Santiago de Cali, 1 de noviembre de 2020

Parcial Final Servicios Telemáticos

Estudiantes:

* David Corredor [davidcorredor10/UAO: servicios telematicos (github.com)](https://github.com/davidcorredor10/UAO/)
* Julian A. Nuñez

1. *[1.5 Puntos (Funcionamiento + Sustentación)] SERVICIO + FIREWALL. Instalar el servidor de streaming Streama protegido por Firewall como se muestra en la figura. Todas las solicitudes hacia el servidor Streama deberán ser realizadas al firewall y no directamente al servicio configurado. El firewall debe redirigir las peticiones al servicio. Compruebe el funcionamiento desde el navegador del anfitrión y del SmartPhone.*

**R//**

**Configuración VagrantFile:**

Vagrant.configure("2") do |config|

config.vm.define :firewall1 do |firewall1|

firewall1.vm.box = "bento/centos-7"

firewall1.vm.network :private\_network, ip: "192.168.50.2"

firewall1.vm.network :public\_network, bridge: "Realtek 8822CE Wireless LAN 802.11ac PCI-E NIC", ip: "192.168.10.29"

firewall1.vm.hostname = "firewall1"

end

config.vm.define :servidorSTRM do |servidorSTRM|

servidorSTRM.vm.box = "bento/centos-7"

#servidorSTRM.vm.network :public\_network, bridge: "Realtek 8822CE Wireless LAN 802.11ac PCI-E NIC", ip: "192.168.1.55"

servidorSTRM.vm.network :private\_network, ip: "192.168.50.2"

servidorSTRM.vm.hostname = "servidorSTRM"

end

end

**Procedimiento en máquina servidorSTRM:**

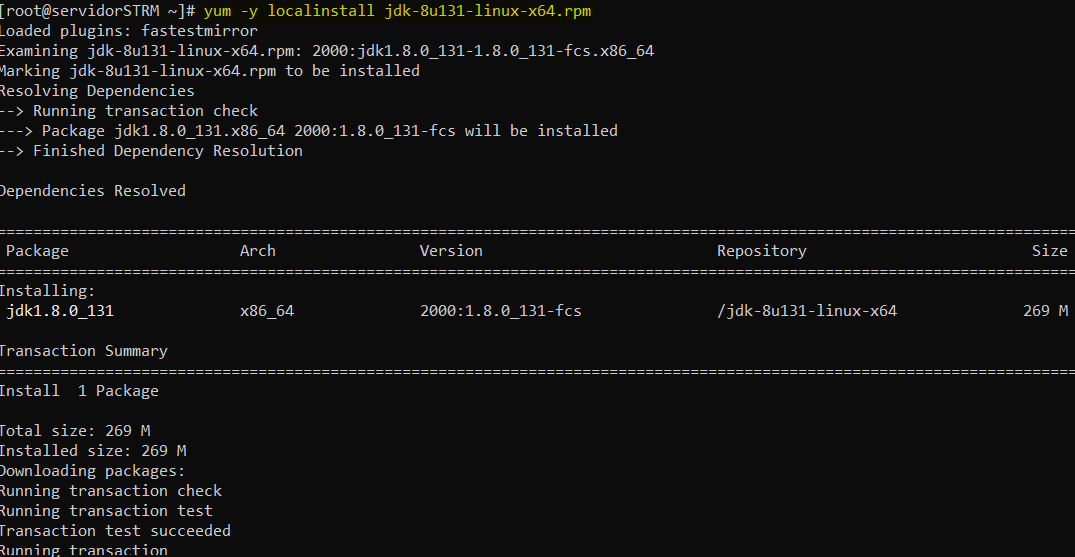
* REALIZAR EL WGET Y DESCARGAR EL JAVA.RPM

*wget --no-cookies --no-check-certificate --header "Cookie:oraclelicense=accept-securebackup-cookie" "*[*http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm*](http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm)*"*

