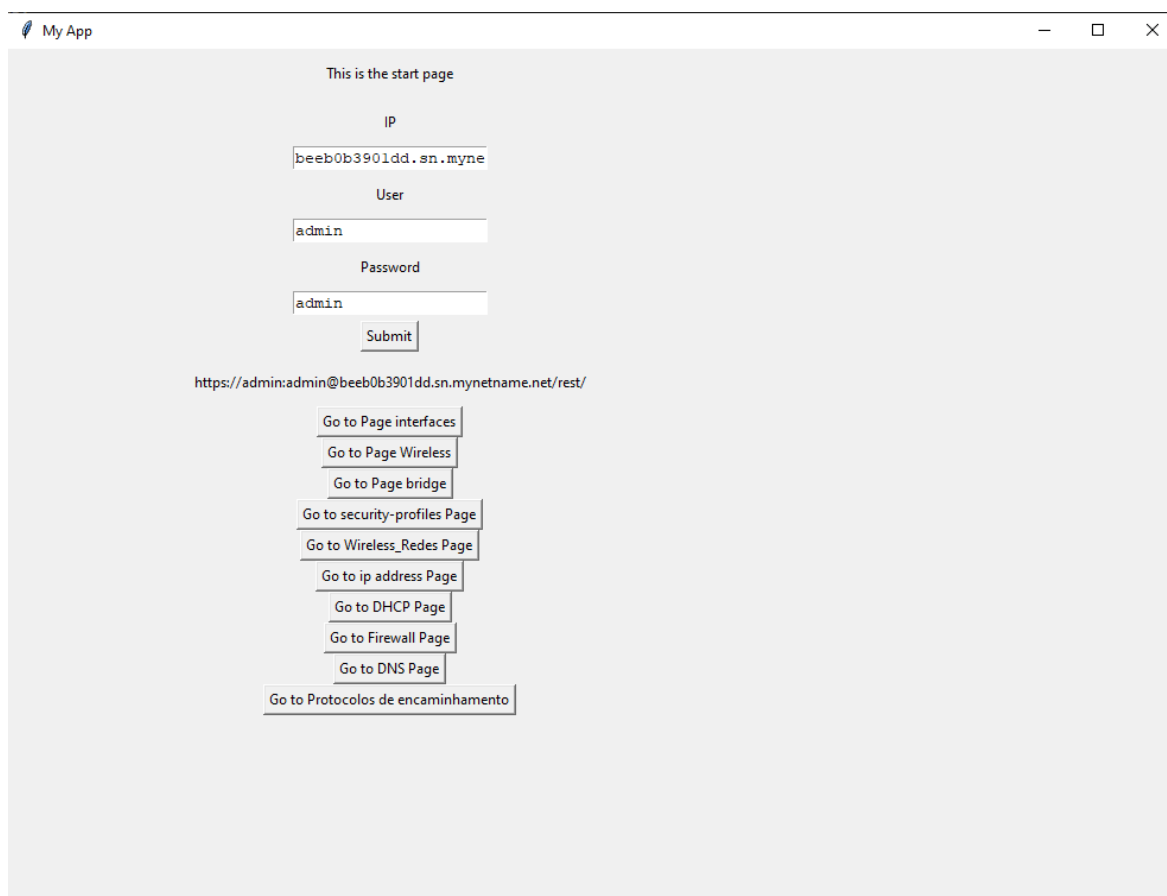


Figura 3.1 : main page

## 4. Ações disponibilizadas no projeto

Na nossa SDN existem os seguintes métodos:

- Controlar mais do que um dispositivo com a mesma aplicação
- Listar todas as interfaces do dispositivo
- Listar apenas as interfaces wireless
- Listar/criar/editar/apagar interfaces bridge e respectivas portas associadas
- Criar/editar/apagar perfis de segurança para utilizar nas redes wireless
- Ativar/desativar/configurar redes wireless
- Listar/criar/editar/apagar rotas estáticas
- Listar/criar/editar/apagar endereços IP
- Listar/criar/editar/apagar servidores de DHCP
- Ativar/desativar/configurar o servidor de DNS
- Listar/criar/editar/apagar regras de firewall
- 

### 4.1. Controlar mais do que um dispositivo com a mesma aplicação

Para controlar mais do que um MikroTik a partir da SDN foi acrescentado na página principal 3 Labels, uma para inserir o IP/Domain e as outras 2 para o login e a password.



Figura 4.2 : IP, User, Password

```
def submit_text():
    """Update the display label with the entered text"""
    global ip
    ip = ip_Text.get("1.0", "end-1c")
    user = user_Text.get("1.0", "end-1c")
    password = password_Text.get("1.0", "end-1c")
    global url
    url = "https://" + user + ":" + password + "@" + ip + "/rest/"
    StartPage_display_text.config(text=url)
```

Figura 4.1 : Concatenação das variáveis para URL



## 4.2. Listar todas as interfaces do dispositivo

Para listar todas as interfaces foi criada uma página á parte chamada “PageInterfaces” que irá mostrar todas as interfaces existentes.

Nessa tal página foi criado um botão que ao ser pressionado irá mostrar todas a interfaces existentes no MikroTik como demonstrado na figura 4.3.

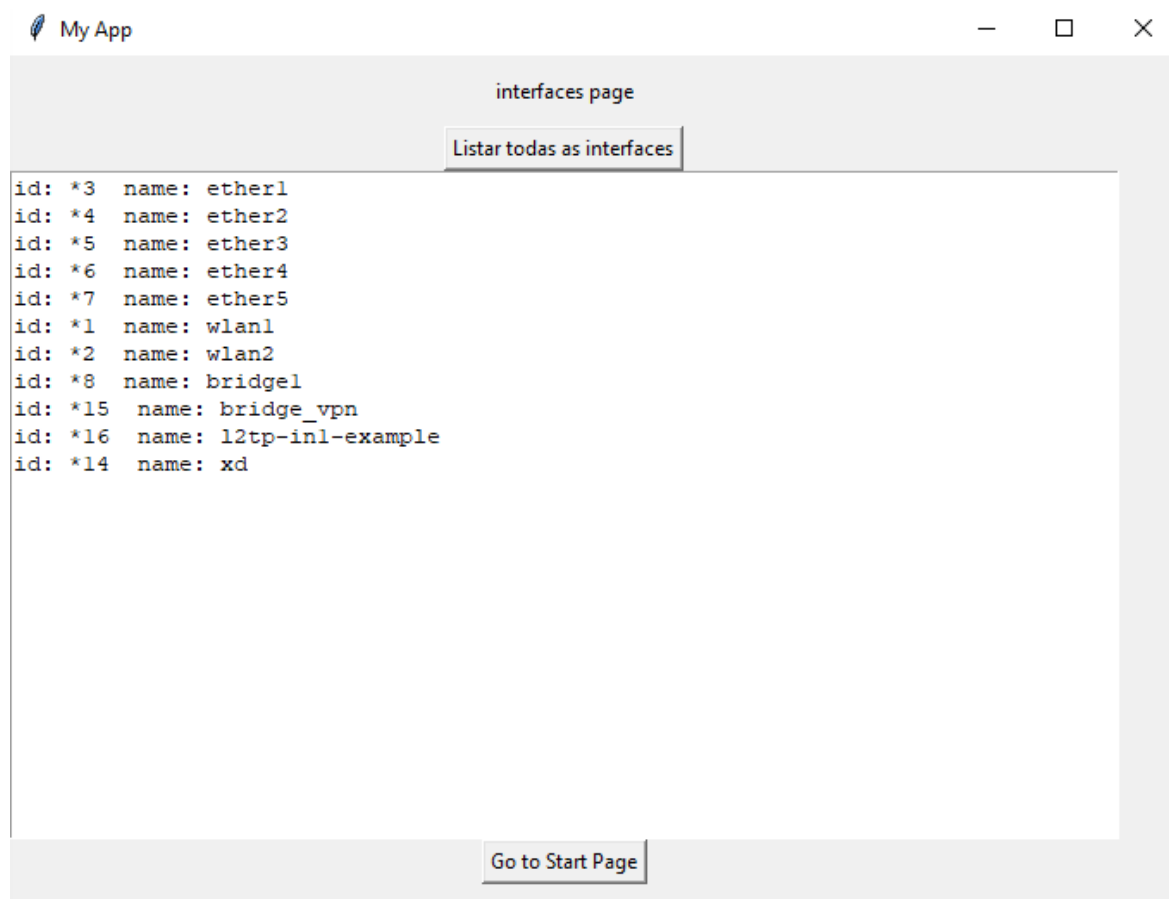


Figura 4.3 : interfaces

```
def listar_intefaces():
    global url

    response = requests.get(url+"interface",verify=False)
    response_data = response.json()
    devolver = ""

    for i in range(len(response_data)):
        devolver = devolver + ("id: "+response_data[i]['.id'] +" name: "+ response_data[i]['name']+"\n")
    print(devolver)

    response_label.delete('1.0', 'end')
    response_label.insert('end', devolver)
```

Figura 4.4 : método get das interfaces

### 4.3. Listar apenas as interfaces wireless

Para listar todas as interfaces Wireless foi criada uma página á parte chamada “PageWireless” que irá mostrar todas as interfaces Wireless existentes.

Nessa tal página foi criado um botão que ao ser pressionado irá mostrar todas a interfaces Wireless existentes no MikroTik como está representado na figura 4.5.

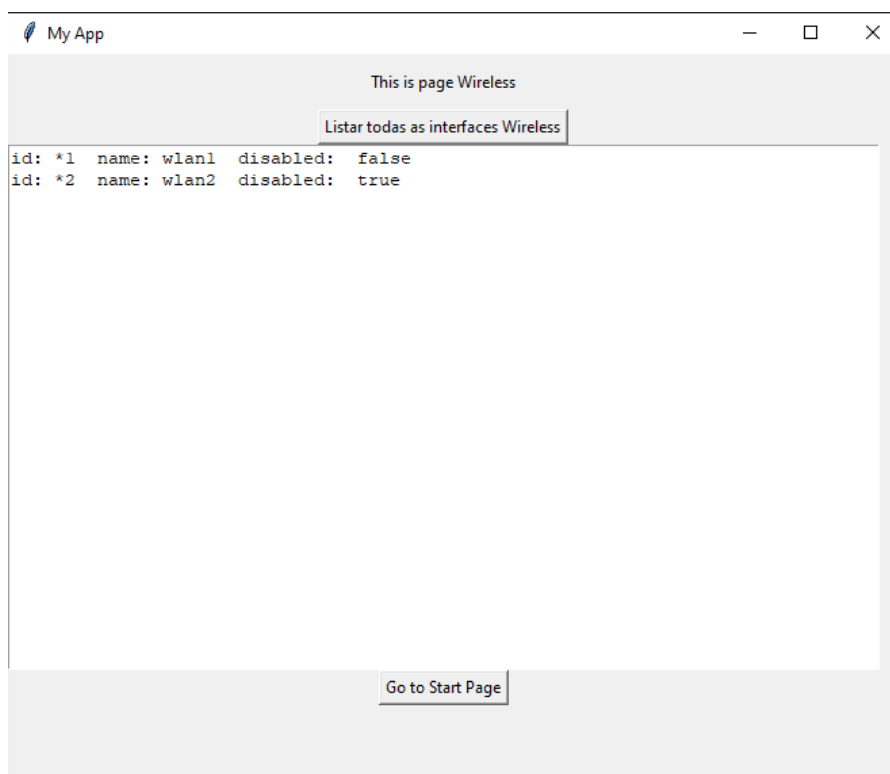


Figura 4.5 : Interface Wireless

```
def listar_intefaces_wireless():
    global url
    response = requests.get(url+"interface/wireless",verify=False)
    response_data = response.json()
    devolver = ""

    for i in range(len(response_data)):
        devolver = devolver + ("id: "+response_data[i]['.id'] +" name: "+ response_data[i]['name']+" disabled: "+ response_data[i]['disabled']+"\\n")
    response_label_wireless.delete('1.0', 'end')
    response_label_wireless.insert('end', devolver)
```

Figura 4.6 : método get das interfaces Wireless

## 4.4. Listar/criar/editar/apagar interfaces bridge e respectivas portas associadas

Para manipular as interfaces bridge todas temos uma página só para elas o que ira facilitar o uso SDN.

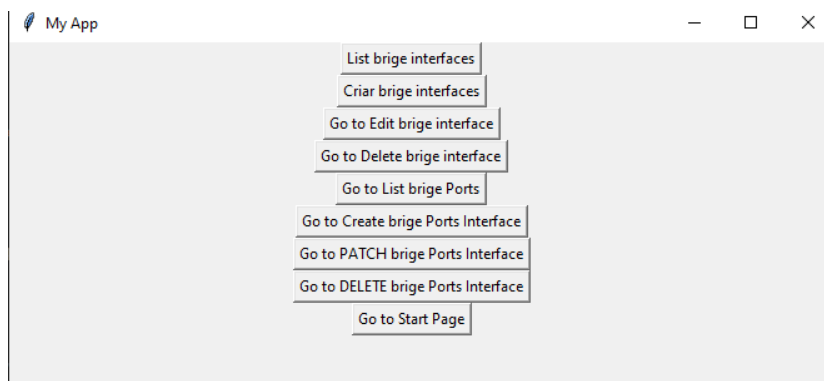
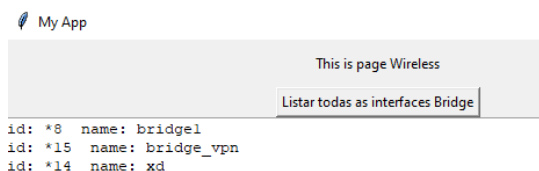


Figura 4.7 : interface bridge

### 4.4.1. Listar interfaces bridge

Para listar todas as bridge's foi criada uma página a parte que ira mostrar todas as interfaces bridge existentes ao pressionar no botão dedicado para o mesmo.



```
def listar_intefaces_Bridge():
    global url
    response = requests.get(url+"interface/bridge",verify=False)
    response_data = response.json()
    devolver = ""
    for i in range(len(response_data)):
        devolver = devolver + ("id: "+response_data[i]['.id'] + " name: "+ response_data[i]['name']+"\n")
    response_label_bridges.delete('1.0', 'end')
    response_label_bridges.insert('end', devolver)
```

Figura 4.8 : método get para listar interfaces bridge

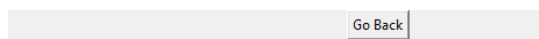


Figura 4.9 : Listar interfaces bridge

#### 4.4.2. Criar interfaces bridge

Na página dedicada para criar as interfaces bridge temos uma Label para inserirmos o nome da interface bridge e ao submetermos irá criá-la, sendo possível obter um resultado de sucesso (200) no CMD.

