Escuela Superio de Cómputo
Instituto Politécnico Nacional
Administración de Servicios en Red
Practica 2
Curso impartido por: Ricardo Martinez Rosales

Adrian González Pardo

2 de abril de 2021

1. Descripción y Desarrollo

Para desarrollar esta practica, previamente se estudio y se leyo acerca de los siguientes temas:

- Webservice con Flask
- Sqlite en Python
- Enrutamiento estatico
- Modelo de desarrollo REST
- Uso de Interfaces Virtuales de Red para GNS3
- Creación, Modificación y Eliminación de usuarios en CISCO
- Configuración de protocolos para conexión remota (SSH y Telnet)

Con esto se desarrollo una aplicación REST que fuese capaz de procesar la información de tal modo que pudiese administrar los usuarios que se encuentran en la misma, por ello analizando las instrucciones que procesa HTTP se encontro que existen más peticiones como POST, PUT, DELETE en las cuales nos pudimos auxiliar para el desarrollo de la app.
Finalmente con la gracia y el deseo de realizar una reutilización del código se obto por crear algunos modulos en archivos de Python, para poder reutilizarlos más adelante.

2. Modulos

```
#!/usr/bin/env python3
"""
    Author: Adrian González Pardo
    Email: gozapaadr@gmail.com
    A.k.a: d3vcr4ck / DevCrack
    Fecha de modificación: 01/04/2021
    GitHub: AdrianPardo99
    Licencia Creative Commons CC BY-SA
"""

import telnetlib

def conexion(host,user,password,cmd_list):
    tn=telnetlib.Telnet(host)
    tn.read_until(b"Username: ")
    tn.write(user.encode('ascii')+b"\n")
```

```
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")
    for i in cmd_list:
        tn.write(i.encode("ascii")+b"\n")
    print(tn.read_all().decode("ascii"))
    tn.close()
#host="192.168.0.1"
#user="cisco"
#password="cisco"
#cmd=["sh ip route","sh ip int br","exit"]
#tn=conexion(host,user,password,cmd)
```

```
#!/usr/bin/env python3
import sqlite3
"""
    Author: Adrian González Pardo
    Email: gozapaadr@gmail.com
    A.k.a: d3vcr4ck / DevCrack
    Fecha de modificación: 09/08/2020
    GitHub: AdrianPardo99
    Licencia Creative Commons CC BY-SA
"""
def create_database(name):
    """
        @name       ->  Es el nombre bajo el cual se resguardara el archivo de sqlite3
    """
    return sqlite3.connect(name)

def close_database(conexion):
    """
        @conexion   ->  Es la conexion a la base de datos existente gracias al create_database
    """
    conexion.close()


def create_table_with_params(conexion,table,params):
    """
        @conexion   ->  Es la conexion a la base de datos existente gracias al create_database
        @table      ->  Es el nombre de la tabla que se almacenara de acuerdo a la variable @name
        @params     ->  Son los campos/parametros que componen a la tabla
    """
    cursorTable=conexion.cursor()
    print(f'Query execute:\nCREATE TABLE IF NOT EXISTS {table} ({params})')
    cursorTable.execute(f'CREATE TABLE IF NOT EXISTS {table} ({params})')
    conexion.commit()

def insert_into_table(conexion,table,params,data):
    """
        @conexion   ->  Es la conexion a la base de datos existente gracias al create_database
        @table      ->  Es el nombre de la tabla que se accedera
        @params     ->  Son los campos/parametros que componen a la tabla
        @tuple      ->  Es la tupla de datos que evitaran un hardcoding de los mismo y añadiran
    una capa de seguridad
    """
    cursorTable=conexion.cursor()
    print(f'Query execute: \n INSERT INTO {table} VALUES ({params})')
    if type(data) is list:
        cursorTable.executemany(f'INSERT INTO {table} VALUES ({params})',data)
    elif type(data) is tuple:
        cursorTable.execute(f'INSERT INTO {table} VALUES ({params})',data)
    conexion.commit()

def select_query_all(conexion,table):
    """
        @conexion   ->  Es la conexion a la base de datos existente gracias al create_database
        @table      ->  Es el nombre de la tabla que se accedera con el query select*from
    """
    cursorTable=conexion.cursor()
    cursorTable.execute(f'SELECT*FROM {table}')
    return cursorTable.fetchall()