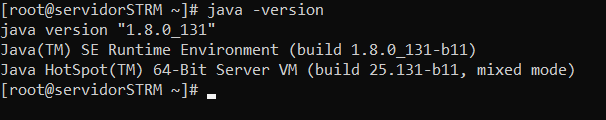
**

* INSTALAR EL JAVA

*yum -y localinstall jdk-8u131-linux-x64.rpm*

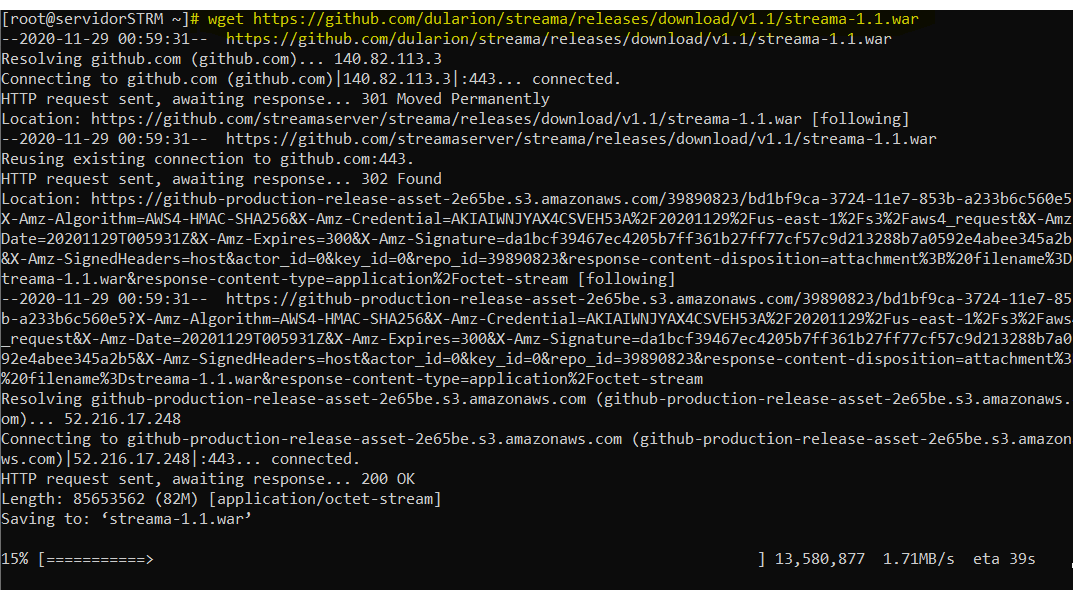


* *java -version*

**

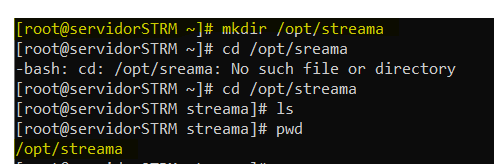
* *Descargar el war streama*

*wget https://github.com/dularion/streama/releases/download/v1.1/streama-1.1.war*

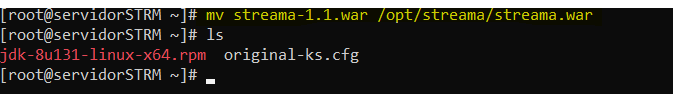
**

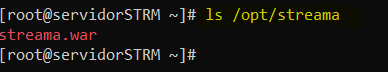
* *Creamos una carpeta en /opt/streama y movemos el archivo descargado*

*mkdir /opt/streama*

**

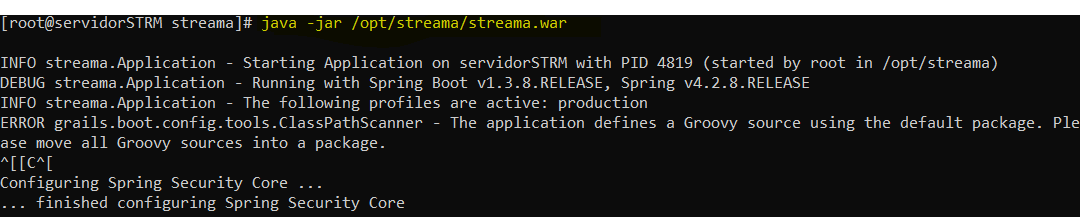
*mv streama-1.1.war /opt/streama/streama.war*