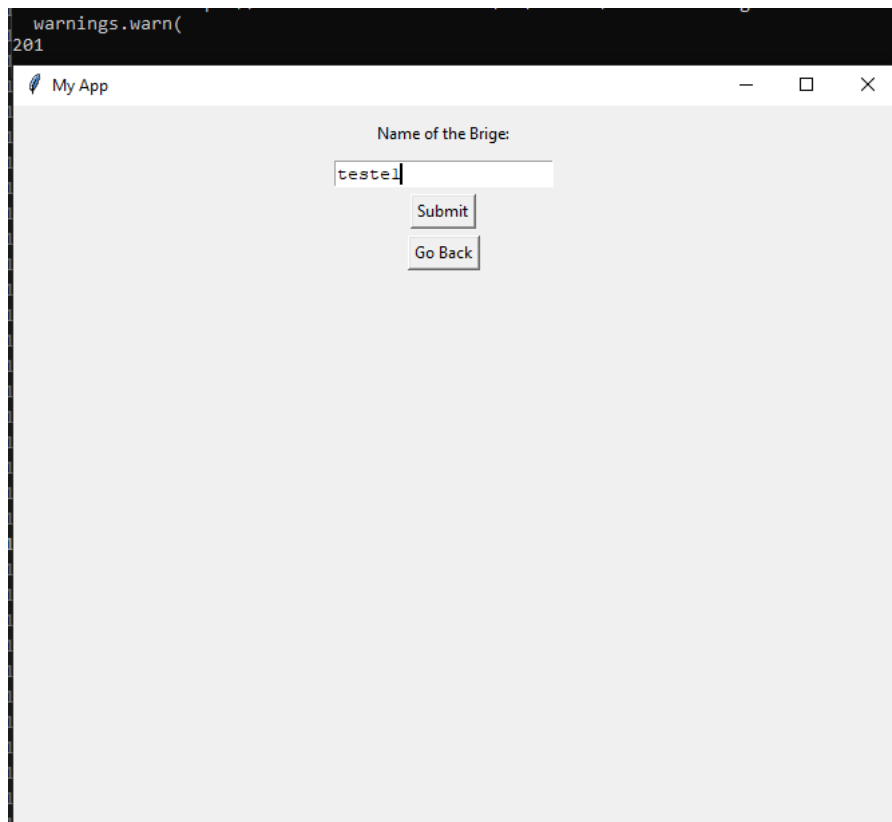


Figura 4.11 : Criar interface Bridge

```
def submit_Brige():
    global url

    Name = Brige_Name_Text.get("1.0", "end-1c")
    payload = {"name": Name}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.put(url+"interface/bridge", headers=headers, data=json_payload, verify=False)
    response_data = response.json()

    print(response.status_code)
```

Figura 4.10 : método put da interface bridge

### 4.4.3. Editar interfaces bridge

Na página dedicada para editar as interfaces bridge temos a primeira Label onde podemos inserir o nome da interface que queremos mudar e a segunda serve para qual nome queremos mudar, ao clicarmos no submit isso irá fazer um get para ver a qual ID pertence o tal nome e só depois vai enviar um patch para mudar nome com a ajuda do ID.

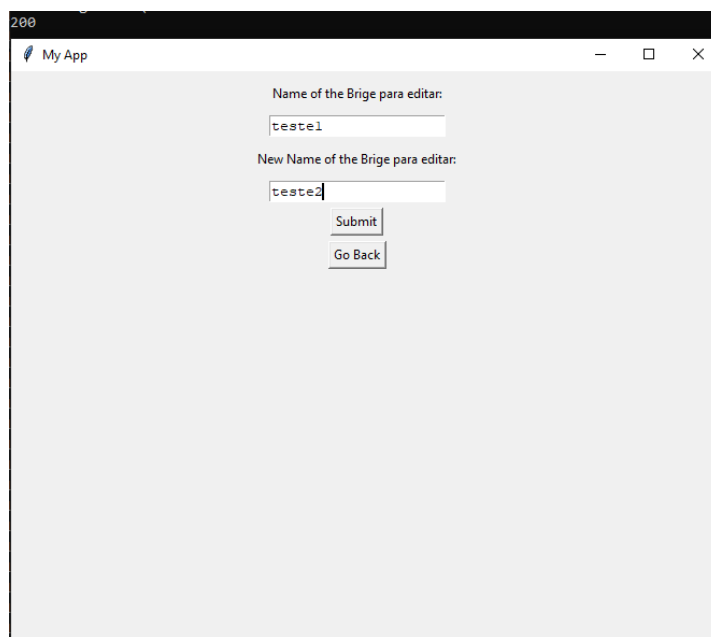


Figura 4.12 : Editar interfaces bridge

```
def Editar_intefaces_Bridge():
    global url
    response = requests.get(url+"interface/bridge",verify=False)
    response_data = response.json()
    Name = Brige_Name_Text_Get.get("1.0","end-1c")
    #print(Name)
    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name:
            ID_a_mudar = response_data[i]['.id']
            print(ID_a_mudar)

    Name = Brige_Name_Text_Edit.get("1.0","end-1c")
    payload = {"name": Name}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"interface/bridge/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 4.13 : método patch para a interface bridge

#### 4.4.4. Apagar interfaces bridge

Na página dedicada para apagar as interfaces bridge temos uma Label onde podemos inserir o nome da interface que queremos apagar ao submetermos isso irá fazer um get para ver a qual ID pertence o tal nome e só depois vai apagar a interface como mostra na figura 4.14.

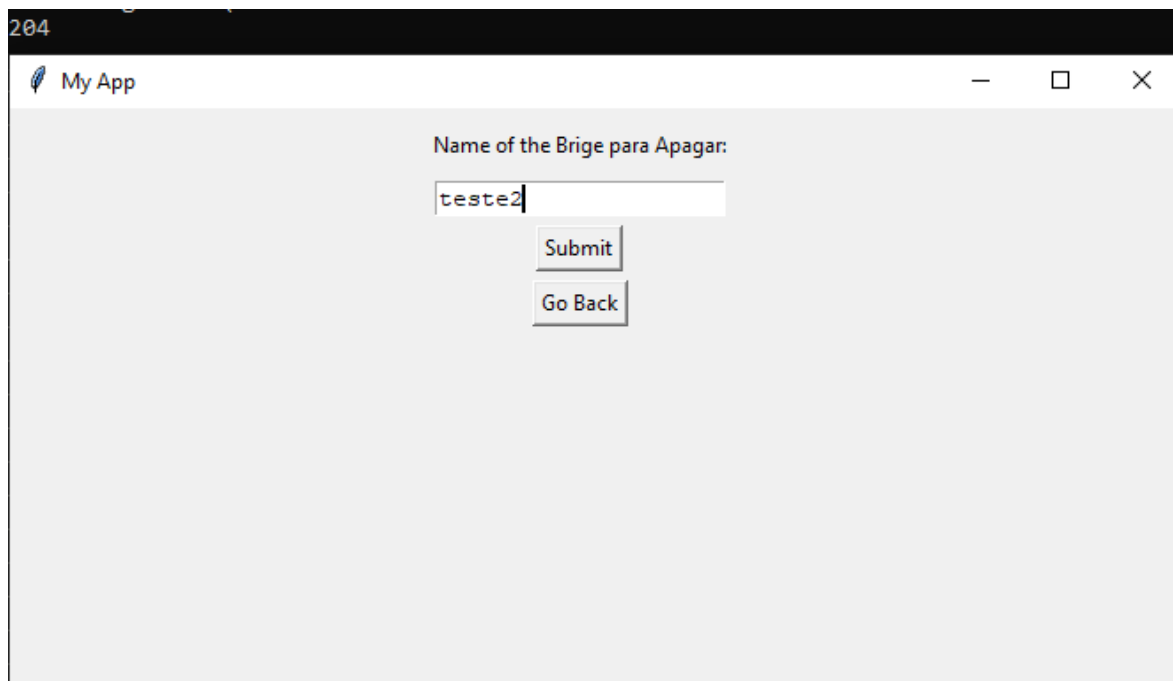


Figura 4.15 : Apagar interfaces bridge

```
def Delete_intefaces_Bridge():  
    global url  
    response = requests.get(url+"interface/bridge",verify=False)  
    response_data = response.json()  
    Name = Brige_Name_Text_Get_Delete.get("1.0","end-1c")  
    ID_a_mudar = ""  
    for i in range(len(response_data)):  
        if response_data[i]['name'] == Name:  
            ID_a_mudar = response_data[i]['.id']  
    response = requests.delete(url+"interface/bridge/"+ID_a_mudar,verify=False)  
    print(response.status_code)
```

Figura 4.14 : método delete para a interface bridge

#### 4.4.5. Listar portas das interfaces bridge

Nesta página podemos listar as portas das interfaces bridge ao pressionarmos no botão dedicado o que irá executar um método get para as listar na nossa SDN.

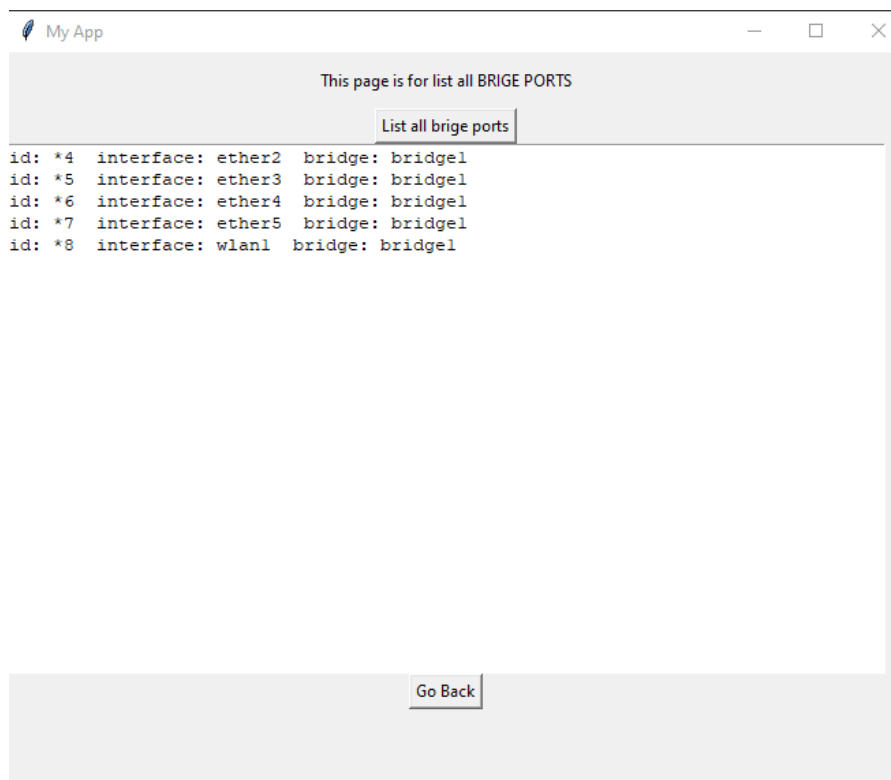


Figura 4.16 : : Listar portas das interfaces bridge

```
def listar_Bridge_Ports():  
    global url  
    response = requests.get(url+"interface/bridge/port",verify=False)  
    response_data = response.json()  
    devolver = ""  
    for i in range(len(response_data)):  
        devolver = devolver + ("id: "+response_data[i]['.id'] + " interface: "+ response_data[i]['interface']+ " bridge: "+ response_data[i]['bridge']+"\n")
```

Figura 4.17 : método get para listar portas das interfaces bridge

#### 4.4.6. Criar portas para as interfaces bridge

Esta página é dedicada para criar as portas das interfaces bridge, para tal temos a primeira Label onde podemos inserir o nome da interface, e a segunda Label onde podemos inserir o nome da Bridge como se pode ver na figura 4.18.

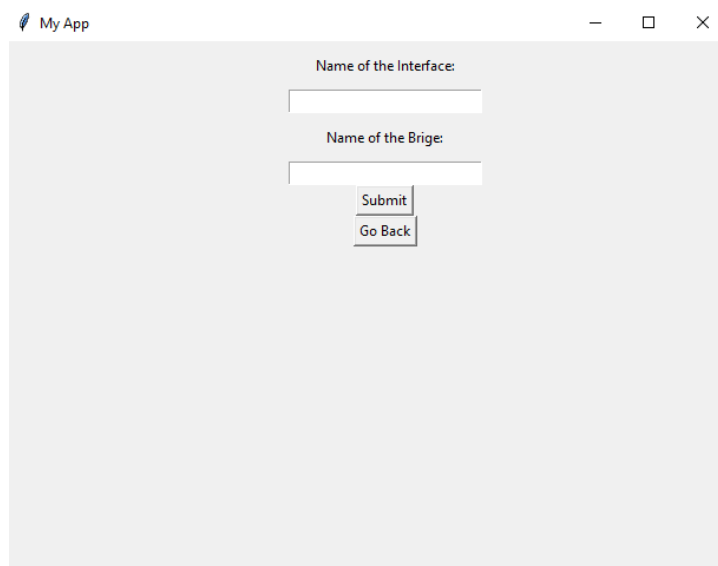


Figura 4.18 : Criar porta para a interface Bridge

