Instituto Politécnico Nacional

IPN

Escuela Superior de Cómputo

ESCOM

Academia de Redes Aplicaciones para comunicaciones en red

6CV2

Práctica 8

“Seralización”

Alumna:

Navarrete Becerril Sharon Anette

Fecha de entrega: “14-Noviembre-2024”

Profesor: Ojeda Santillan Rodrigo

OBJETIVO

Se buscará en la presente práctica enviar el contenido de un documento de tipo PDF al servidor con el apoyo de la serialización siguiendo un proceso de serializado y que el servidor se va a encargar de procesar ese contenido y deserializarlo para agregarlo en un documento PDF nuevo.

INTRODUCCIÓN

La serialización es el proceso mediante el cual un objeto o estructura de datos se convierte en un formato que puede ser fácilmente almacenado o transmitido, y posteriormente reconstruido en su estado original. Este proceso implica transformar la representación interna de los datos en una secuencia de bytes o en un formato de texto, permitiendo su almacenamiento en archivos, su envío a través de una red o su guardado en bases de datos. La deserialización es el proceso inverso, donde esos bytes o texto se convierten nuevamente en el objeto o estructura de datos original.

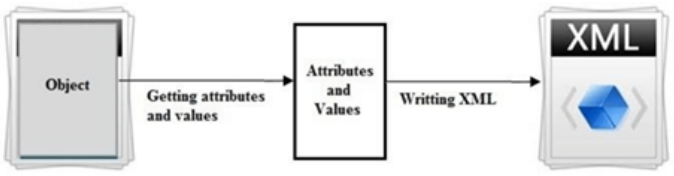
Algunas de las características clave de la serialización incluyen:

- **Compatibilidad**: Permite que datos estructurados se intercambien entre diferentes lenguajes de programación.
- **Persistencia**: Permite guardar el estado de un objeto en un archivo o base de datos para su futura recuperación.
- **Portabilidad**: Facilita la transferencia de datos entre diferentes sistemas y plataformas.
- **Eficiencia**: Optimiza el almacenamiento y la transmisión de datos complejos.
- **Flexibilidad**: Admite varios formatos de serialización, como binario o texto.

Las aplicaciones más comunes de la serialización son:

- **Almacenamiento de datos**: Guardar el estado de objetos en bases de datos o archivos.
- **Comunicación entre procesos**: Transferir datos entre procesos dentro de una misma máquina o entre diferentes máquinas.
- **Transmisión de datos en redes**: Intercambiar datos estructurados entre clientes y servidores en aplicaciones distribuidas.
- **Interoperabilidad entre lenguajes**: Facilitar el intercambio de datos entre programas escritos en distintos lenguajes de programación.
- **Caching**: Almacenar objetos en memoria caché para mejorar el rendimiento.

El proceso de serialización comienza con un objeto en su estado original, que se transforma en un formato fácilmente almacenable o transmisible, como JSON, XML o binario. Esto convierte la representación interna del objeto en una secuencia de bytes o texto, que puede almacenarse o transmitirse. La deserialización, por su parte, es el proceso por el cual esos datos serializados se reconstruyen en su forma original, recuperando toda la información y el estado del objeto inicial.



DESARROLLO

Para esta práctica se utilizará Flask como microframework para la facilidad de enviar peticiones de tipo POST y request es una biblioteca en Python que tiene como funcionalidad es facilitar el envío de peticiones.

Después de haber instalado el microframework y la biblioteca de Python, se puede proceder con el desarrollo de la práctica. El primer paso es añadir la lógica necesaria para el cliente, que será responsable de enviar el documento PDF. Para ello, se debe incluir el siguiente bloque de código en el archivo P_serial_cliente.py:

Código del cliente:

```
import pickle
import requests

# Nombre del archivo PDF que se va a enviar
archivo_pdf = 'documento.pdf'

try:
    # Leer el archivo PDF
    with open(archivo_pdf, 'rb') as f:
        pdf_data = f.read()
        print(f"Tamaño del archivo PDF leído: {len(pdf_data)} bytes")

    # Serializar el contenido del PDF
    try:
        serialized_data = pickle.dumps(pdf_data)
        print(f"Tamaño de los datos serializados: {len(serialized_data)} bytes")
    except pickle.PicklingError as e:
        print(f"Error al serializar el archivo PDF: {e}")
        exit(1)

    # Enviar el contenido serializado al servidor
    try:
        response = requests.post(
            'http://localhost:8080/prueba_serializacion',
            files={'file': (archivo_pdf, serialized_data)}
        )

        # Imprimir la respuesta del servidor
        print("Respuesta del servidor:", response.status_code)
        print(response.text)

    except requests.exceptions.RequestException as e:
        print(f"Error al enviar la solicitud al servidor: {e}")

except FileNotFoundError:
     print(f"Error: El archivo '{archivo_pdf}' no se encontró. Asegúrate de que el archivo está en la misma carpeta que este script.")
except Exception as e:
    print(f"Error inesperado: {e}")
```

Primero se tiene el siguiente contenido dentro de un documento de tipo PDF: *"La serialización es el proceso de convertir un objeto en un formato que se puede almacenar o transmitir y luego reconstruirlo en su forma original. En otras palabras, se trata de transformar una estructura de datos o un objeto en una secuencia de bytes que se puede guardar en un archivo, enviar a través de una red, o almacenar en una base de datos."* Entonces una vez teniendo dicho documento se procede a ejecutar nuestro servidor.

Este bloque de código se encarga de leer un archivo PDF, serializar su contenido utilizando la biblioteca pickle, y enviar los datos serializados a un servidor mediante una solicitud POST utilizando la biblioteca requests. Pickle permite transformar objetos de Python en una secuencia de bytes para su transmisión o almacenamiento, mientras que requests es una biblioteca popular que facilita la realización de solicitudes HTTP en Python. A continuación, se procederá a definir la lógica del servidor, añadiendo el siguiente fragmento de código en el archivo P_serial_servidor.py como se muestra:

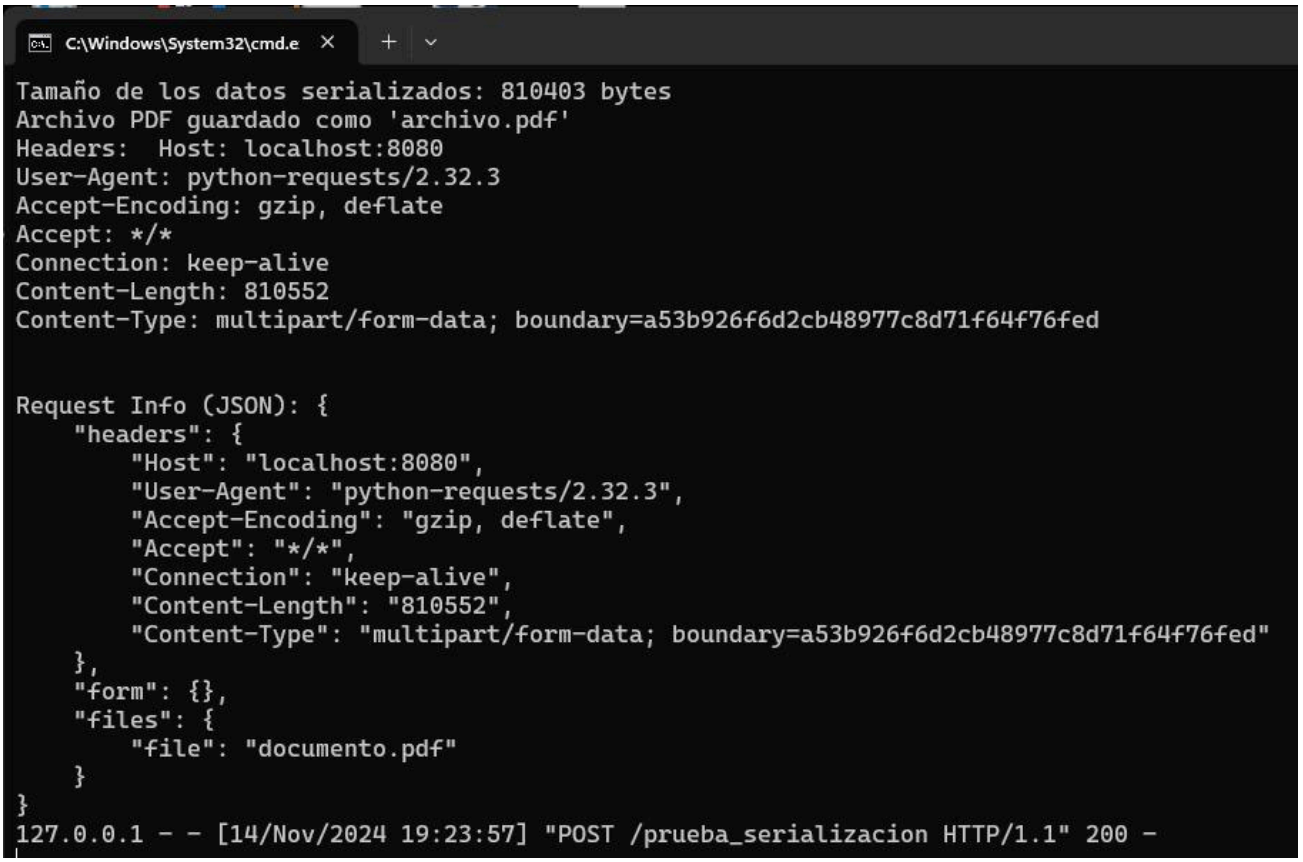
Código del sevidor:

```
 from flask import Flask, request
import pickle
import json

app = Flask(_name_)

@app.route('/prueba_serializacion', methods=['POST'])
def upload():
    try:
        # Verificar si el archivo 'file' está presente en la solicitud
        if 'file' not in request.files:
            return 'No se ha enviado ningún archivo', 400

        # Obtener el archivo serializado
        serialized_data = request.files['file'].read()
        print(f"Tamaño de los datos serializados: {len(serialized_data)}
bytes")

        # Deserializar el contenido
        try:
            pdf_data = pickle.loads(serialized_data)
        except Exception as e:
            print(f"Error al deserializar el archivo: {e}")
            return 'Error al deserializar el archivo', 400

        # Guardar el archivo PDF deserializado
        with open('archivo.pdf', 'wb') as f:
            f.write(pdf_data)

        print("Archivo PDF guardado como 'archivo.pdf'")

        # Imprimir los headers
        print("Headers: ", request.headers)

        # Crear un diccionario con la información relevante del request
        request_info = {
```