```

```
def select_query_all_with_where(conexion,table,whereTab,dataWhere):
    """
        @conexion    ->  Es la conexion a la base de datos existente gracias al create_database
        @table       ->  Es el nombre de la tabla que se accedera con el query select*from
        @whereTab    ->  Datos que van en la sentencia where sin hardcoding,
        puede igual ponerse operaciones de orden
        @dataWhere   ->  Tupla de datos que van en el where
    """
    cursorTable=conexion.cursor()
    cursorTable.execute(f'SELECT*FROM {table} WHERE {whereTab}',dataWhere)
    return cursorTable.fetchall()

def select_query_2_all(conexion,selectArg,table):
    """
        @conexion    ->  Es la conexion a la base de datos existente gracias al create_database
        @selectArg   ->  Puede seleccionarse que elementos seran vistos en la consulta
        @table       ->  Es el nombre de la tabla que se accedera con el query select {} from
    """
    cursorTable=conexion.cursor()
    cursorTable.execute(f'SELECT {selectArg} FROM {table}')
    return cursorTable.fetchall()

def select_query_2_with_where(conexion,selectArg,table,whereTab,dataWhere):
    """
        @conexion    ->  Es la conexion a la base de datos existente gracias al create_database
        @selectArg   ->  Puede seleccionarse que elementos seran vistos en la consulta
        @table       ->  Es el nombre de la tabla que se accedera con el query select {} from
        @whereTab    ->  Datos que van en la sentencia where sin hardcoding,
        puede igual ponerse operaciones de orden
        @dataWhere   ->  Tupla de datos que van en el where
    """
    cursorTable=conexion.cursor()
    cursorTable.execute(f'SELECT {selectArg} FROM {table} WHERE {whereTab}',dataWhere)
    return cursorTable.fetchall()

def delete_from(conexion,table,whereTab,dataWhere):
    """
        @conexion    ->  Es la conexion a la base de datos existente gracias al create_database
        @table       ->  Es el nombre de la tabla que se accedera
        @whereTab    ->  Datos que van en la sentencia where sin hardcoding
        @dataWhere   ->  Tupla de datos que van en el where
    """
    cursorTable=conexion.cursor()
    cursorTable.execute(f'DELETE FROM {table} where {whereTab}',dataWhere)
    conexion.commit()


def update_data(conexion,table,dataSet,whereTab,data):
    """
        @conexion    ->  Es la conexion a la base de datos existente gracias al create_database
        @table       ->  Es el nombre de la tabla que se accedera
        @dataSet     ->  Son los datos que van a ser modificados por el
        @whereTab    ->  Datos que van en la sentencia where sin hardcoding
    """
    cursorTable=conexion.cursor()
    cursorTable.execute(f'UPDATE {table} SET {dataSet} where {whereTab}',data)
    conexion.commit()
```