```
def submit_Brige_Port():  
    """Update the display label with the entered text"""  
    global url  
    interface = Port_Interface_Name_Text.get("1.0", "end-1c")  
    bridge = Port_Brige_Name_Text.get("1.0", "end-1c")  
    payload = {"interface": interface, "bridge": bridge}  
    json_payload = json.dumps(payload)  
    headers = {"Content-Type": "application/json"}  
    response = requests.put(url+"interface/bridge/port", headers=headers, data=json_payload, verify=False)  
    response_data = response.json()  
    print(response.status_code)
```

Figura 4.19 : método put das portas bridge



#### 4.4.7. Editar portas bridge

Na página dedicada para editar as portas bridge, temos a primeira Label onde podemos inserir o nome da porta que queremos mudar, e a segunda serve para qual nome queremos mudar, ao clicarmos no botão submit será feito um get para ver a qual ID pertence o tal nome e só depois vai enviar um patch para mudar nome com a ajuda do ID.

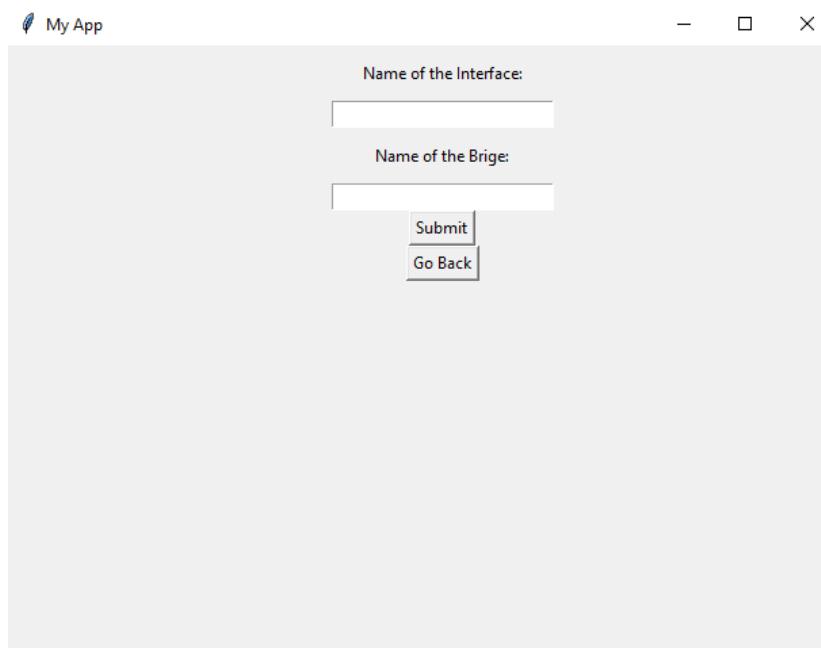


Figura 4.20 : Editar porta Bridge

```
def Submit_Brige_Port_Patch():
    """Update the display label with the entered text"""
    global url
    response = requests.get(url+"interface/bridge/port",verify=False)
    response_data = response.json()
    interface = Port_Interface_Name_Text_Patch.get("1.0","end-1c")
    bridge = Port_Brige_Name_Text_Patch.get("1.0","end-1c")
    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['interface'] == interface:
            ID_a_mudar = response_data[i]['.id']
    payload = {"bridge":bridge}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"interface/bridge/port/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 4.21: método patch das portas bridge

#### 4.4.8. Apagar portas bridge

Na página dedicada para apagar as interfaces bridge temos uma Label onde podemos inserir o nome da interface que queremos apagar ao submetermos isso irá fazer um get para ver a qual ID pertence o tal nome e só depois vai apagar a interface como mostra na figura 4.22.

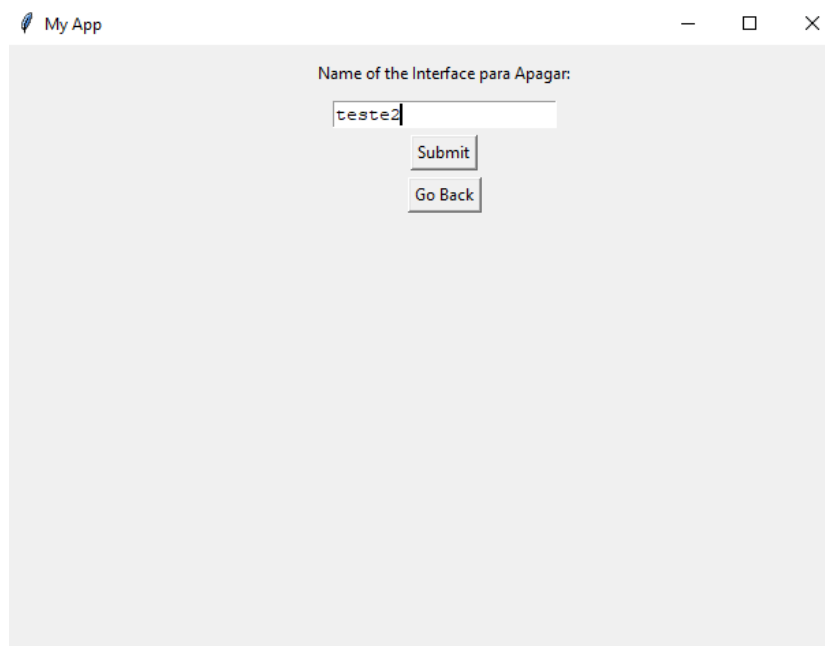


Figura 4.22 : Apagar porta bridge

```
def Delete_intefaces_Bridge_Port():
    global url
    response = requests.get(url+"interface/bridge/port",verify=False)
    response_data = response.json()
    Name = Interface_Name_Text_Get_Delete.get("1.0","end-1c")
    ID_a_apagar = ""
    for i in range(len(response_data)):
        if response_data[i]['interface'] == Name:
            ID_a_apagar = response_data[i]['.id']
            print(ID_a_apagar)
            print(url+"interface/bridge/port"+ID_a_apagar)
    response = requests.delete(url+"interface/bridge/port/"+ID_a_apagar,verify=False)
    print(response.status_code)
```

Figura 4.23 : método delete para portas bridge

## 5. Criar/editar/apagar perfis de segurança para utilizar nas redes wireless

Para manipular todos os perfis de segurança foi criada uma página á parte que irá servir para criar/editar/apagar perfis de segurança com a ajuda da SDN como mostra a figura

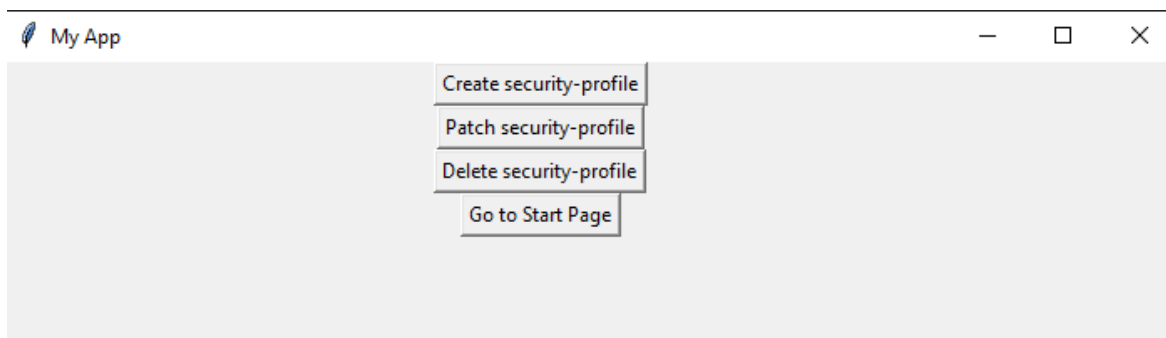


Figura 5.1 : perfis de segurança main page

### 5.1.1. Criar perfis de segurança

Na página dedicada para criar perfis de segurança temos uma Label para inserirmos o nome do perfil e outra para tipo de autenticação ao pressionarmos o botão submit e enviado um método put que nos faz obter um resultado de sucesso (200) no CMD.

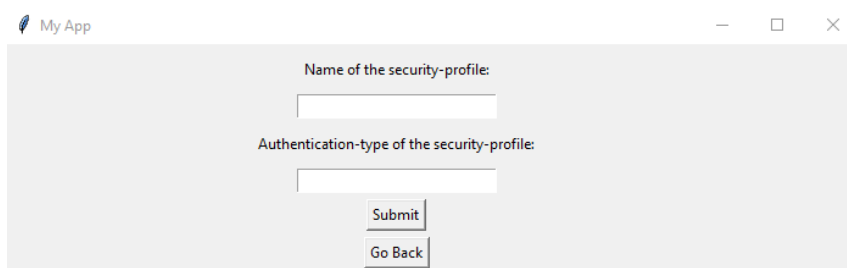


Figura 5.2 : criar perfis de segurança

```
def submit_SecurityProfile():
    """Update the display label with the entered text"""
    global url
    Name = Security_Profile_Name_Text.get("1.0", "end-1c")
    Authentication_Type = Authentication_Type_Text.get("1.0", "end-1c")
    print(Name)
    print(Authentication_Type)
    payload = {"name": Name, "authentication-types": Authentication_Type}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.put(url+"interface/wireless/security-profiles", headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 5.3 : método put de perfis de segurança

### 5.1.2. Editar perfis de segurança

Na página dedicada para editar os perfis de segurança temos a primeira Label onde podemos inserir o nome dos perfis de segurança que queremos mudar e a segunda serve para qual nome queremos mudar, ao clicarmos no submit isso irá fazer um get para ver a qual ID pertence o tal nome e só depois vai enviar um patch para mudar nome com a ajuda do ID.



Figura 5.5 : editar perfis de segurança

```
def Submit_Security_Profile_Patch():
    global url
    response = requests.get(url+"interface/wireless/security-profiles",verify=False)
    response_data = response.json()
    Name_Text_To_Patch = Security_Profile_Name_Text_To_Patch.get("1.0","end-1c")
    Name_Text_Patch = Port_Bridge_Name_Text_Patch.get("1.0","end-1c")

    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name_Text_To_Patch:
            ID_a_mudar = response_data[i]['.id']
            print(ID_a_mudar)
    payload = {"name":Name_Text_Patch}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"interface/wireless/security-profiles/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 5.4 : método patch de perfis de segurança

### 5.1.3. Apagar perfis de segurança

Na página dedicada para apagar perfis de segurança temos uma Label onde podemos inserir o nome do perfil de segurança que queremos apagar ao submetermos isso irá apagar o mesmo.

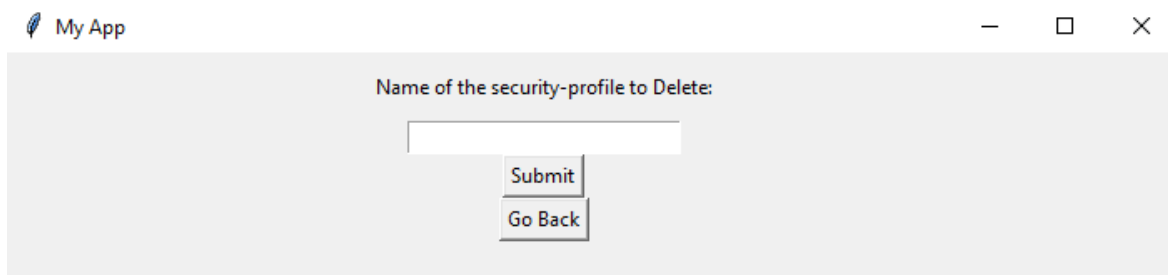


Figura 5.6 : apagar perfis de segurança

```
def Submit_Security_Profile_Delete():
    global url
    response = requests.get(url+"interface/wireless/security-profiles",verify=False)
    response_data = response.json()
    Name = Security_Profile_Name_Text_To_Delete.get("1.0","end-1c")
    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name:
            ID_a_mudar = response_data[i]['.id']
    response = requests.delete(url+"interface/wireless/security-profiles/"+ID_a_mudar,verify=False)
    print(response.status_code)
def delete_Security_Profile_Delete():
    Security_Profile_Name_Text_To_Delete.delete("1.0","end-1c")
```

Figura 5.7 : método delete de perfis de segurança

## 6. Ativar/desativar/configurar redes wireless

Para manipular todas as redes wireless foi criada uma página á parte que irá servir para ativar/desativar/configurar redes wireless com a ajuda da SDN como mostra a figura



Figura 6.1 : : redes wireless main page

### 6.1. Ativar redes wireless

Na página dedicada para ativar as redes wireless temos uma Label onde podemos inserir o nome, de seguida ao pressionarmos o botão submit irá fazer um patch que irá ativar a rede wireless.