**

**

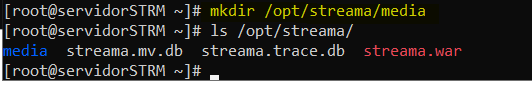
* *Ejecutar el archivo java:*

*java -jar /opt/streama/streama.war*

**

*Control + c para salir*

* *Crear carpeta media mkdir /opt/streama/media*

**

*ls /opt/streama/media*

* *Damos permisos a la carpeta media*

**

*chmod 664 /opt/streama/media*

* *Crear el servicio streama “Para poder reiniciarlo como httpd u otro”*

**

*vim /etc/systemd/system/streama.service*

* *Agregar al archivo streama.service*

*[Unit]*

*Description=Streama Server*

*After=syslog.target*

*After=network.target*

*[Service]*

*User=root*

*Type=simple*

*ExecStart=/bin/java -jar /opt/streama/streama.war*

*Restart=always*

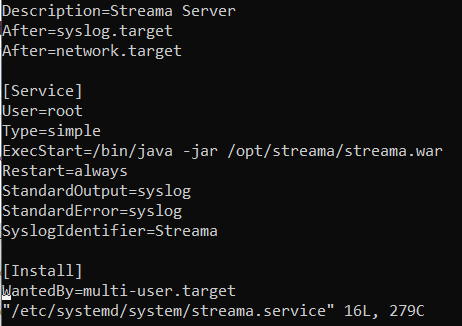
*StandardOutput=syslog*

*StandardError=syslog*

*SyslogIdentifier=Streama*

*[Install]*

*WantedBy=multi-user.target*

**

* *Iniciar el servicio*

**

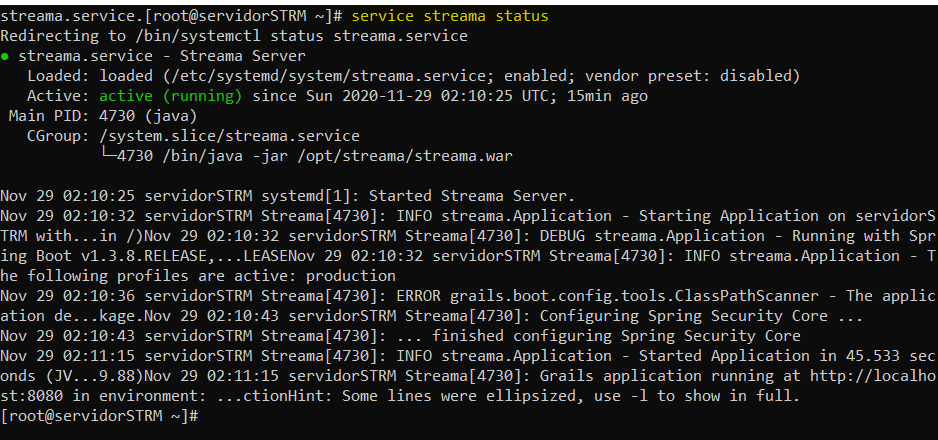
*service streama start*

* *Para que el servicio inicie junto al sistema*

**

*systemctl enable streama*

* *Para ver el estado del servicio*

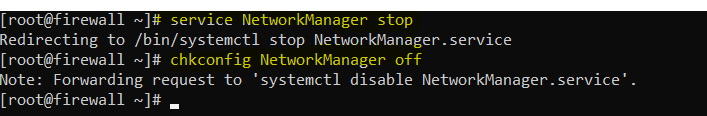
**

**Procedimiento en máquina firewall:**

Para evitar algún tipo de conflicto en la configuración de nuestro Firewall detenemos nuestro “NetworkManager”:

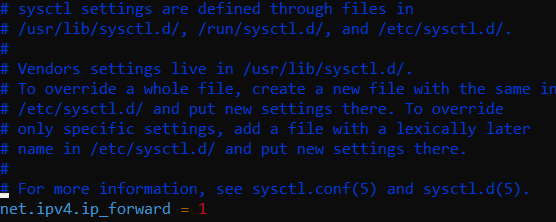
*service NetworkManager stop*

*chkconfig NetworkManager off*

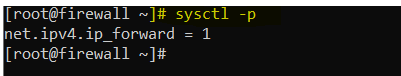
**

* Modificar el sysctl.conf y agregar net.ipv4.ip\_forward = 1

*vim /etc/sysctl.conf*



* ejecutar sysctl -p

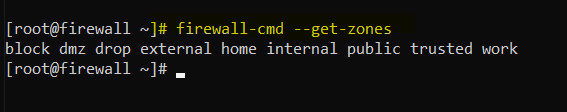


(Y nos debe mostrar la regla que configuramos)

