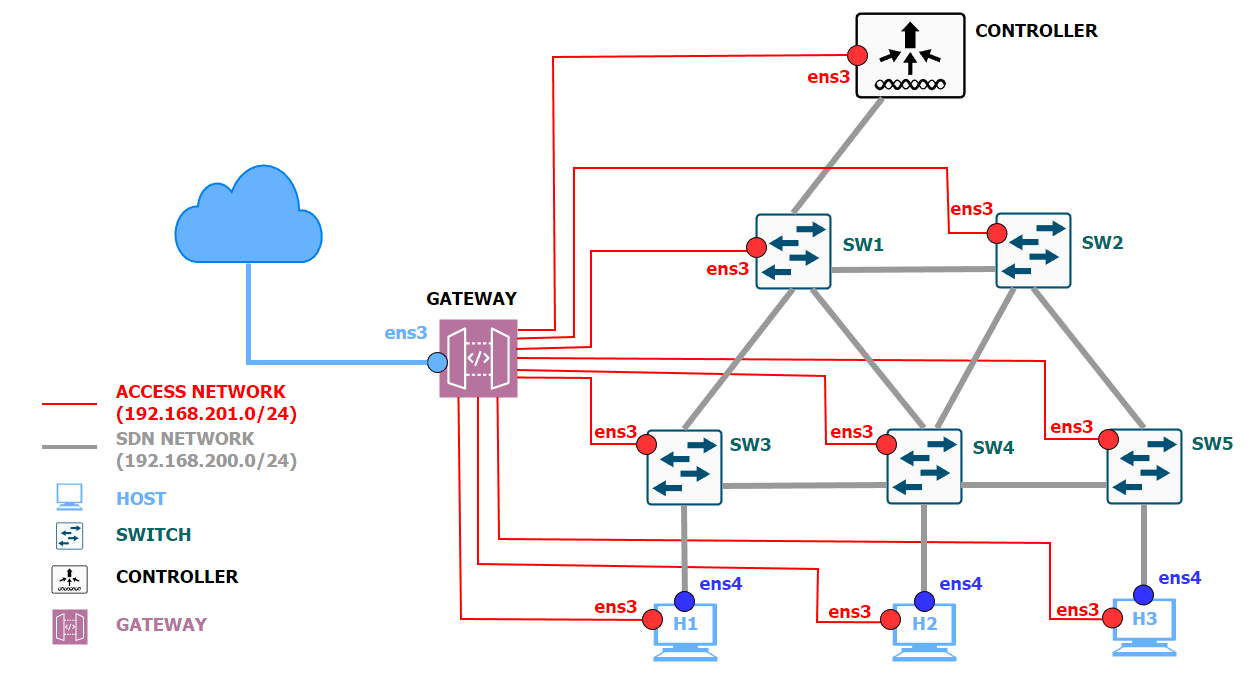
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| **TEL354 - REDES DEFINIDAS POR SOFTWARE (SDN)** | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| **Laboratorio N°:** | | | 4 |  |  | | **Semestre:** | 2025-2 |  |  |
|  | | | | | | | | | | |
| **Tema:** | | | Northbound API: Aplicación REST en Python | | | | | |  | |
|  | | | | | | | | | | |
| **Profesor:** Christian Quispe / César Santiváñez | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| **INDICACIONES GENERALES** | | | | | | | | | | |
|  | | | | | | | | | | |
| * Durante el laboratorio se puede usar el material provisto por los docentes y blogs de internet. | | | | | | | | | | |
| * Seguir atentamente las indicaciones de los jefes de práctica. | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| **OBJETIVOS** | | | | | | | | | | |
| * Adquirir herramientas necesarias para interactuar con el Northbound API (REST API) de un controlador SDN * Ser capaz de automatizar la interacción con un controlador SDN * Desarrollar una aplicación para el controlador Floodlight | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| **ACTIVIDADES A REALIZAR** | | | | | | | | | | |
|  | | | | | | | | | | |
|  | N° | ACTIVIDAD | | | | TIEMPO ESTIMADO | | PUNTAJE | |  |
|  | 1 | Experiencia de Laboratorio | | | | 1 hora y 30 minutos | | 5 ptos | |  |
|  | 2 | Informe Previo | | | | 4 días | | 5 ptos | |  |
|  | 3 | Cuestionarios | | | | 2 semanas | | 5 ptos | |  |
|  | 4 | Evaluación continua de PROYECTO / LABORATORIO | | | | 2 semanas | | 5 ptos | |  |