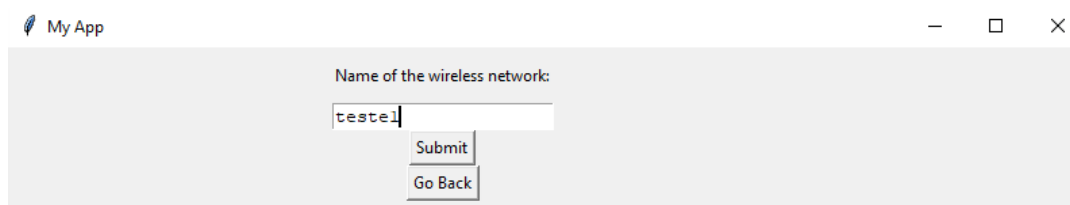


Figura 6.3 : Ativar rede wireless

```
def Submit_Wireless_Redes_Ativar():
    """Update the display label with the entered text"""
    global url
    response = requests.get(url+"interface/wireless",verify=False)
    response_data = response.json()
    #print(response_data)

    Name_Text_To_Patch = Wireless_Name_Text.get("1.0", "end-1c")

    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name_Text_To_Patch:
            ID_a_mudar = response_data[i]['.id']
            print(ID_a_mudar)

    payload = {"disabled": "no"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"interface/wireless/"+ID_a_mudar, headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 6.2 : método patch para a rede wireless

## 6.2.Desativar redes wireless

Na página dedicada para desativar as redes wireless temos uma Label onde podemos inserir o nome, de seguida ao pressionarmos o botão submit irá fazer um patch que irá desativar a rede wireless.

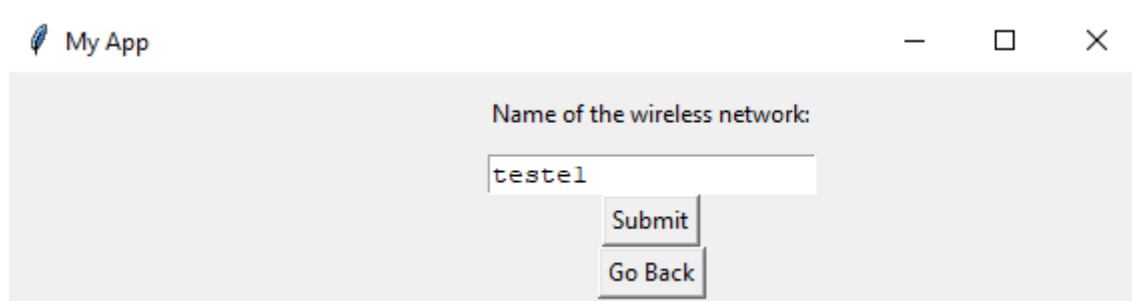


Figura 6.4 : Desativar rede wireless

```
def Submit_Wireless_Redes_Desativar():
    """Update the display label with the entered text"""
    global url
    response = requests.get(url+"interface/wireless",verify=False)
    response_data = response.json()
    Name_Text_To_Patch = Wireless_Name_Text_Desativar.get("1.0","end-1c")
    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name_Text_To_Patch:
            ID_a_mudar = response_data[i]['.id']
            print(ID_a_mudar)

    payload = {"disabled":"yes"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"interface/wireless/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 6.5 : método patch para desativar a rede wireless

### 6.3. Configurar redes wireless

Na página dedicada para configurar as redes wireless temos a primeira Label, onde podemos inserir o nome da rede que queremos mudar, a segunda serve para mudar o Mode da wireless network, de seguida vai o SSID e por fim temos o security profile que queremos mudar por fim ao darmos submit isso irá fazer um patch irá fazer as mudanças necessárias.

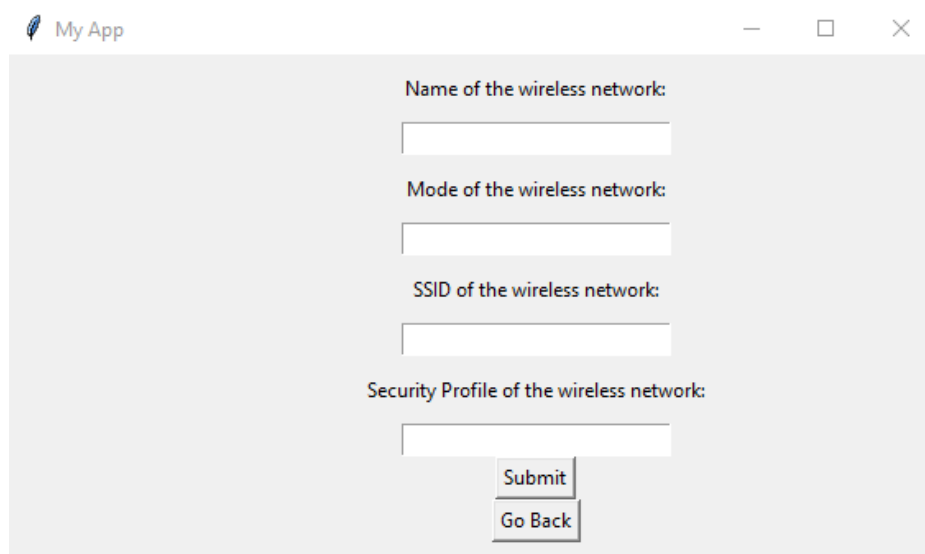


Figura 6.7 : Configurar rede wireless

```
def Submit_Wireless_Redes_Configurar():
    """Update the display label with the entered text"""
    global url
    response = requests.get(url+"interface/wireless",verify=False)
    response_data = response.json()
    Name_Text_Name = Wireless_Name_Text_Name.get("1.0","end-1c")
    Name_Text_Mode = Wireless_Name_Text_Mode.get("1.0","end-1c")
    Name_Text_SSID = Wireless_Name_Text_SSID.get("1.0","end-1c")
    Name_Text_Security_Profile = Wireless_Name_Text_Security_Profile.get("1.0","end-1c")
    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name_Text_Name:
            ID_a_mudar = response_data[i]['.id']
            if Name_Text_Mode == "":
                Name_Text_Mode = response_data[i]['mode']
                print(Name_Text_Mode)
            if Name_Text_SSID == "":
                Name_Text_SSID = response_data[i]['ssid']
                print(Name_Text_SSID)
            if Name_Text_Security_Profile == "":
                Name_Text_Security_Profile = response_data[i]['security-profile']
                print(Name_Text_Security_Profile)
    payload = {"mode":Name_Text_Mode,"ssid":Name_Text_SSID,"security-profile":Name_Text_Security_Profile}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"interface/wireless/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 6.6 : método patch para configurar a rede wireless



## 7. Listar/criar/editar/apagar rotas estáticas

Para manipular todas as rotas estáticas foi criada uma página á parte que irá servir para Listar/criar/editar/apagar rotas estáticas com a ajuda da SDN como mostra a figura

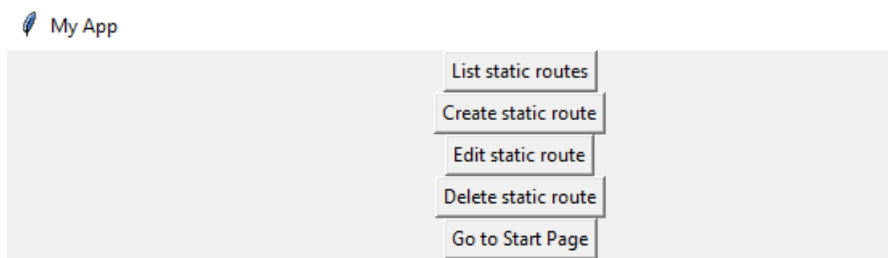


Figura 7.1: rotas estáticas main page

### 7.1.Listar rotas estáticas

Nesta página podemos listar as rotas estáticas ao pressionarmos no botão dedicado o que irá executar um método get para as listar na nossa SDN.

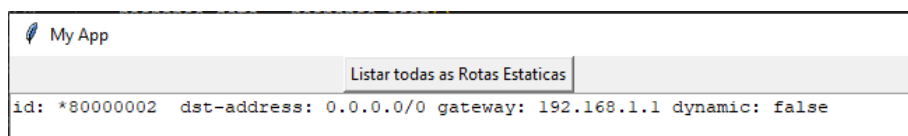


Figura 7.2 : listar rotas estáticas

```
def listar_Rotas_Estaticas():
    global url
    response = requests.get(url+"ip/route?static=yes",verify=False)
    response_data = response.json()
    devolver = ""
    for i in range(len(response_data)):
        devolver = devolver + ("id: "+response_data[i]['.id'] + " dst-address: "+ response_data[i]['dst-address']+
        " gateway: "+response_data[i]['gateway']+" dynamic: "+response_data[i]['dynamic']+"\\n")
    Response_Label_Rotas_Estaticas.delete('1.0', 'end')
    Response_Label_Rotas_Estaticas.insert('end', devolver)
```

Figura 7.3 : método get das rotas estáticas

## 7.2. Criar rotas estáticas

Na página dedicada para criar rotas estáticas temos uma Label para inserirmos o Dst. Address e outra para a Gateway, por fim ao pressionarmos o botão submit é enviado um método put que nos faz obter um resultado de sucesso (200) no CMD

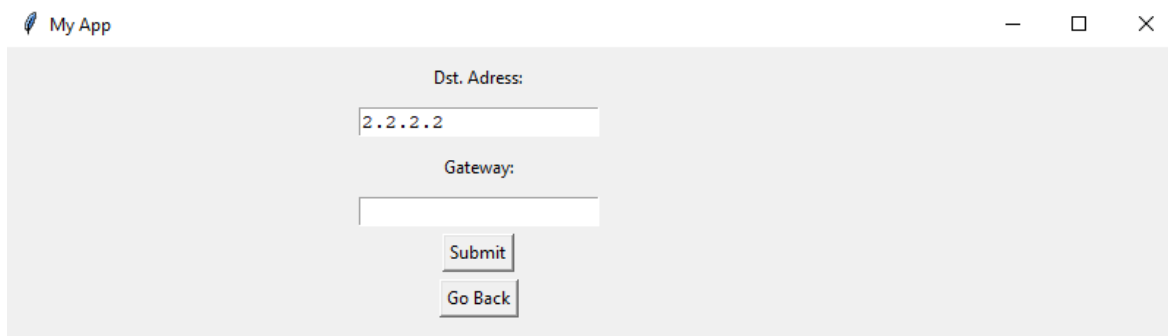


Figura 7.4 : Criar rotas estáticas

```
def submit_PageStatic_routes_Criar():
    """Update the display label with the entered text"""
    global url

    Dst_Address = Static_routes_Dst_Address_Text.get("1.0", "end-1c")
    Gateway = Static_routes_Gateway_Text.get("1.0", "end-1c")

    payload = {"dst-address": Dst_Address, "gateway": Gateway}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.put(url+"ip/route?static=yes", headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 7.5 : método put para criar rotas estáticas

### 7.3. Editar rotas estáticas

Na página dedicada para editar as rotas estáticas temos a primeira Label, onde podemos inserir o ID da rota estática que queremos mudar, a segunda serve para mudarmos o nome da Dst. Address e por fim o nome do Gateway, ao clicarmos no submit isso irá fazer um patch dos campos que foram alterados.

Figura 7.6 : Editar rotas estáticas

```
def submit_PageStatic_routes_Editar():
    """Update the display label with the entered text"""
    global url
    response = requests.get(url+"ip/route?static=yes",verify=False)
    response_data = response.json()
    Id_Text = PageStatic_routes_Editar_Id_Text.get("1.0","end-1c")
    Dst_Adress_Text = PageStatic_routes_Editar_Dst_Adress_Text.get("1.0","end-1c")
    Gateway_Text = PageStatic_routes_Editar_Gateway_Text.get("1.0","end-1c")

    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['.id'] == Id_Text:
            ID_a_mudar = response_data[i]['.id']
            if Dst_Adress_Text == "":
                Dst_Adress_Text = response_data[i]['dst-address']
                print(Dst_Adress_Text)
            if Gateway_Text == "":
                Gateway_Text = response_data[i]['gateway']
                print(Gateway_Text)
    payload = {"dst-address":Dst_Adress_Text,"gateway":Gateway_Text}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"ip/route/"+ID_a_mudar+"?static=yes",headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 7.7 : método patch para editar rotas estáticas

## 7.4. Apagar rotas estáticas

Na página dedicada para apagar as rotas estáticas temos uma Label onde podemos inserir o ID da rota estática que queremos apagar, ao submetermos isso irá fazer um delete que vai apagar a interface como mostra na figura 5.6

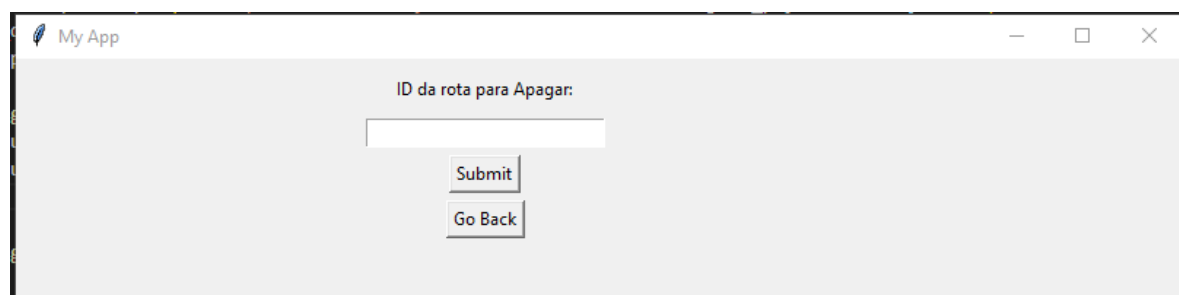


Figura 7.8 : Apagar rotas estáticas

```
def Submit_PageStatic_Delete():  
    global url  
    ID_Text = Static_routes_Apagar_ID_Text.get("1.0", "end-1c")  
    response = requests.delete(url+"ip/route/"+ID_Text+"?static=yes", verify=False)  
    response_data = response.json()  
    print(response.status_code)
```

Figura 7.9 : método delete para apagar rotas estáticas

## 8. Listar/criar/editar/apagar endereços IP

Para manipular todos os endereços IP foi criada uma página á parte que irá servir para Listar/criar/editar/apagar rotas estáticas com a ajuda do SDN como mostra a figura

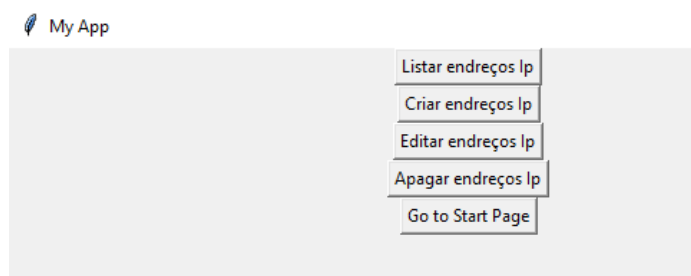


Figura 8.1 : Endereços Ip main page

### 8.1.Listar endereços IP

Nesta página podemos listar os endereços IP ao pressionarmos no botão dedicado o que irá executar um método get para as listar na nossa SDN.

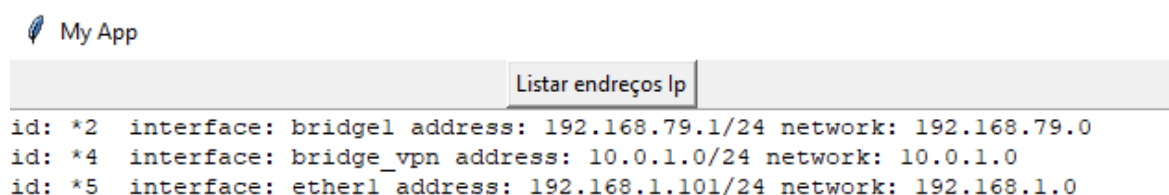


Figura 8.2 : Listar endereços IP

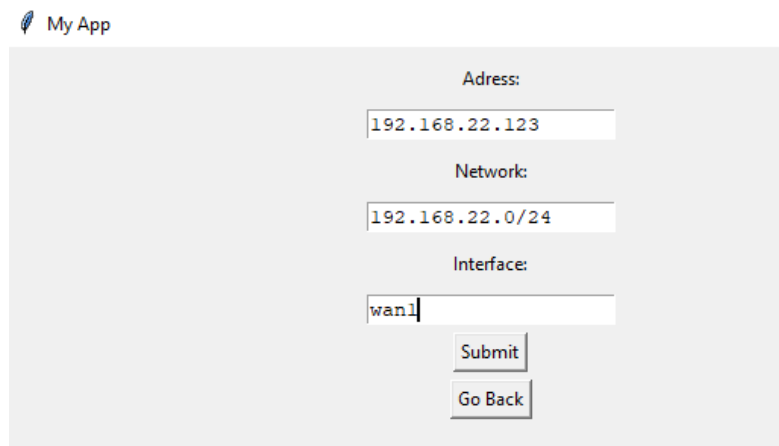
```
def listar_Ip_Address():
    global url
    response = requests.get(url+"ip/address",verify=False)
    response_data = response.json()
    devolver = ""
    for i in range(len(response_data)):
        devolver = devolver + ("id: "+response_data[i]['id']+" interface: "+ response_data[i]['interface']+" address: "
        +response_data[i]['address']+" network: "+response_data[i]['network']+"\n")

