



# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

REDES DE COMPUTADORAS

---

## Practica 04

---

Alumno David Pérez Jacome

*Profesor:* Paulo Santiago de Jesús Contreras Flores

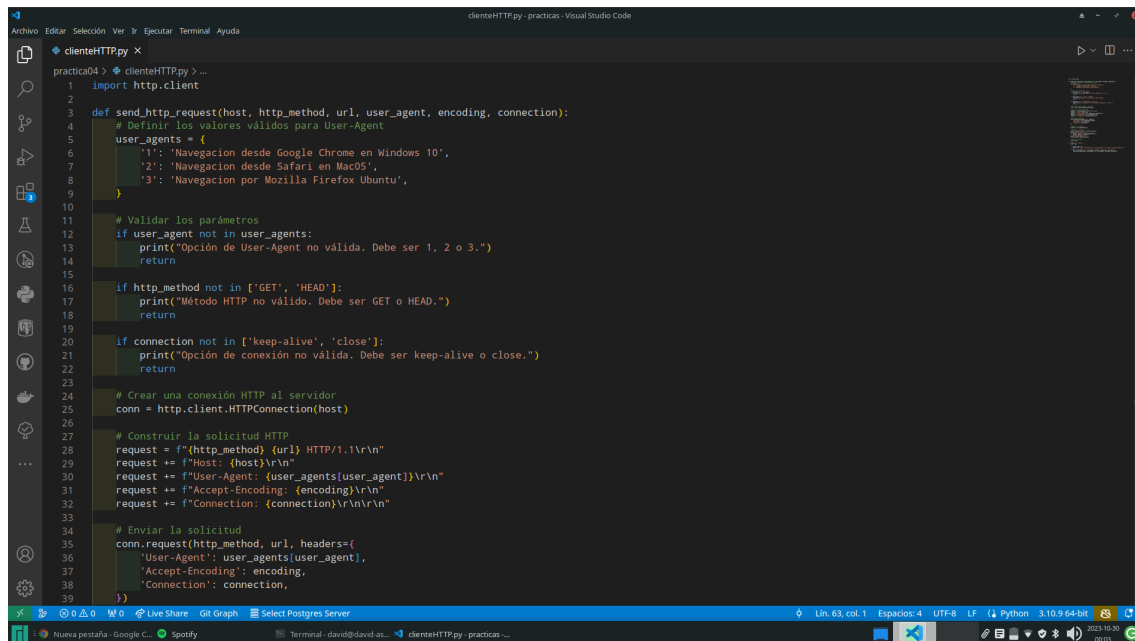
2023

## Reporte de practica de laboratorio.

### Desarrollo.

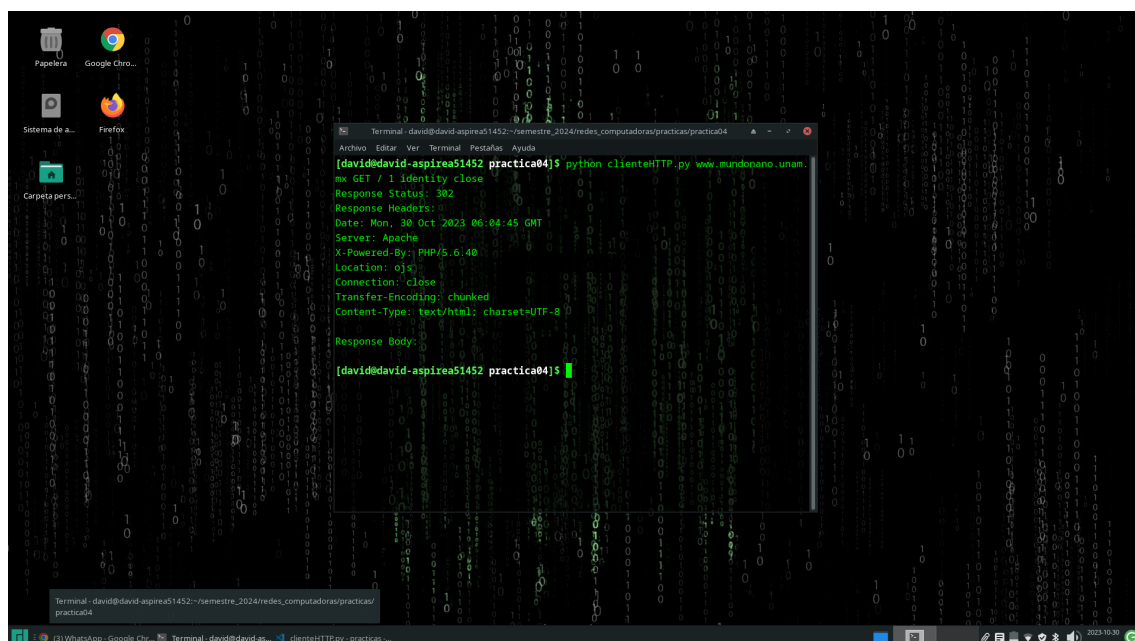
Es importante recalcar que este es el readme de la practica 04.