3. App

```
#!/usr/bin/env python3
"""
    Author: Adrian González Pardo
    Email: gozapaadr@gmail.com
    A.k.a: d3vcr4ck / DevCrack
    Fecha de modificación: 01/04/2021
    GitHub: AdrianPardo99
    Licencia Creative Commons CC BY-SA
"""
from flask import Flask, redirect, request
```

```
from conecta import *
from databaseController import *
from markupsafe import escape
import json

usur="admin"
password="admin01"
con=create_database("routers.db")
create_table_with_params(con,"rest","ip TEXT, usuario TEXT, pass TEXT")
close_database(con)
app = Flask(__name__)

@app.route("/router",methods=["POST","PUT","DELETE","GET"])
@app.route("/router/all")
def router():
    con=create_database("routers.db")
    cmd=[]
    if request.method=="POST":
        ip=request.form["ip"]
        user=request.form["user"]
        pwd=request.form["pwd"]
        print("Alta")
        if len(select_query_all_with_where(con,"rest","ip=? and usuario=?",(ip,user,)))>0:
            close_database(con)
            return {"Error":"Usuario en existencia"}
        insert_into_table(con,"rest","?,?,?",(ip,user,pwd))
        cmd=["conf t",f'username {user} privilege 15 password {pwd}',"end","exit"]
        conexion(ip,usur,password,cmd)
        close_database(con)
        return {"Operacion":f'Alta de usuario {user}'}
    elif request.method=="PUT":
        ip=request.form["ip"]
        user=request.form["user"]
        pwd=request.form["pwd"]
        info=select_query_all_with_where(con,"rest","ip=? and usuario=?",(ip,user,))
        if len(info)==0:
            return {"Error":"Usuario no existe para modificar"}
        update_data(con,"rest","pass=?","ip=? and usuario=?",(pwd,ip,user))
        cmd=["conf t",f'username {user} privilege 15 password {pwd}',"end","exit"]
        conexion(ip,usur,password,cmd)
        close_database(con)
        return {"Operacion":f'Modificacion de usuario {user}'}
    elif request.method=="DELETE":
        ip=request.form["ip"]
        user=request.form["user"]
        pwd=request.form["pwd"]
        info=select_query_all_with_where(con,"rest","ip=? and usuario=?",(ip,user,))
        if len(info)==0:
            return {"Error":"Usuario no existe para eliminar"}
        delete_from(con,"rest","ip=? and usuario=?",(ip,user,))
        cmd=["conf t",f'no username {user} privilege 15 password {pwd}',"end","exit"]
        conexion(ip,usur,password,cmd)
        print("Eliminación")
        close_database(con)
        return {"Operacion":f'Eliminacion de usuario {user}'}
    elif request.method=="GET":
        if "all" in str(request.url_rule):
            print("Encontrado")
            l=select_query_all(con,"rest")
            if len(l)==0:
                return {"Informacion":"No hay datos que mostrar"}
            jj=[]
            for i in l:
                st="{"+f'"ip":"{i[0]}", "usuario":"{i[1]}", "pass":"{i[2]}"'+"}"
                jj.append(json.loads(st))
            close_database(con)
            return {"About":jj}
        close_database(con)
        return {"No info":f'{request.url_rule}'}

if __name__=="__main__":
```

```
    app.run("0.0.0.0",50000,debug=True)
```

Finalmente con este código y debido a que no se desea realizar uso o crear un archivo html para esto, se procede con la realización de scripts en bash para el consumo de la aplicación

4. Scripts

```bash
#!/usr/bin/env bash

if [[ $# -ne 4 ]];then
  echo "Usage script $0 <URL-Server> <IP-Router> <user> <password>"
  exit 1
fi

url=$1
ip=$2
usr=$3
pss=$4

curl ${url} -X POST -d "ip=${ip}&user=${usr}&pwd=${pss}"
echo "Hecho"
```

```bash
#!/usr/bin/env bash

if [[ $# -ne 4 ]];then
  echo "Usage script $0 <URL-Server> <IP-Router> <user> <password>"
  exit 1
fi

url=$1
ip=$2
usr=$3
pss=$4

curl ${url} -X PUT -d "ip=${ip}&user=${usr}&pwd=${pss}"
echo "Hecho"
```

```bash
#!/usr/bin/env bash

if [[ $# -ne 4 ]];then
  echo "Usage script $0 <URL-Server> <IP-Router> <user> <password>"
  exit 1
fi

url=$1
ip=$2
usr=$3
pss=$4

curl ${url} -X DELETE -d "ip=${ip}&user=${usr}&pwd=${pss}"
echo "Hecho"
```

5. Parte de demostración

Para la demostración de como se desarrollo y una pequeña explicación se grabo un vídeo que se puede ver aquí destacando y pidiendo disculpa por la cantidad de sueño que se expresa directamente o indirectamente durante la grabación y presentación.