Jefe de práctica responsable: Juan Manuel Becerra Avila (a20090417@pucp.pe)

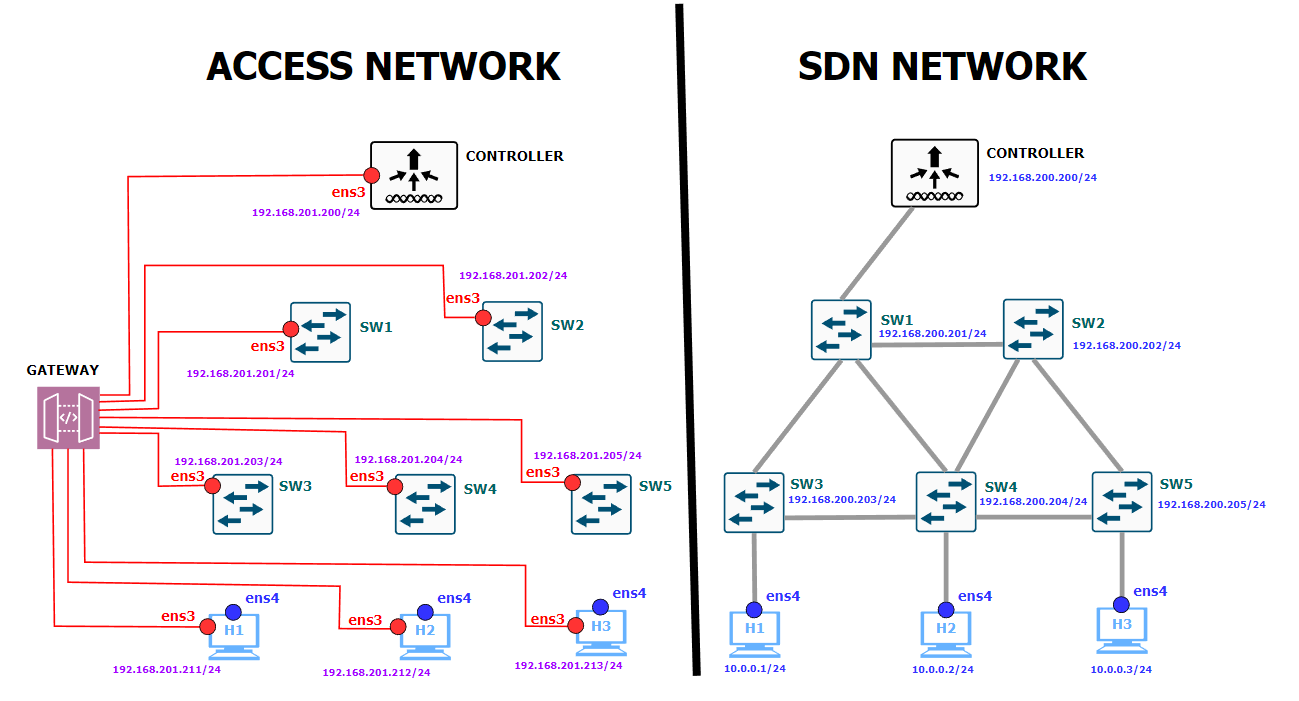
**ESCENARIO**

El Grupo de Investigación de Redes Avanzadas (GIRA-PUCP), perteneciente a la sección de Telecomunicaciones, cuenta con un emulador de redes avanzadas denominado VNRT, en el cual se ha desplegado la topología con la que se trabajará en el presente ciclo. A continuación se describen los componentes de la topología en cuestión:



Tal y como se puede observar, se tienen 10 máquinas virtuales distribuidas de la siguiente manera: 3 hosts (clientes de la red), 5 switches ([Open vSwitch](https://docs.openvswitch.org/en/latest/intro/what-is-ovs/)), un controlador([Floodlight Controller](https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview)) y un gateway (puerta de ingreso con accesos limitados).