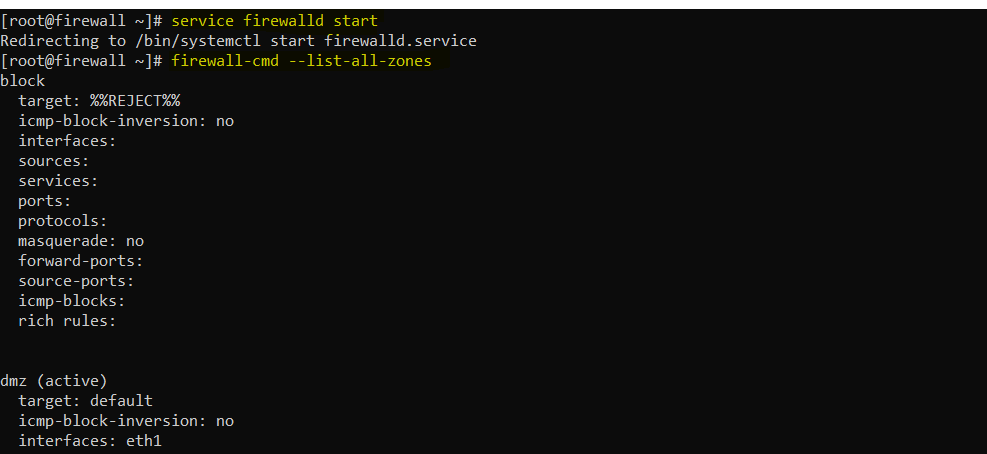
Ahora debemos definir Zonas en firewallNAT:

* Zona internal, para la interfaz que va con la red privada eth2
* Zona dmz, para la interfaz que va con la red pública eth1

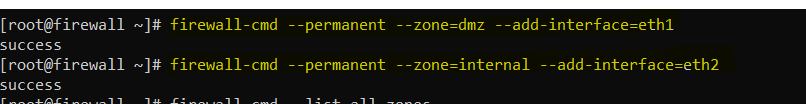
*firewall-cmd --get-zones*



*firewall-cmd --list-all-zones*



* Asignamos interfaces a cada una

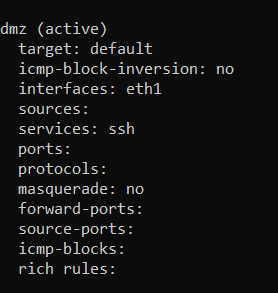


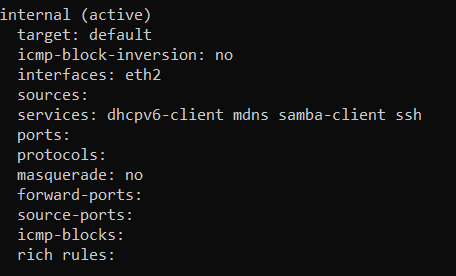
*firewall-cmd --permanent --zone=dmz --add-interface=eth1*

*firewall-cmd --permanent --zone=internal --add-interface=eth2*

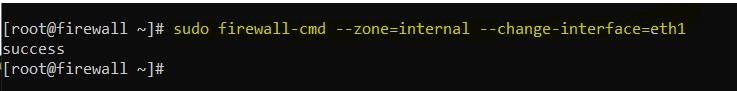
*firewall-cmd --reload*

*firewall-cmd --list-all-zones*

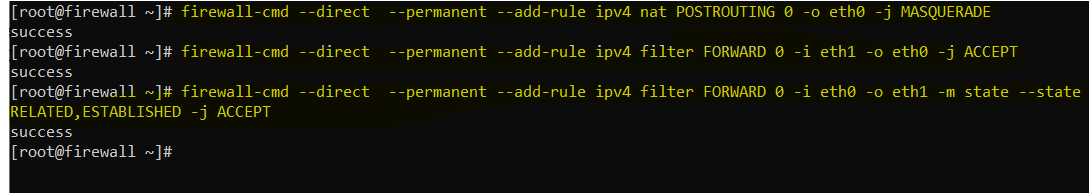




*sudo firewall-cmd --zone=internal --change-interface=eth1*



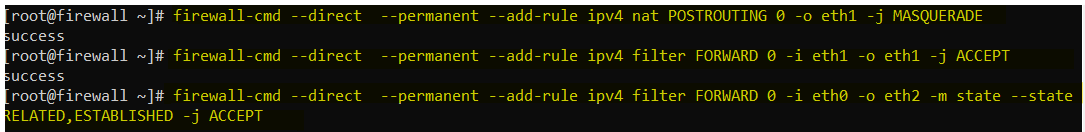
* Generar las siguientes reglas



*firewall-cmd --direct --permanent --add-rule ipv4 nat POSTROUTING 0 -o eth1 -j MASQUERADE*

*firewall-cmd --direct --permanent --add-rule ipv4 filter FORWARD 0 -i eth2 -o eth1 -j ACCEPT*

*firewall-cmd --direct --permanent --add-rule ipv4 filter FORWARD 0 -i eth1 -o eth2 -m state --state RELATED,ESTABLISHED -j ACCEPT*