Para iniciar esta practica tomé como referencia el archivo que nos brindaron en la pagina de git. Con base a esto nos quedo el siguiente archivo:



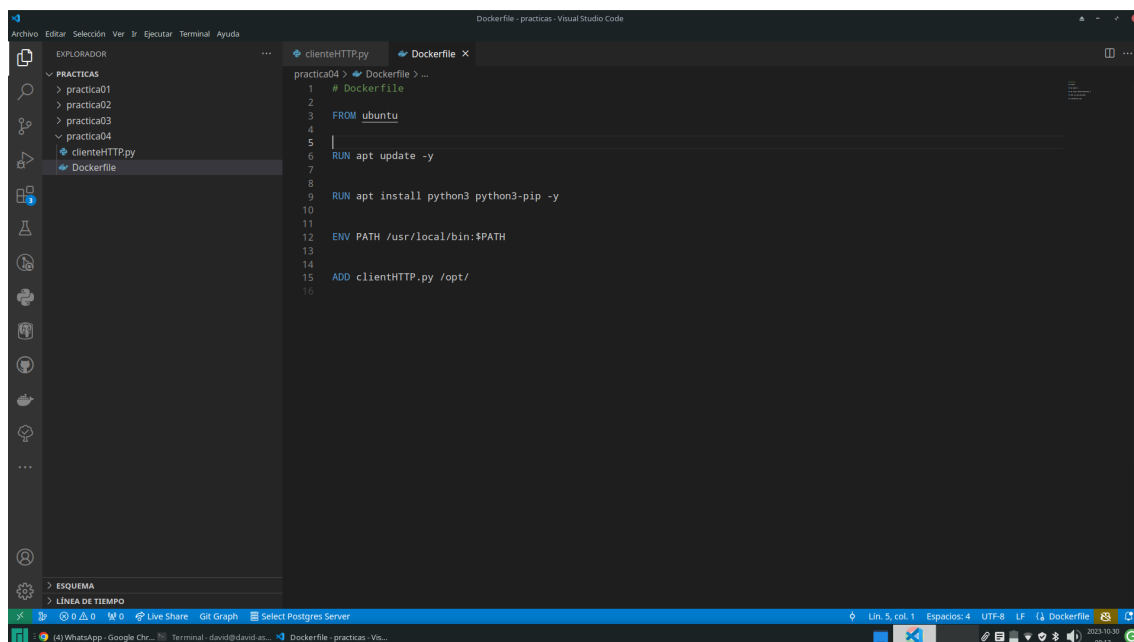
```
1 import http.client
2
3 def send_http_request(host, http_method, url, user_agent, encoding, connection):
4     # Definir los valores válidos para User-Agent
5     user_agents = {
6         '1': 'Navegacion desde Google Chrome en Windows 10',
7         '2': 'Navegacion desde Safari en MacOS',
8         '3': 'Navegacion por Mozilla Firefox Ubuntu',
9     }
10
11     # Validar los parámetros
12     if user_agent not in user_agents:
13         print("Opción de User-Agent no válida. Debe ser 1, 2 o 3.")
14         return
15
16     if http_method not in ['GET', 'HEAD']:
17         print("Método HTTP no válido. Debe ser GET o HEAD.")
18         return
19
20     if connection not in ['keep-alive', 'close']:
21         print("Opción de conexión no válida. Debe ser keep-alive o close.")
22         return
23
24     # Crear una conexión HTTP al servidor
25     conn = http.client.HTTPConnection(host)
26
27     # Construir la solicitud HTTP
28     request = f"({http_method}) {url} HTTP/1.1\r\n"
29     request += f"Host: {host}\r\n"
30     request += f"User-Agent: {user_agents[user_agent]}\r\n"
31     request += f"Accept-Encoding: {encoding}\r\n"
32     request += f"Connection: {connection}\r\n\r\n"
33
34     # Enviar la solicitud
35     conn.request(http_method, url, headers={
36         'User-Agent': user_agents[user_agent],
37         'Accept-Encoding': encoding,
38         'Connection': connection,
39     })
```