Si desglosamos la topología mostrada líneas arriba, podemos identificar dos tipos de redes:



Para acceder a cada uno de los elementos dentro de la topología, es necesario establecer conexiones ssh a diferentes puertos previamente configurados.

|  |  |
| --- | --- |
| Virtual Machine | PORT |
| Gateway | 22 |
| CONTROLLER | 5800 |
| SW1 | 5801 |
| SW2 | 5802 |
| SW3 | 5803 |
| SW4 | 5804 |
| SW5 | 5805 |
| H1 | 5811 |
| H2 | 5812 |
| H3 | 5813 |

El comando a usar para tener un acceso vía ssh exitoso es:

|  |
| --- |
| $ ssh ubuntu@$ip\_gateway -p $port |

|  |
| --- |
| ✔ **Nota**  Las credenciales para acceder a la máquinas virtuales son:   * user: ubuntu * password: ubuntu |
| ✔ **Nota**  Debe reemplazar las variables **ip\_gateway** y **port** de acuerdo a su conveniencia. |

|  |
| --- |
| ✔ **Importante**  El usuario ubuntu tiene permisos de sudoers en todas las VMs a excepción del Gateway. Cualquier tipo de modificación dentro del Gateway está prohibida, recuerden que este dispositivo sólo será usado para acceder a sus demás VMs. |

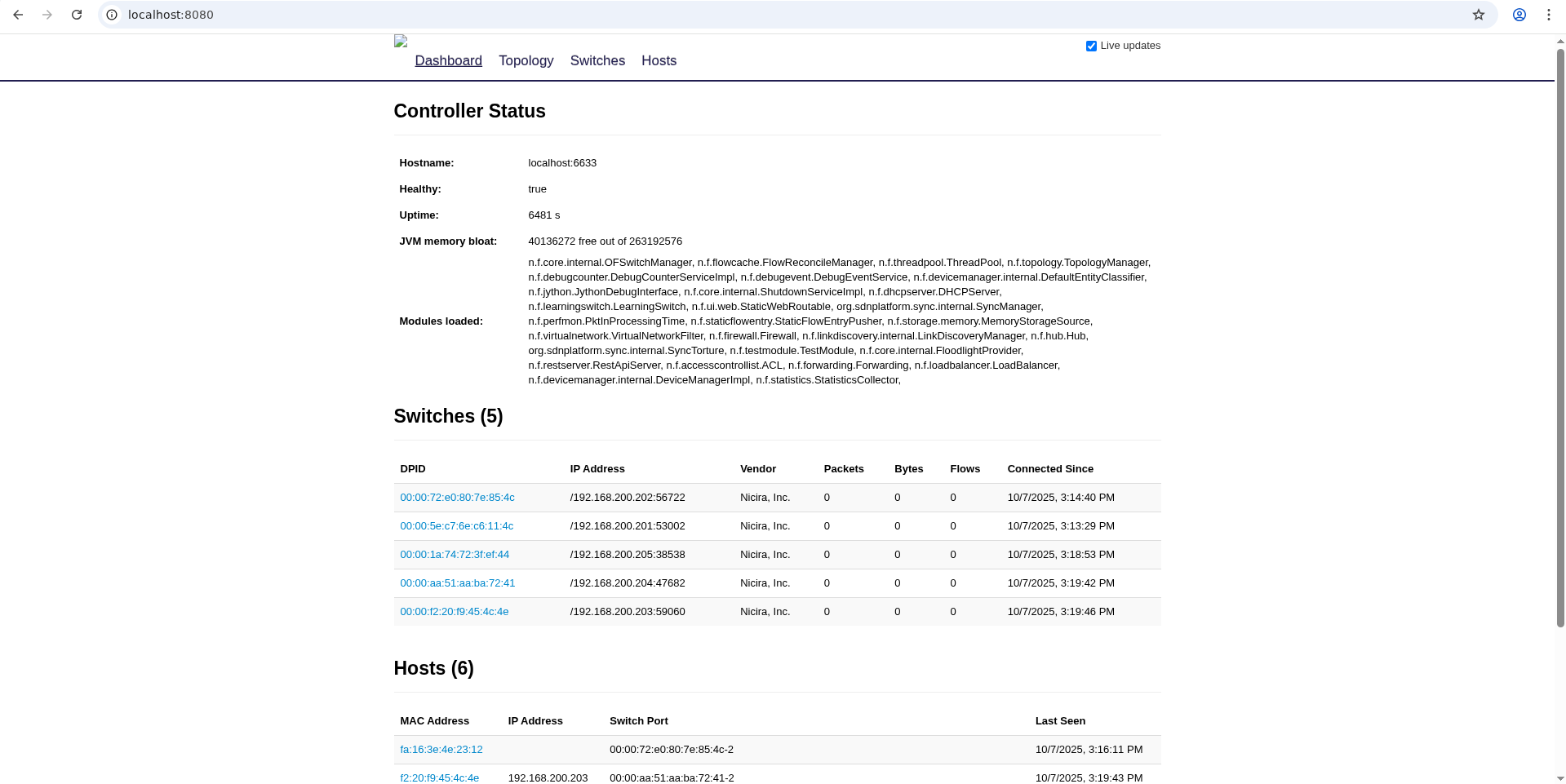
**INFORME PREVIO**

Indicaciones generales:

* **Puntaje total: 5 pts**
* El desarrollo del informe previo es personal, por lo que cada alumno deberá presentar un archivo en formato **(.pdf)** con sus respectivas respuestas.
* Fecha máxima de entrega del IP: **Martes 07 de Octubre a las 17:00 horas para ambos horarios (vía Google Classroom)**. No se podrán subir los informes después de dicha hora ni se recibirán por ningún otro medio. Sólo se podrá extender la fecha de presentación del informe previo a los alumnos en casos extraordinarios y bajo consentimiento expreso de los profesores del curso.
* Subir el informe previo a la actividad de CLASSROOM correspondiente (LABORATORIO 4 - INFORME PREVIO) con el siguiente formato: TEL354\_IP\_LAB4\_[YYY].pdf, donde [YYY] es su código PUCP.
* Es responsabilidad del alumno revisar y practicar los conceptos y herramientas indicados en el informe previo, pues estos serán vitales para el desarrollo de la experiencia en laboratorio.

|  |
| --- |
|  |

1. **Acceso a REST APIs utilizando curl y Python (1.5 pts)**
2. Acceder a la interfaz gráfica del controlador Floodlight: http://<IP\_CONTROLADOR>:8080/ui/index.html



1. ¿Podríamos considerar que la interfaz gráfica es una REST API?

Como tal la interfaz gráfica no es una REST API, es un cliente web que consume una REST API de Floodlight.

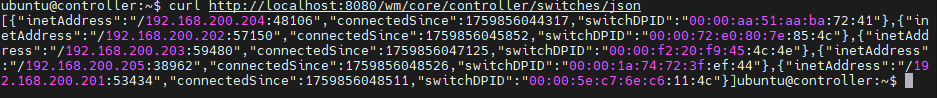
1. ¿Qué REST APIs podrían consultarse para obtener información similar a la mostrada en la pestaña Dashboard? Consultar la documentación de Floodlight de ser necesario. Pista: deben ser 6 APIs.

* /wm/device/
  + Lista todos los dispositivos detectados por el controlador. Esto incluye MACs, IPs y attachment points.
* /wm/core/controller/switches/json
  + Lista todos los switches conectados al controlador (DPID, IP, estado).
* /wm/core/module/loaded/json
  + Devuelve todos los modos activamente cargados por el module loader.
* /wm/core/health/json
  + Indica el estado/salud de la REST API.
* /wm/core/memory/json
  + Información del uso de memoria del controlador (memoria libre, usada)
* /wm/core/system/uptime/json
  + Tiempo de actividad (uptime) del controlador