* Agregar el servicio http a el firewall

firewall-cmd --permanent --zone=dmz --add-service=http



* Agregar el protocolo con el puerto

*sudo firewall-cmd --zone=dmz --add-port=8080/tcp --permanent*

* Agregar redireccionamiento

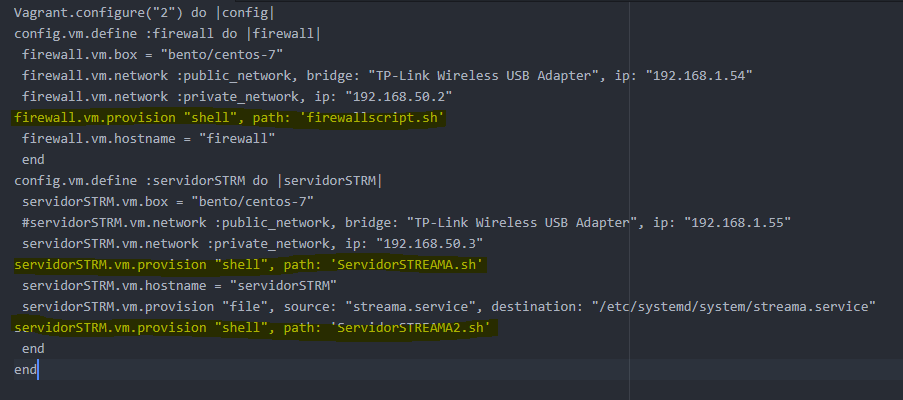
*firewall-cmd --zone="dmz" --add-forward-port=port=8080:proto=tcp:toport=8080:toaddr=192.168.50.2 --permanent*

*firewall-cmd --zone="internal" --add-forward-port=port=8080:proto=tcp:toport=8080:toaddr=192.168.50.2 --permanent*

firewall-cmd --reload

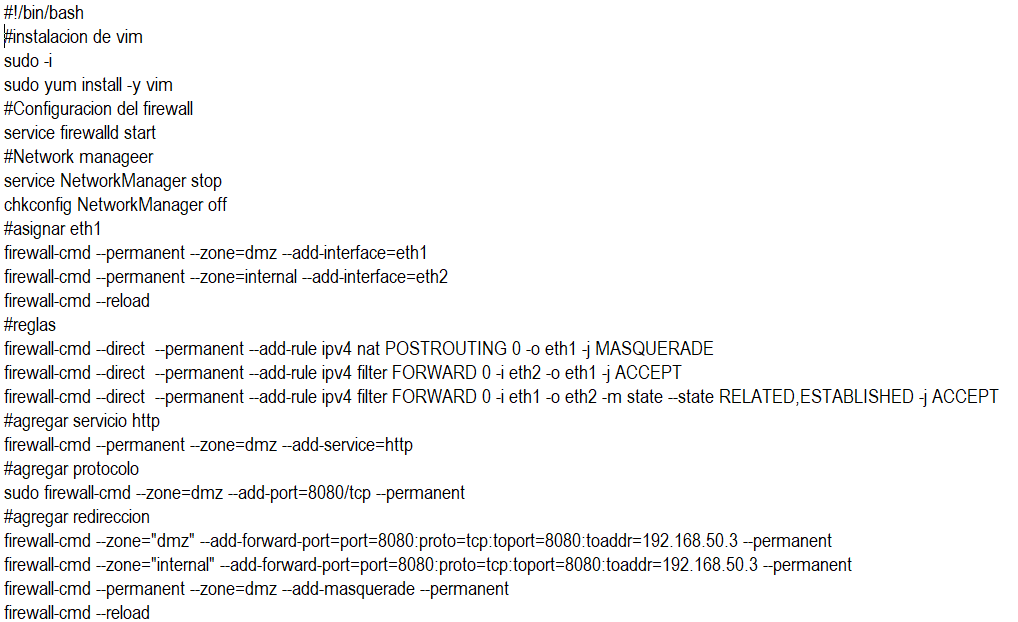
2. [1.0 Puntos (Funcionamiento + Sustentación)]] APROVISIONAMIENTO. Utilice los servicios de aprovisionamiento que provee Vagrant usando Shell para que los servicios del punto anterior (Firewall + Streama) queden provisionados de manera automática.