Después de esto probamos nuestro código con los parámetros seleccionados y entonces probamos que la ejecución es todo un éxito: **python clienteHTTP.py www.mundonano.unam.mx GET / 1 identity close**



```
[david@david-aspirea51452 practica04]$ python clienteHTTP.py www.mundonano.unam.mx GET / 1 identity close
Response Status: 302
Response Headers:
Date: Mon, 30 Oct 2023 06:04:45 GMT
Server: Apache
X-Powered-By: PHP/5.6.40
Location: 0's
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
Response Body:
[david@david-aspirea51452 practica04]$
```

Para la segunda parte de la practica simplemente solo creamos el siguiente archivo con base a nuestro sistema **ubuntu**



```
1 # Dockerfile
2
3 FROM ubuntu
4
5
6 RUN apt update -y
7
8
9 RUN apt install python3 python3-pip -y
10
11
12 ENV PATH /usr/local/bin:$PATH
13
14
15 ADD clienteHTTP.py /opt/
16
```

## Preguntas.

1. ¿Cuál es la función de los métodos de HTTP HEAD, GET, POST, PUT y DELETE?  
Los métodos HTTP son utilizados para indicar la acción que se desea realizar sobre un recurso web. A continuación se describen los métodos HTTP más comunes:
  - (a) HEAD: Es similar a GET, pero el servidor no devuelve el cuerpo de la respuesta. Se utiliza para obtener información sobre el recurso sin descargarlo.
  - (b) GET: Se utiliza para obtener un recurso web. El servidor devuelve el cuerpo de la respuesta.
  - (c) POST: Se utiliza para enviar datos al servidor para su procesamiento. El cuerpo de la solicitud contiene los datos que se envían al servidor.
  - (d) PUT: Se utiliza para actualizar un recurso web existente. El cuerpo de la solicitud contiene los nuevos datos que se deben almacenar en el servidor.
  - (e) DELETE: Se utiliza para eliminar un recurso web existente.
2. ¿Investigue y enliste junto con su significado las categorías de códigos de estado que usa HTTP?  
Los códigos de estado HTTP son utilizados por los servidores web para indicar el resultado de una solicitud HTTP. Los códigos de estado se dividen en cinco categorías:
  - (a) Respuestas informativas (100-199): Indican que la solicitud ha sido recibida y que el servidor está procesando la solicitud.
  - (b) Respuestas satisfactorias (200-299): Indican que la solicitud ha sido recibida, entendida y aceptada por el servidor.
  - (c) Redirecciones (300-399): Indican que el cliente debe tomar medidas adicionales para completar la solicitud.

- (d) **Errores del cliente (400-499):** Indican que la solicitud contenía información incorrecta o no pudo ser procesada por el servidor.
- (e) **Errores del servidor (500-599):** Indican que el servidor encontró un error al procesar la solicitud.

3. ¿Para qué se usan los campos `encoding` y `connection`?

Los campos `encoding` y `connection` se utilizan en las solicitudes HTTP para especificar cómo se deben codificar los datos de respuesta y cómo se debe establecer la conexión con el servidor, respectivamente. El campo `encoding` especifica qué algoritmo de compresión se debe utilizar para comprimir los datos de respuesta, mientras que el campo `connection` especifica si se debe mantener abierta o cerrar la conexión después de que se haya completado la solicitud.