1. Utilizando el comando curl, consultar las APIs para listar los devices detectados por el controlador

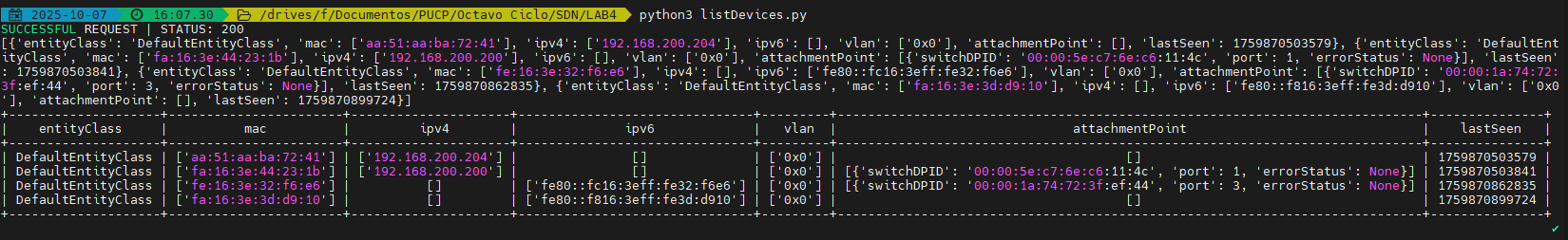


1. Utilizando el comando curl, consultar las APIs para listar switches

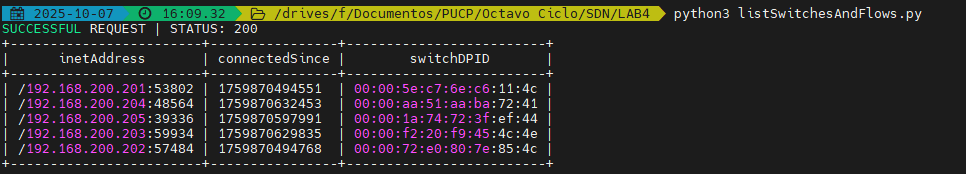


Acceder a la carpeta Reporte Final y usar los scripts en python3 para completar los siguientes incisos.

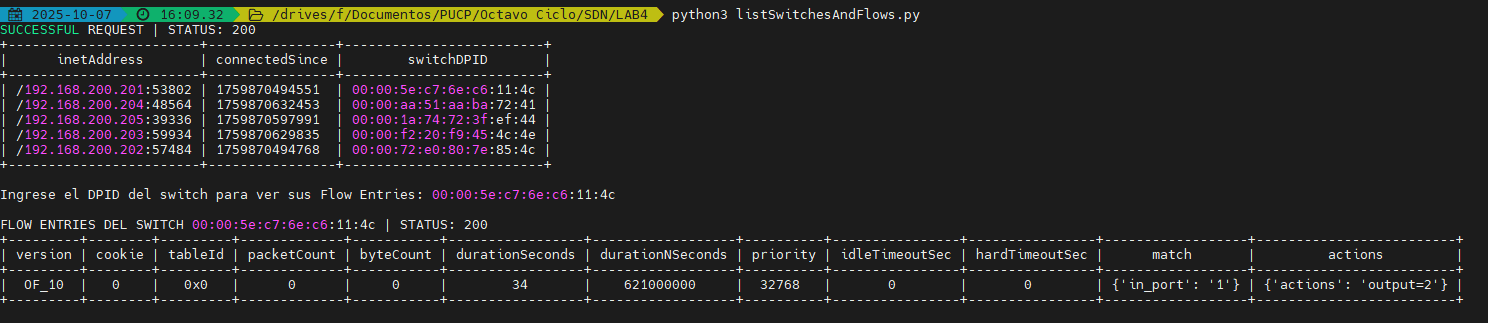
1. Modificar el script ‘listDevices.py' para consultar la API que lista los devices detectados por el controlador. Pista: solo tiene que definir las variables 'controller\_ip' (como la ip del head node de su slice) y ‘target\_api’ (como la api necesaria para hacer la consulta respectiva. Note que esta api ya la obtuvo en la pregunta 1e). Mostrar el output.



1. Modificar el script ‘listSwitchesAndFlows.py' para consultar la API que lista los switches. Pista: solo tiene que definir las variables 'controller\_ip' (como la ip del head node de su slice) y ‘target\_api’ (como la api necesaria para hacer la consulta respectiva. Note que esta api ya la obtuvo en la pregunta 1f). Muestre el output.



1. Ahora, usando el script que acaba de editar (‘listSwitchesAndFlows.py'), agregar código para que luego de listar los switches el nuevo script pida que el usuario ingrese el DPID de uno de los switches y al presionar enter debe mostrar los Flow Entries de tal switch
2. Compare las Flow Entries obtenidas al ejecutar el script resultante de la pregunta 1i con las mostradas en la interfaz gráfica del controlador (pestaña Switches). ¿Para un switch determinado, S2 por ejemplo, las Flow Entries que muestra el script son iguales a las que se observan en la interfaz gráfica? En caso no lo sean, ¿a que se podría deber la diferencia?



Sí, deberían ser iguales en la mayoría de casos. Sin embargo, podrían no ser idénticas debido a que el panel web no actualiza los flujos en tiempo real por tanto no puede mostrar datos del último polling de la API.