R// esta es nuestra configuración de vagrant, la cual le añadimos las líneas resaltadas. para que inicie y cargue todos los comandos



FIREWALL.SH

añadimos las interfaces de red, instalamos el editor de texto, agregamos el masquerade para que nuestros equipos puedan comunicarse, dirigimos el firewall con la ip del servidor y añadimos los puertos de conexión.



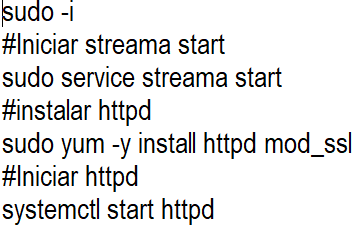
ServidorSTREAMA

Descargamos el repositorio del Wget, el java, posteriormente configuramos el steama y la ubicación de los videos



ServidorSTREAMA2

Iniciamos el servicio de streama y el httpd



3. [1.0 Puntos (Funcionamiento + Sustentación)]] API REST. Escriba un código en su lenguaje de programación preferido, que funcione como cliente de la aplicación de API REST + MYSQL suministrada en clase. Este código debe permitir generar peticiones GET, POST, PUT, DELETE y funcionar de manera análoga a curl o Postman.

R//

**Código en Python:**

# coding=utf-8

#import time

import requests

#import math

#import random

# MÉTODO PARA CONSULTAR LIBROS

def get\_request():

url = "http://192.168.60.3:5000/books"

req = requests.get(url=url)

# Processes results

status = req.status\_code

if status >= 400:

print("[ERROR] Status: ", status)

return False

print("[OK] Status: ", status)

print(req.text)

return True

# MÉTODO PARA AGREGAR LIBRO NUEVO

def post\_request(): #Nuevo

# Creates the headers for the HTTP requests

url = "http://192.168.60.3:5000/books"

headers = {"Content-Type": "application/json"}

# Makes the HTTP requests

#while status >= 400 and attempts <= 5: #Original

#while status >= 400:

identificador = raw\_input('Ingrese ID ')

title = raw\_input('Ingrese Titulo ')

description = raw\_input('Ingrese Descripción ')

author = raw\_input('Ingrese Autor ')

payload = {

"description": description,

"author": author,

"title": title

}

req = requests.post(url=url, headers=headers, json=payload)

status = req.status\_code

#attempts += 1

#time.sleep(1)

# Processes results

if status >= 400:

print("[ERROR] Status: ", status)

return False

print("[OK] Status: ", status)

return True

# MÉTODO PARA MODIFICAR LIBRO EXISTENTE

def put\_request():

identificador = raw\_input('Ingrese ID del libro modificar: ')

url = "http://192.168.60.3:5000/books/" + str(identificador)

headers = {"Content-Type": "application/json"}

title = raw\_input('Ingrese Titulo ')

description = raw\_input('Ingrese Descripción ')

author = raw\_input('Ingrese Autor ')

payload = {

"author": author,

"description": description,

"title": title

}

req = requests.put(url=url, headers=headers, json=payload)

# Processes results

status = req.status\_code

if status >= 400:

print("[ERROR] Status: ", status)

return False

print("[OK] Status: ", status)

return True

# MÉTODO PARA BORRAR UN LIBRO

def delete\_request():

identificador = raw\_input('Ingrese ID de libro a borrar: ')

url = "http://192.168.60.3:5000/books/" + str(identificador)

headers = {"Content-Type": "application/json"}

req = requests.delete(url=url)

# Processes results

status = req.status\_code

if status >= 400:

print("[ERROR] Status: ", status)

return False

print("[OK] Status: ", status)

return True

def main():

#El usuario digita el método a realizar

metodo = raw\_input('Método: ').upper()

if metodo == "POST":

print("Método POST")

post\_request()

#print("Método post\_request finalizado")

elif metodo == "GET":

print("Método GET")

get\_request()

#print("Método get\_request finalizado")

elif metodo == "PUT":

print("Método PUT")

put\_request()

#print("Método put\_request finalizado")

elif metodo == "DELETE":

print("Método DELETE")

delete\_request()

#print("Método delete\_request finalizado")

else:

print("¡El método digitado no es valido!")

if \_\_name\_\_ == '\_\_main\_\_':

i=0

while (i<5):

main()

i = i+1

El servidor API-REST lo proporcionó el profesor en el siguiente repositorio:

<https://github.com/omondragon/APIRestFlaskMySQL>

4. [0.5 Puntos] ENTREGA DE LOS RESULTADOS

a. Genere un proyecto publico en GitHub donde incluya todos los scripts y código fuente (separado por preguntas)

b. Agregue a classroom un comprimido con los scripts y código separado por preguntas.