    Response_Label_Rotas_Estaticas_ip.delete('1.0', 'end')
    Response_Label_Rotas_Estaticas_ip.insert('end', devolver)
```

Figura 8.3 : código endereços IP

## 8.2.Criar endereços IP

Na página dedicada para criar endereços IP temos uma Label para inserirmos o Address de seguida vem a Network e a última serve para inserirmos a Interface, por fim ao pressionarmos o botão submit é enviado um método put que nos faz obter um resultado de sucesso (200) no CMD



My App

Address:  
192.168.22.123

Network:  
192.168.22.0/24

Interface:  
wan1

Submit

Go Back

Figura 8.4 :Criar endereços IP

```
def submit_PageIp_Address_Criar():
    global url
    Address_Text = PageIp_Address_Criar_Address_Text.get("1.0", "end-1c")
    Network_Text = PageIp_Address_Criar_Network_Text.get("1.0", "end-1c")
    Interface_Text = PageIp_Address_Criar_Interface_Text.get("1.0", "end-1c")

    payload = {"address":Address_Text,"network":Network_Text,"interface":Interface_Text}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.put(url+"ip/address",headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 8.5 : : método put para editar endereços IP

### 8.3.Editar endereços IP

Na página dedicada para editar os endereços IP temos a primeira Label, onde podemos inserir o ID da rota estática que queremos mudar, a segunda serve para mudarmos o Endereço a próxima serve para mudar a Network e por fim a Interface, ao clicarmos no submit isso irá fazer um patch dos campos que foram alterados.

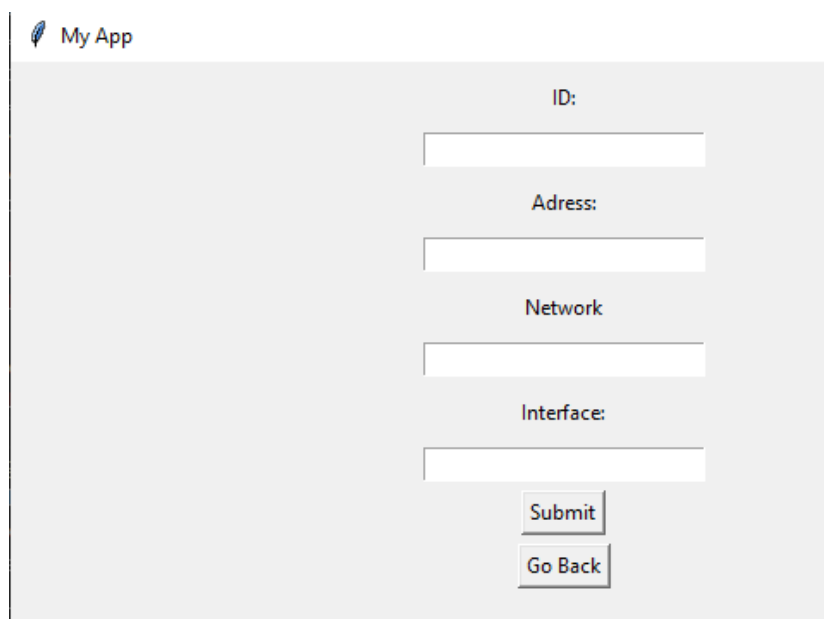


Figura 8.7 : Editar endereços IP

```
def submit_PageIp_Address_Editar():
    global url
    response = requests.get(url+"ip/address",verify=False)
    response_data = response.json()
    Id_Text = PageIp_Address_Editar_Id_Text.get("1.0","end-1c")
    Address_Text = PageIp_Address_Editar_Address_Text.get("1.0","end-1c")
    Network_Text = PageIp_Address_Editar_Network_Text.get("1.0","end-1c")
    Interface_Text = PPageIp_Address_Editar_Interface_Text.get("1.0","end-1c")
    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['.id'] == Id_Text:
            ID_a_mudar = response_data[i]['.id']
            print(ID_a_mudar)
            if Address_Text == "":
                Address_Text = response_data[i]['address']
                print(Address_Text)
            if Network_Text == "":
                Network_Text = response_data[i]['network']
                print(Network_Text)
            if Interface_Text == "":
                Interface_Text = response_data[i]['interface']
                print(Interface_Text)
    payload = {"address":Address_Text,"network":Network_Text,"interface":Interface_Text}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"ip/address/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 8.6 : método patch para editar endereços IP

## 8.4. Apagar endereços IP

Na página dedicada para apagar os endereços IP temos uma Label onde podemos inserir o ID dos endereços IP que queremos apagar, ao submetermos isso irá fazer um delete que vai apagar o endereço IP como mostra na figura 8.8

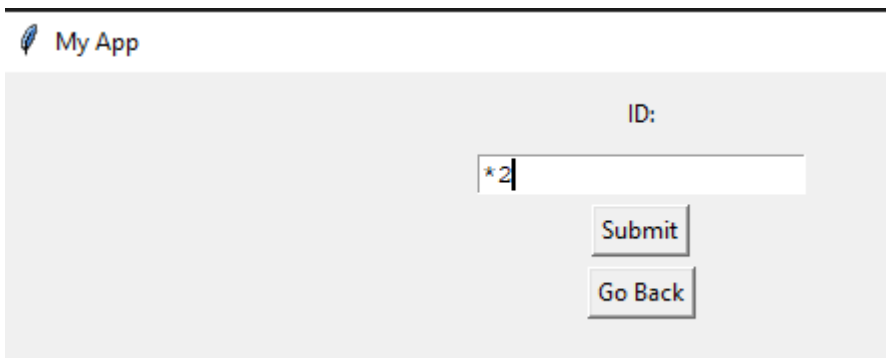


Figura 8.8 : Apagar endereços IP

```
def Submit_PageIp_Address_Apagar():  
    global url  
    ID_Text = PageIp_Address_Apagar_ID_Text.get("1.0", "end-1c")  
    response = requests.delete(url+"ip/address/"+ID_Text, verify=False)  
    response_data = response.json()  
    print(response.status_code)
```

Figura 8.9 : método delete para editar endereços IP



## 9. Listar/ criar/editar/apagar servidores de DHCP

Para manipular todas os servidores de DHCP foi criada uma página á parte que irá servir para Listar/criar/editar/apagar os servidores de DHCP com a ajuda da SDN como mostra a figura



Figura 9.1: servidores de DHCP main page

### 9.1.Listar servidores de DHCP

Nesta página podemos listar os servidores de DHCP ao pressionarmos no botão dedicado o que irá executar um método get para as listar na nossa SDN

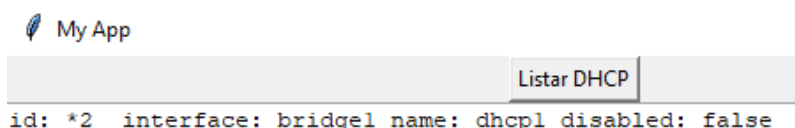


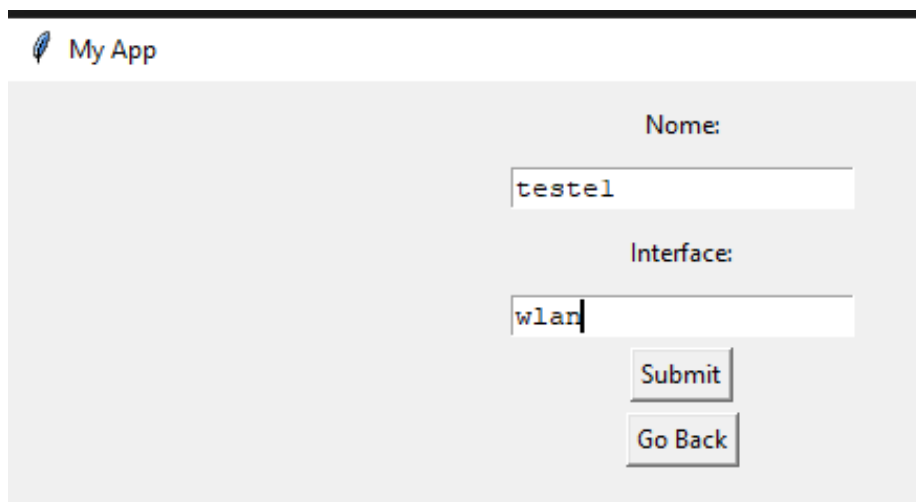
Figura 9.2 : Listar servidores de DHCP

```
def listar_PageDHCP():
    global url
    response = requests.get(url+"ip/dhcp-server",verify=False)
    response_data = response.json()
    devolver = ""
    for i in range(len(response_data)):
        devolver = devolver + ("id: "+response_data[i]['id'] + " interface: "+ response_data[i]['interface']+" name: "
        +response_data[i]['name']+" disabled: "+response_data[i]['disabled']+"\n")
    Response_PageDHCP_Listar.delete('1.0', 'end')
    Response_PageDHCP_Listar.insert('end', devolver)
```

Figura 9.3 : : método get para servidores de DHCP

## 9.2. Criar servidores de DHCP

Na página dedicada para criar servidores de DHCP temos uma Label para inserirmos o nome de seguida vem a Interface, por fim ao pressionarmos o botão submit é enviado um método put que nos faz obter um resultado de sucesso (200) no CMD



The screenshot shows a web interface with a header 'My App'. Below it, there are two labels: 'Nome:' and 'Interface:'. The 'Nome:' label is followed by a text input field containing 'testel'. The 'Interface:' label is followed by a text input field containing 'wlan'. Below these fields are two buttons: 'Submit' and 'Go Back'.

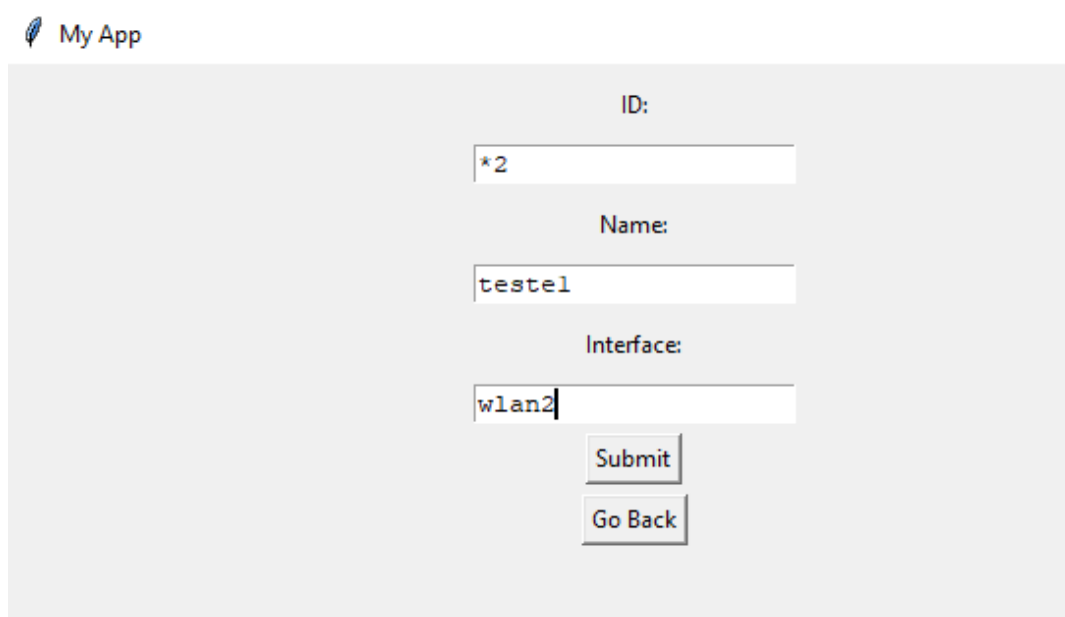
Figura 9.4 : Criar servidores de DHCP

```
def submit_PageDHCP_Criar():  
    global url  
    Nome_Text = PageDHCP_Criar_Nome_Text.get("1.0", "end-1c")  
    Interface_Text = PageDHCP_Criar_Interface_Text.get("1.0", "end-1c")  
    payload = {"name":Nome_Text,"interface":Interface_Text}  
    json_payload = json.dumps(payload)  
    headers = {"Content-Type": "application/json"}  
    response = requests.put(url+"ip/dhcp-server",headers=headers,data=json_payload,verify=False)  
    response_data = response.json()  
    print(response.status_code)
```

Figura 9.5 : método put para servidores de DHCP

### 9.3. Editar servidores de DHCP

Na página dedicada para editar os servidores de DHCP temos a primeira Label, onde podemos inserir o ID que queremos mudar, a segunda serve para mudarmos o nome e por fim a próxima serve para mudara a Interface, ao clicarmos no submit isso irá fazer um patch dos campos que foram alterados.



My App

ID:

Name:

Interface:

Figura 9.7 : editar servidores de DHCP