1. Asumiendo que usted está desarrollando una aplicación SDN donde el tiempo de respuesta ante una consulta es sumamente crítico, es decir, que la latencia de una consulta podría afectar en gran proporción la funcionalidad de su aplicación ¿Sería recomendable que su aplicación obtenga información a través de REST APIs?¿Por qué?¿En qué casos es preferible hacer uso de REST APIs? Indique al menos dos ejemplos

En una aplicación SDN donde el tiempo de respuesta es crítico, no se recomienda depender de las REST APIs, porque la latencia de red y el procesamiento HTTP pueden afectar el rendimiento. En cambio, las REST APIs son muy útiles para tareas de administración, monitoreo o integración entre sistemas externos y el controlador, donde la latencia no es determinante.

Ejemplos:

* Monitoreo y visualización de red: Herramientas externas consultan /wm/core/switches/json o /wm/device/ periódicamente para mostrar topología, tráfico o estado de enlaces.
* Gestión y configuración automatizada: Scripts agregan o eliminan flows mediante /wm/staticflowpusher/json, integrándose fácilmente con otros sistemas (Ansible, Prometheus, etc.).

1. **Clases y objetos en Python (1.5 pts)**

En un script llamado clases.py, implementar el código que realice lo siguiente:

* Definir clases que represente las siguientes entidades:
* Alumno
  + Atributos: nombre, PC (dirección MAC)
  + Métodos: ninguno
* Curso
  + Atributos: nombre, estado, alumnos, servidores (lista de servicios)
  + Métodos: agregar alumno, remover alumno, añadir servidor/servicio
* Servidor
  + Atributos: nombre, Dirección IP, lista de Servicio(nombre, protocolo, puerto)
  + Métodos: ninguno
* Servicio
  + Atributos: nombre, protocolo, puerto
* En la función main, instancie al menos un objeto de cada clase, llamando a sus constructores. Asimismo, utilizar por lo menos una vez los métodos de cada clase (en caso existan).

Sugerencia: revisar ejemplos básicos de clases en Python:

<https://www.w3schools.com/python/python_classes.asp>

Implementado en el archivo clases.py

1. **Lectura de archivos YAML en Python (1 pto)**

Leer el archivo datos.yaml de la carpeta ‘Informe Previo’. Usando un loop, recorrer la lista e imprimir en pantalla únicamente el nombre de los servidores

Sugerencia: descargue e instale la librería PyYAML:

<https://pypi.org/project/PyYAML/>

<https://pyyaml.org/wiki/PyYAMLDocumentation>

En el archivo yaml.py

1. **Librería Requests (1 pto)**

La librería requests proporciona una interfaz HTTP de alto nivel, por lo que las consultas REST se pueden realizar de forma más sencilla y con menos líneas de código.

Documentación: <https://requests.readthedocs.io/en/master/>

Utilizar la documentación de esta librería para realizar las siguientes consultas a la rest API de Floodlight

* Identificar el método para encontrar el punto de conexión de un host, compuesto por el DPID y el puerto del switch

El método para encontrar el punto de conexión de un host es:

GET http://<IP\_controlador>:8080/wm/device/

Este endpoint lista cada host con sus direcciones IP y MAC con los attachment points donde están conectados que se component por los DPIP y el Puerto del switch.

Ejm:

[

{

"mac": ["00:1A:2B:3C:4D:5E"],

"ipv4": ["10.0.0.1"],

"attachmentPoint": [

{

"switchDPID": "00:00:00:00:00:00:00:01",

"port": 1

}

]

}

]

* Implementar la consulta en una función de Python llamada get\_attachement\_points, que reciba como input la MAC de un host y retorne el switch DPID y el puerto

En el archivo requests\_example.py

* Identificar el método para encontrar la ruta entre un par de puntos de conexión (Switch DPID y puerto)

Floodlight tiene un endpoint para poder obtener la ruta entre dos switches.

GET http://<IP\_controlador>:8080/wm/topology/route/{src-dpid}/{src-port}/{dst-dpid}/{dst-port}/json

Y el resultado sería similar a esto:

[

{"src-switch":"00:00:00:00:00:00:00:01","src-port":1,"dst-switch":"00:00:00:00:00:00:00:02","dst-port":2},

{"src-switch":"00:00:00:00:00:00:00:02","src-port":3,"dst-switch":"00:00:00:00:00:00:00:03","dst-port":1}

]

* Implementar la consulta en una función de Python llamada get\_route, que recibe como input el DPID origen, puerto origen, DPID destino, puerto destino y retorne la lista de Switches y puertos que componen la ruta.

En el archivo requests\_example.py