```
def submit_PageDHCP_Editar():
    global url
    response = requests.get(url+"ip/dhcp-server",verify=False)
    response_data = response.json()
    Id_Text = PageDHCP_Editar_ID_Text.get("1.0","end-1c")
    Name_Text = PageDHCP_Editar_Name_Text.get("1.0","end-1c")
    Interface_Text = PageDHCP_Editar_Interface_Text.get("1.0","end-1c")

    for i in range(len(response_data)):
        if response_data[i]['.id'] == Id_Text:
            print(Id_Text)
            if Name_Text == "":
                Name_Text = response_data[i]['name']
            print(Name_Text)
            if Interface_Text == "":
                Interface_Text = response_data[i]['interface']
            print(Interface_Text)

    payload = {"name":Name_Text,"interface":Interface_Text}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"ip/dhcp-server/"+Id_Text,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)
```

Figura 9.6 : método patch para servidores de DHCP

## 9.4. Apagar servidores de DHCP

Na página dedicada para apagar os servidores de DHCP temos uma Label onde podemos inserir o ID dos endereços IP que queremos apagar, ao submetermos isso irá fazer um delete que vai apagar o endereço IP como mostra na figura



Figura 9.8 : Apagar servidores de DHCP

```
def Submit_PageDHCP_Apagar():  
    global url  
    ID_Text = PageDHCP_Apagar_ID_Text.get("1.0", "end-1c")  
    response = requests.delete(url+"ip/dhcp-server/"+ID_Text, verify=False)  
    response_data = response.json()  
    print(response.status_code)
```

Figura 9.9 : método delete para servidores de DHCP

## 10. Ativar/desativar/configurar servidor DNS

Para manipular o servidor DNS foi criada uma página á parte que irá servir para ativar/desativar/configurar o servidor DNS com a ajuda da SDN como mostra a figura.

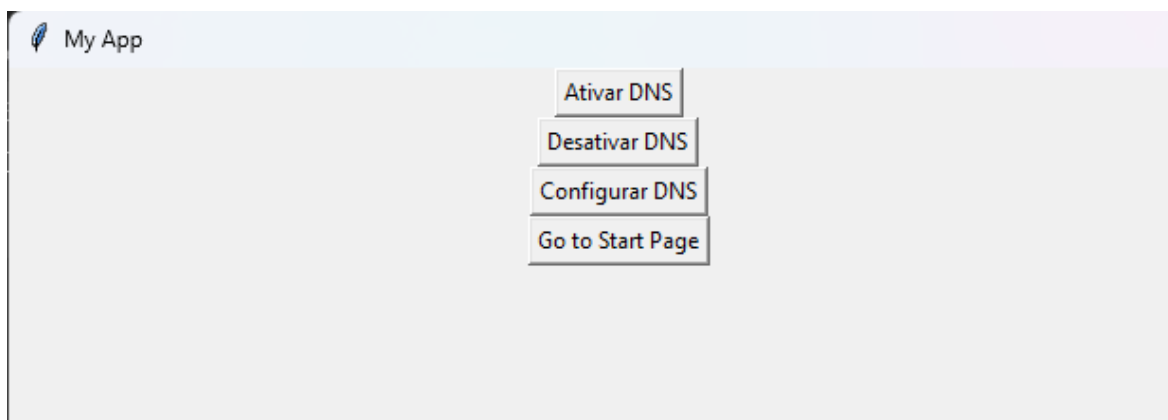


Figura 10.1 : Página DNS

### 10.1.1. Ativar DNS

Para ativar o DNS basta apenas presionar o botão Ativar.

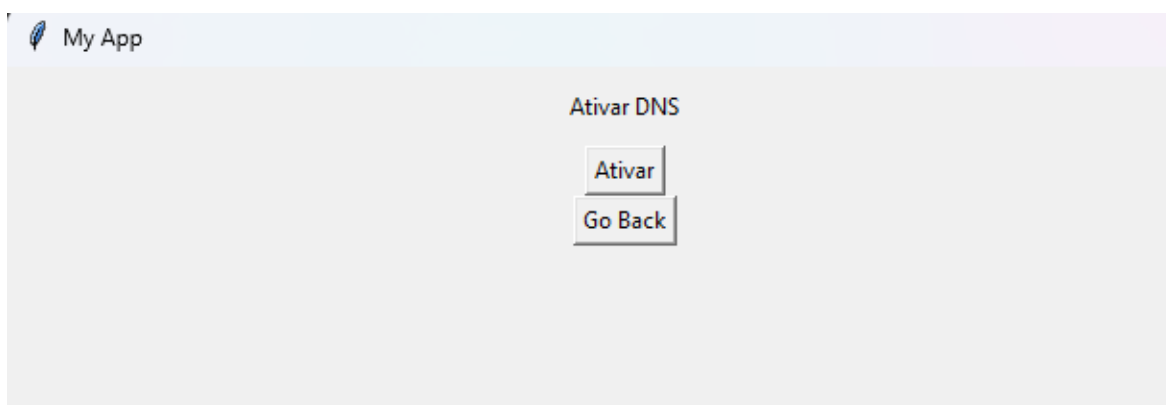


Figura 10.2 : Ativar DNS

### 10.1.2. Desativar DNS

Para ativar o DNS basta apenas presionar o botão Desativar.



Figura 10.3 : Desativar DNS

### 10.1.1. Configurar DNS

Para configurar o DNS basta apenas expicificar os endereços DNS e o DoH Server, de seguida pressionar Submit e é criado o DNS server.

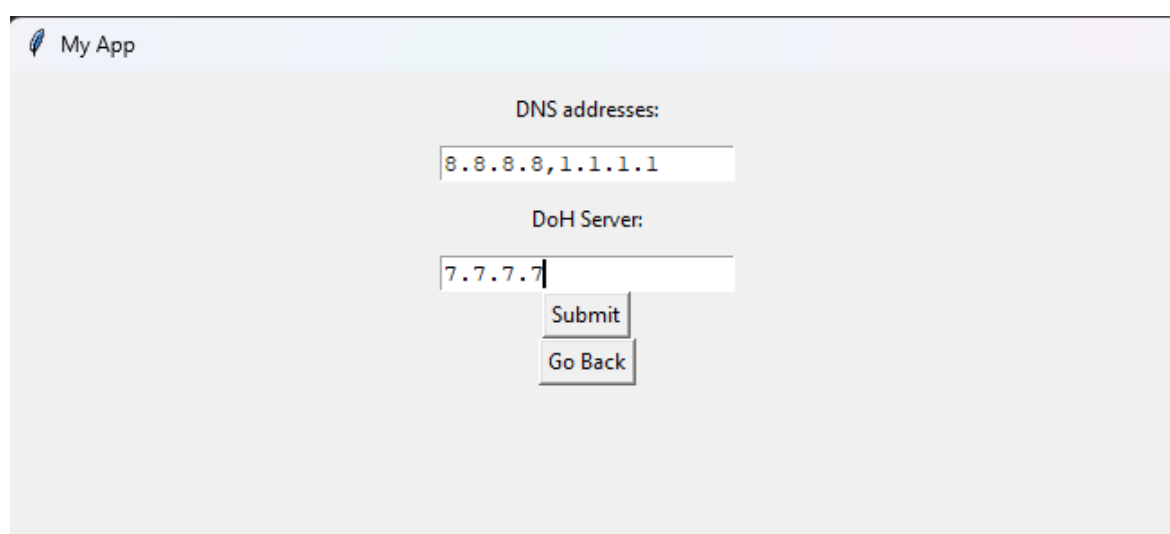


Figura 10.4 : Configurar DNS

## 11. Listar/criar/editar/apagar regras de firewall

Para manipular as firewalls foi criada uma página á parte que irá servir para listar/criar/editar/apagar regras de firewall com a ajuda da SDN como mostra a figura.

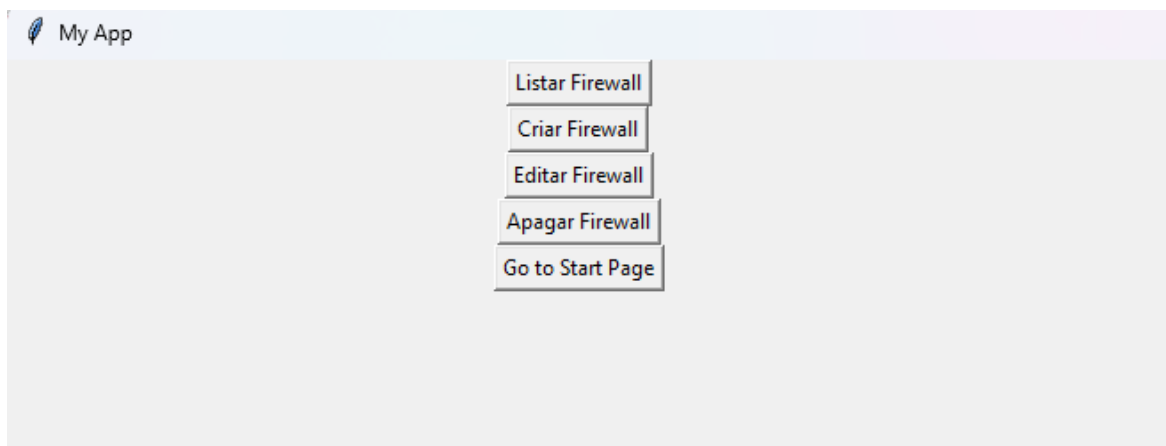


Figura 11.1 : Página de Firewall

### 11.1.1. Listar Firewall

Na página de listar firewall temos um botão que ao ser pressionado lista todas a firewalls

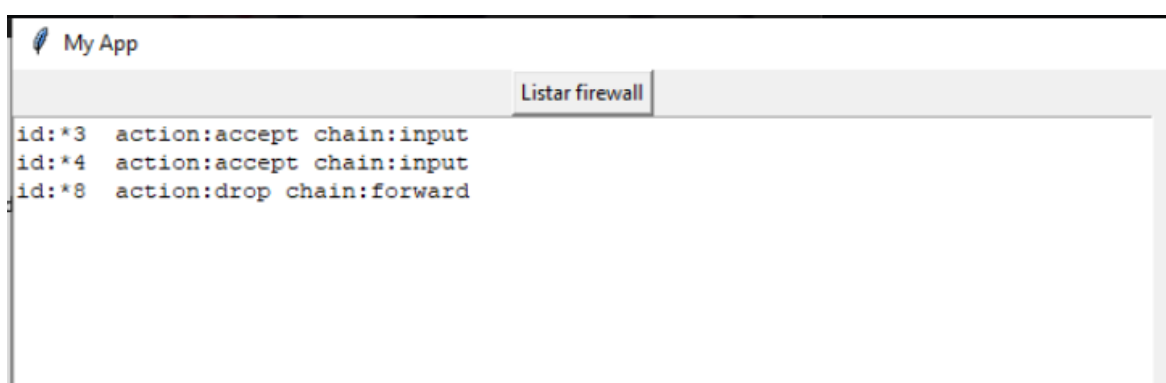


Figura 11.2 : Listar firewall

```
def listar_PageFirewall_Listar():
    global url
    response = requests.get(url+"ip/firewall/filter",verify=False)
    response_data = response.json()
    devolver = ""
    for i in range(len(response_data)):
        devolver = devolver + ("id:"+response_data[i]['id'] + " action:" + response_data[i]['action']+" src-address:"+response_data[i]['src-address']+" chain:"+response_data[i]['chain']+"\n")
    Response_PageFirewall_Listar.delete('1.0', 'end')
    Response_PageFirewall_Listar.insert('end', devolver)

def delete_PageFirewall_Listar():
    Response_PageFirewall_Listar.delete("1.0","end-1c")
```

Figura 11.3 : código Listar Firewall

### 11.1.2. Criar Firewall

Na página dedicada para criar a firewall temos várias Labels onde podemos inserir as várias configurações e ao pressionarmos o botão submit irá fazer um post que irá criar a regra de firewall.

The screenshot shows a web application interface titled "My App". It contains a form for creating a firewall rule with the following fields and values:

- action:** accept
- chain:** input
- Disabled:** false
- src-address:** 10.10.10.10
- dst-address:** 20.20.20.20

At the bottom of the form, there are two buttons: "Submit" and "Go Back".

Figura 11.4 : Criar Regra de Firewall

```
def submit_PageFirewall_Criar():
    global url

    Action_Text = PageFirewall_Criar_Action_Text.get("1.0", "end-1c")
    Chain_Text = PageFirewall_Criar_Chain_Text.get("1.0", "end-1c")
    Disabled_Text = PageFirewall_Criar_Disabled_Text.get("1.0", "end-1c")
    Src_Address_Text = PageFirewall_Criar_Src_Address_Text.get("1.0", "end-1c")
    Dst_Address_Text = PageFirewall_Criar_Dst_Address_Text.get("1.0", "end-1c")

    payload = {"action": Action_Text, "chain": Chain_Text, "disabled": Disabled_Text, "src-address": Src_Address_Text, "dst-address": Dst_Address_Text}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.put(url+"ip/firewall/filter", headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response.status_code)

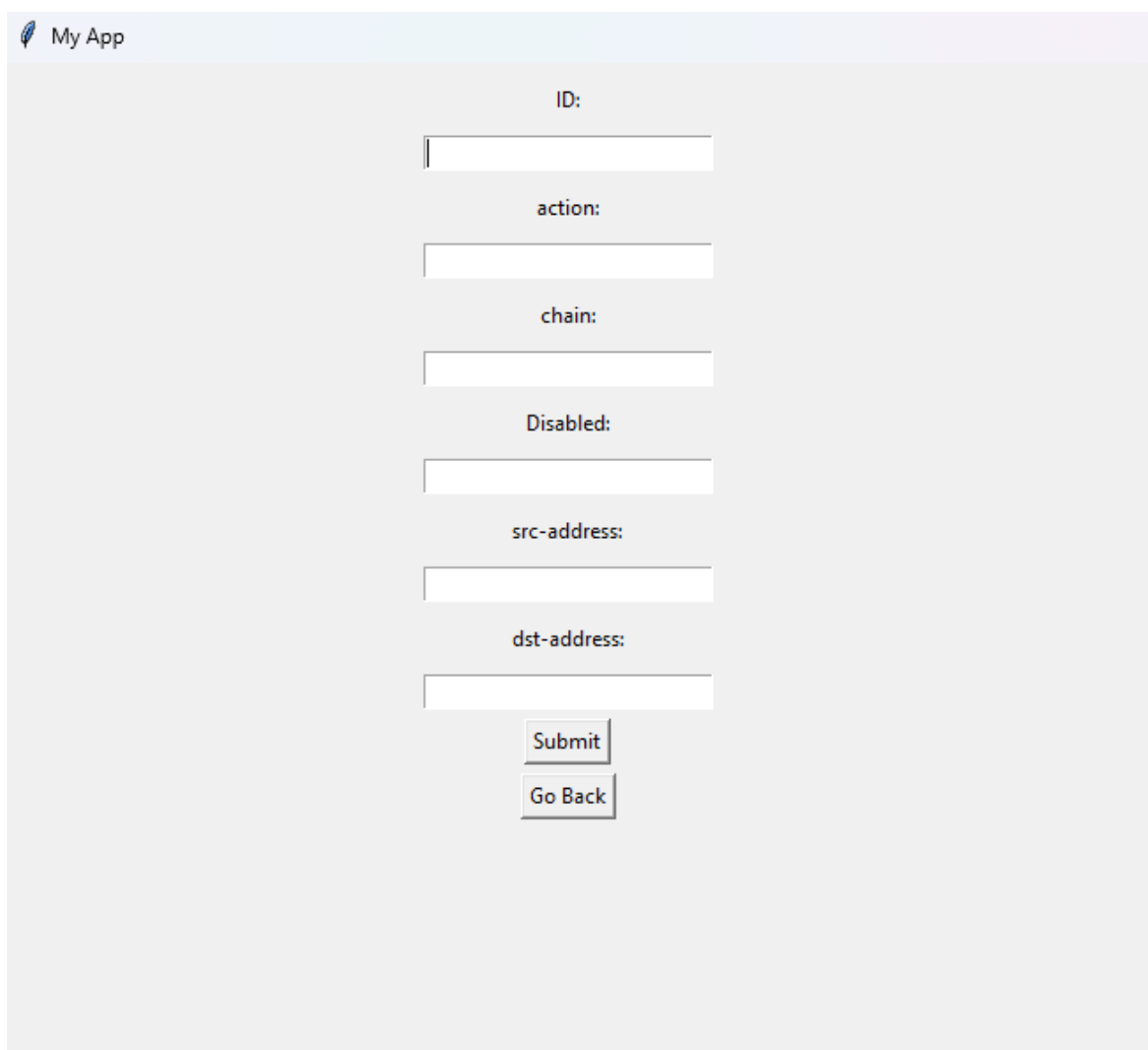
def delete_PageFirewall_Criar():
    PageFirewall_Criar_Action_Text.delete("1.0", "end-1c")
    PageFirewall_Criar_Chain_Text.delete("1.0", "end-1c")
    PageFirewall_Criar_Disabled_Text.delete("1.0", "end-1c")
    PageFirewall_Criar_Src_Address_Text.delete("1.0", "end-1c")
    PageFirewall_Criar_Dst_Address_Text.delete("1.0", "end-1c")
```



Figura 11.5 : Código criar regra Firewall

### 11.1.3. Editar Firewall

Na página dedicada para editar a firewall temos a Label ID onde é inserido o ID da firewall que se quer editar, e as outras Labels onde se pode inserir as várias configurações que se pretende alterar efetuando patch quando pressionamos o botão submit.



My App

ID:

action:

chain:

Disabled:

src-address:

dst-address:

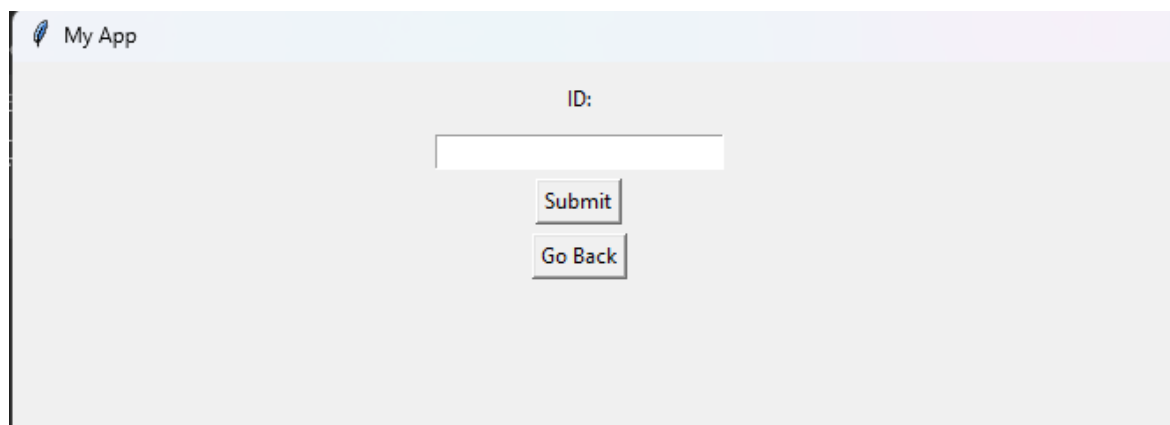
Submit

Go Back

Figura 11.6 : Editar regra firewall

#### 11.1.4. Editar Firewall

Na página dedicada para apagar firewall temos a Label ID, onde é inserido o ID da firewall que se quer apagar, pressionando o botão submit para apagar a regra com o ID inserido.



The screenshot shows a web application interface with a light blue header bar containing a feather icon and the text "My App". The main content area is light gray and contains a form for deleting a firewall rule. The form consists of a label "ID:" followed by a white text input field. Below the input field are two buttons: "Submit" and "Go Back", both with a gray border and a light gray background.

**Figura 11.7 : Apagar regra firewall**

## 12. Ativar/desativar/configurar protocolos de encaminhamento

Para manipular os protocolos de encaminhamento foi criada uma página á parte que irá servir para ativar/desativar/configurar os protocolos de encaminhamento com a ajuda da SDN como mostra a figura.

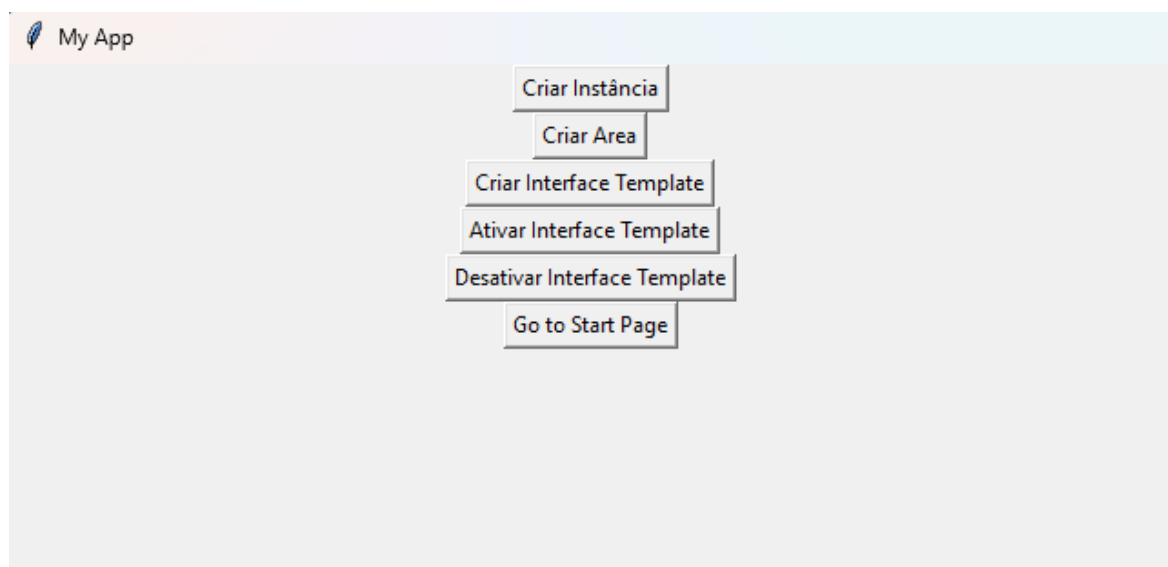


Figura 12.1 : Página de protocolos de encaminhamento

### 12.1.1. Criar protocolo de encaminhamento

Para criar o protocolo de encaminhamento é necessário ter uma instância e uma área para ser possível criar a interface template, sendo os mesmos representados nas seguintes figuras.

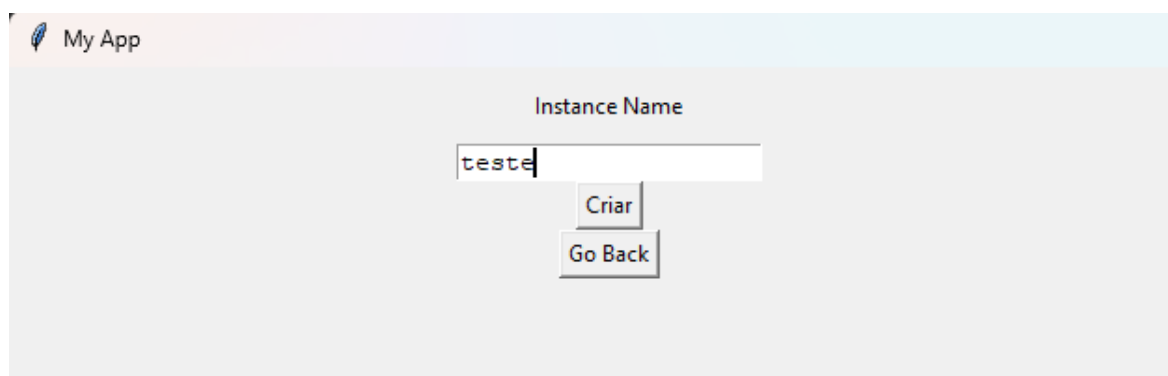


Figura 12.2 : Criar Instância

```
def Submit_PageProtocoloEncaminhamento_Criar_Instanceia():
    """Update the display label with the entered text"""
    global url
    Instancia_Name = PageProtocoloEncaminhamento_Criar_Instanceia_Name.get("1.0", "end-1c")

    payload = {"name": Instancia_Name}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.post(url+"routing/ospf/instance/add", headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response_data)
    print(response.status_code)

def delete_PageProtocoloEncaminhamento_Criar_Instanceia():
    PageProtocoloEncaminhamento_Criar_Instanceia_Name.delete("1.0", "end-1c")
```

Figura 12.3 : código para criar instância

My App

Area Name  
testel

Instance Name  
teste

Area ID  
1.1.1.1

Criar

Go Back

Figura 12.4 : Criar área

```
def Submit_PageProtocoloEncaminhamento_Criar_Area():
    """Update the display label with the entered text"""
    global url
    Area_Name = PageProtocoloEncaminhamento_Criar_Area_Name.get("1.0", "end-1c")
    Instancia_Name = PageProtocoloEncaminhamento_Criar_Area_Instanceia_Name.get("1.0", "end-1c")
    Area_ID = PageProtocoloEncaminhamento_Criar_Area_ID.get("1.0", "end-1c")

    payload = {"name": Area_Name,
              "area-id": Area_ID,
              "instance": Instancia_Name}

    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.post(url+"routing/ospf/area/add", headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response_data)
    print(response.status_code)

def delete_PageProtocoloEncaminhamento_Criar_Area():
    PageProtocoloEncaminhamento_Criar_Area_Name.delete("1.0", "end-1c")
    PageProtocoloEncaminhamento_Criar_Area_Instanceia_Name.delete("1.0", "end-1c")
    PageProtocoloEncaminhamento_Criar_Area_ID.delete("1.0", "end-1c")
```

Figura 12.5 : código para criar área

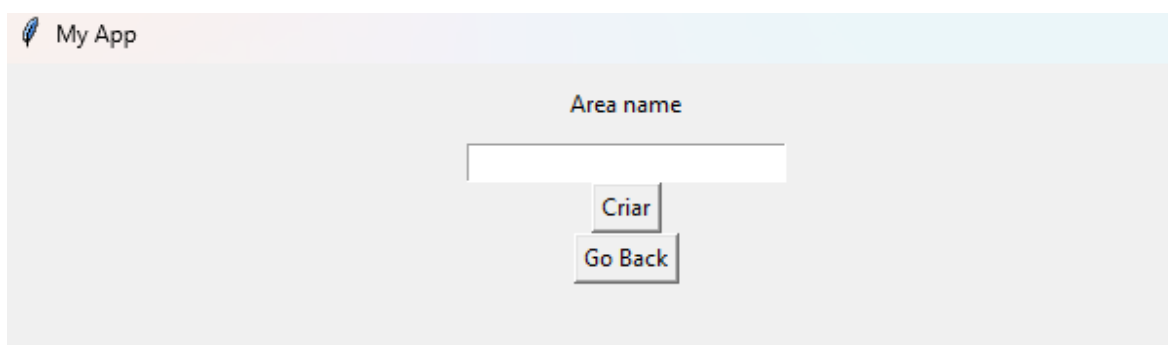


Figura 12.6 : Criar Interface Template

```
def Submit_PageProtocoloEncaminhamento_Interface_Template():
    """Update the display label with the entered text"""
    global url
    Area_Name = PageProtocoloEncaminhamento_Criar_Interface_Template_Area_Name.get("1.0", "end-1c")

    payload = {"area": Area_Name}

    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.post(url+"routing/ospf/interface-template/add", headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response_data)
    print(response.status_code)

def delete_PageProtocoloEncaminhamento_Criar_Interface_Template():
    PageProtocoloEncaminhamento_Criar_Interface_Template_Area_Name.delete("1.0", "end-1c")
```

Figura 12.7 : código para criar Interface template

### 12.1.2. Ativar protocolo de encaminhamento

Na página dedicada para ativar o protocolo de encaminhamento temos uma Label onde podemos inserir o nome e ao pressionarmos o botão submit irá fazer um patch que irá ativar o protocolo de encaminhamento.

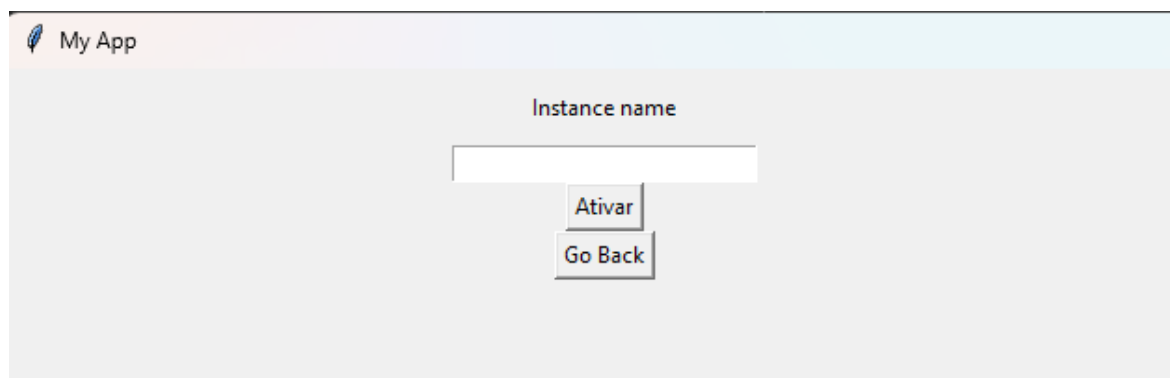


Figura 12.8 : Ativar Instância

```
def Submit_PageProtocoloEncaminhamento_Ativar():
    """Update the display label with the entered text"""
    global url
    response = requests.get(url+"routing/ospf/instance",verify=False)
    response_data = response.json()
    #print(response_data)

    Name_Text_To_Patch = PageProtocoloEncaminhamento_Ativar_Instanceia.get("1.0","end-1c")

    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name_Text_To_Patch:
            ID_a_mudar = response_data[i]['.id']
            print(ID_a_mudar)

    payload = {"disabled":"false"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"routing/ospf/instance/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)

def delete_PageProtocoloEncaminhamento_Ativar_Instanceia():
    PageProtocoloEncaminhamento_Ativar_Instanceia.delete("1.0","end-1c")
```

Figura 12.9 : código para ativar instância

### 12.1.3. Desativar protocolo de encaminhamento

Na página dedicada para ativar o protocolo de encaminhamento temos uma Label onde podemos inserir o nome e ao pressionarmos o botão submit irá fazer um patch que irá desativar o protocolo de encaminhamento.

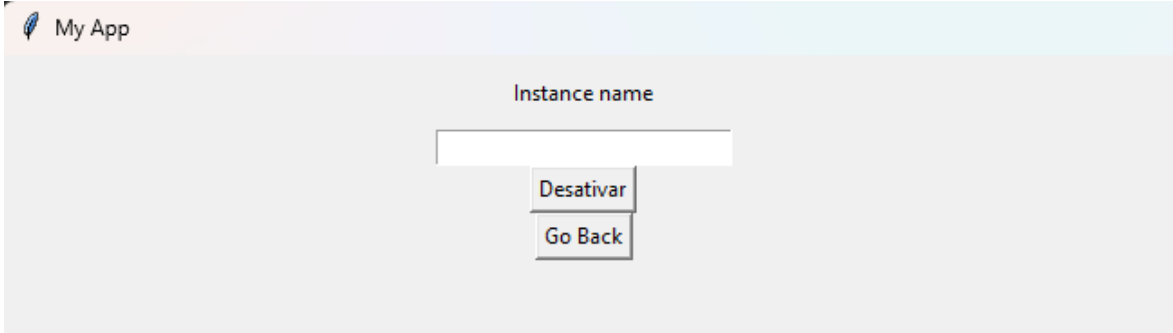


Figura 12.10 : desativar Instância

```
def Submit_PageProtocoloEncaminhamento_Desativar():
    """Update the display label with the entered text"""
    global url
    response = requests.get(url+"routing/ospf/instance",verify=False)
    response_data = response.json()
    #print(response_data)

    Name_Text_To_Patch = PageProtocoloEncaminhamento_Desativar_Instanceia.get("1.0","end-1c")

    ID_a_mudar = ""
    for i in range(len(response_data)):
        if response_data[i]['name'] == Name_Text_To_Patch:
            ID_a_mudar = response_data[i]['.id']
            print(ID_a_mudar)

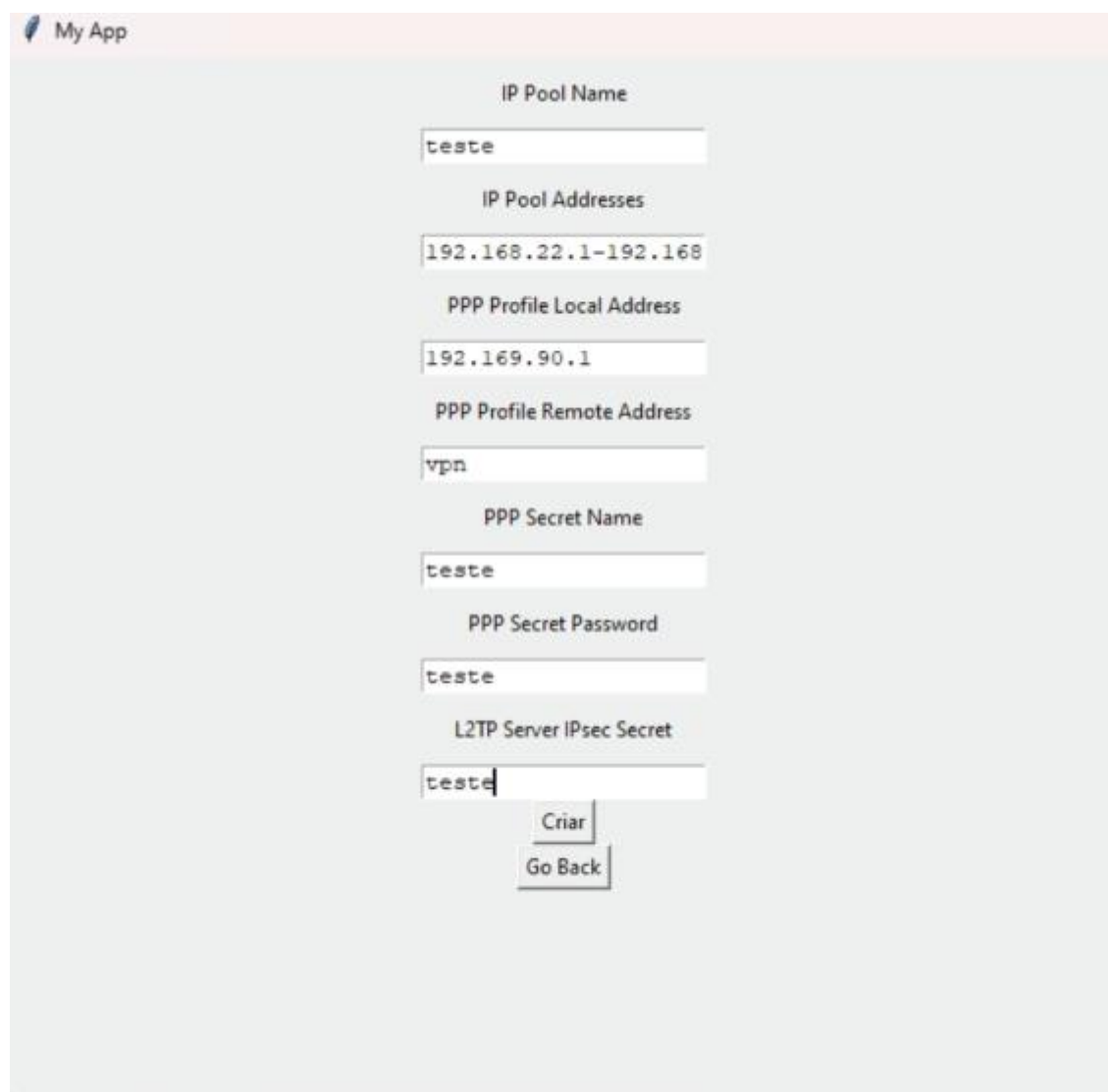
    payload = {"disabled":"true"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.patch(url+"routing/ospf/instance/"+ID_a_mudar,headers=headers,data=json_payload,verify=False)
    response_data = response.json()
    print(response.status_code)

def delete_PageProtocoloEncaminhamento_Desativar_Instanceia():
    PageProtocoloEncaminhamento_Desativar_Instanceia.delete("1.0","end-1c")
```

Figura 12.11 : código para desativar Instância

## 13. VPN

Para a VPN foi criada uma página á parte que irá servir para criar a VPN, para tal é necessário criar um IP Pool, um PPP Profile, PPP Secret e ativar o L2TP server como mostra a figura.



My App

IP Pool Name  
teste

IP Pool Addresses  
192.168.22.1-192.168

PPP Profile Local Address  
192.169.90.1

PPP Profile Remote Address  
vpn

PPP Secret Name  
teste

PPP Secret Password  
teste

L2TP Server IPsec Secret  
teste

Criar  
Go Back

Figura 13.1 : criar VPN



```

def Submit_PageVPN():
    """Update the display label with the entered text"""

    #print(response_profile_data)

    #Make Pool
    Pool_Name = PageVPN_Pool_Name.get("1.0", "end-1c")
    Pool_Addresses = PageVPN_Pool_Addresses.get("1.0", "end-1c")

    payload = {"name": Pool_Name,
              "ranges": Pool_Addresses}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response = requests.post(url="ip/pool/add", headers=headers, data=json_payload, verify=False)
    response_data = response.json()
    print(response.status_code)

    #Patch PPP Profile "default-encryption"

    PPP_Profile_Local = PageVPN_PPP_Profile_Local_Address.get("1.0", "end-1c")
    PPP_Profile_Remote = PageVPN_PPP_Profile_Remote_Address.get("1.0", "end-1c")

    payload = {"local-address": PPP_Profile_Local,
              "remote-address": PPP_Profile_Remote}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response_profile = requests.patch(url="ppp/profile/FFFFFFFE", headers=headers, data=json_payload, verify=False)
    response_data = response_profile.json()
    print(response_profile.status_code)

    #Make Secret
    PPP_Secret_Name = PageVPN_PPP_Secret_Name.get("1.0", "end-1c")
    PPP_Secret_Password = PageVPN_PPP_Secret_Password.get("1.0", "end-1c")

    payload = {"name": PPP_Secret_Name,
              "password": PPP_Secret_Password,
              "profile": "default-encryption",
              "service": "l2tp"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response_profile = requests.post(url="ppp/secret/add", headers=headers, data=json_payload, verify=False)
    response_data = response_profile.json()
    print(response_profile.status_code)

    #Enable IPsec
    IPsec_Secret = PageVPN_L2TP_Server_IPsec_Secret.get("1.0", "end-1c")

    payload = {"enabled": "true",
              "ipsec-secret": IPsec_Secret,
              "authentication": "chap,mschap1,mschap2",
              "use-ipsec": "required",
              "default-profile": "default-encryption"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response_profile = requests.post(url="interface/l2tp-server/server/set", headers=headers, data=json_payload, verify=False)
    response_profile_data = response_profile.json()
    print(response_profile.status_code)

    payload = {"chain": "input",
              "protocol": "ipsec-esp",
              "action": "accept"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response_profile = requests.post(url="ip/firewall/filter/add", headers=headers, data=json_payload, verify=False)
    response_profile_data = response_profile.json()
    print(response_profile.status_code)

    payload = {"chain": "input",
              "protocol": "udp",
              "port": "1701,500,4500",
              "action": "accept"}
    json_payload = json.dumps(payload)
    headers = {"Content-Type": "application/json"}
    response_profile = requests.post(url="ip/firewall/filter/add", headers=headers, data=json_payload, verify=False)
    response_profile_data = response_profile.json()
    print(response_profile.status_code)

def delete_PageVPN():
    PageVPN_L2TP_Server_IPsec_Secret.delete("1.0", "end-1c")
    PageVPN_Pool_Addresses.delete("1.0", "end-1c")
    PageVPN_Pool_Name.delete("1.0", "end-1c")
    PageVPN_PPP_Profile_Local_Address.delete("1.0", "end-1c")
    PageVPN_PPP_Profile_Remote_Address.delete("1.0", "end-1c")
    PageVPN_PPP_Secret_Name.delete("1.0", "end-1c")
    PageVPN_PPP_Secret_Password.delete("1.0", "end-1c")

```

Figura 13.2 : código para criar VPN

## 14. Análise crítica e proposta de melhorias

No âmbito do projeto foi feito um trabalho funcional, e que permite as mínimas configurações básicas do router MikroTik.

Como qualquer solução existe sempre uma margem de melhoria não sendo diferente com o nosso trabalho desenvolvido. Dentro das possíveis melhorias as que se destacam mais são a utilização de uma interface mais bonita e fácil de usar, a criação de testes unitários e a opção de configurar mais campos em cada configuração.

## 15. Conclusão

Com base no que foi apresentado, pode-se concluir que o trabalho foi concluído com sucesso, tendo sido criada uma SDN funcional em Python que permite interagir com qualquer router MikroTik.

O trabalho contribuiu para novos aprendizados e a consolidação dos mesmos tais como a consolidação de conhecimentos em criação de API Rest, RouterOS, e a criação de uma SDN.