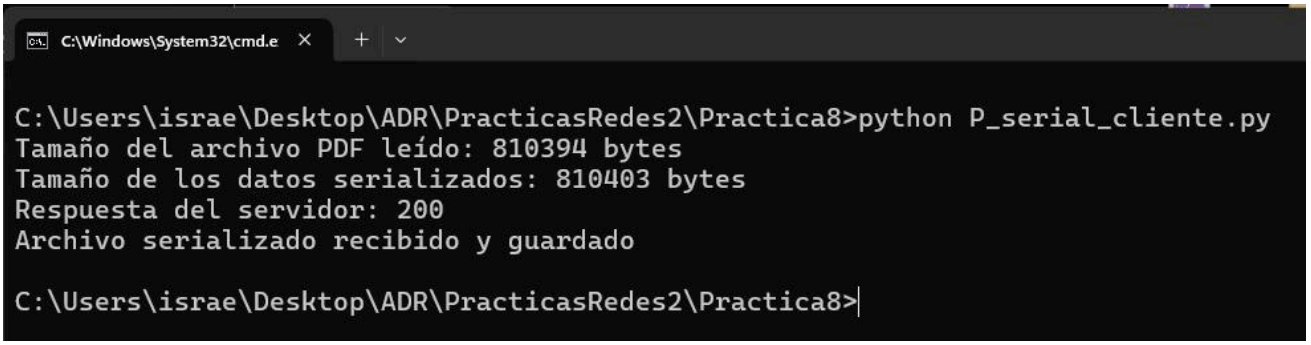
```
            'headers': dict(request.headers),
            'form': request.form.to_dict(),
            'files': {file_name: file.filename for file_name, file in
request.files.items()}
        }

        # Imprimir el mensaje en formato JSON
        print("Request Info (JSON):", json.dumps(request_info, indent=4))

        return 'Archivo serializado recibido y guardado', 200

    except Exception as e:
        print(f"Error general: {e}")
        return 'Error en el servidor', 500

if _name_ == '_main_':
    app.run(port=8080)
```

Capturas de funcionamiento:

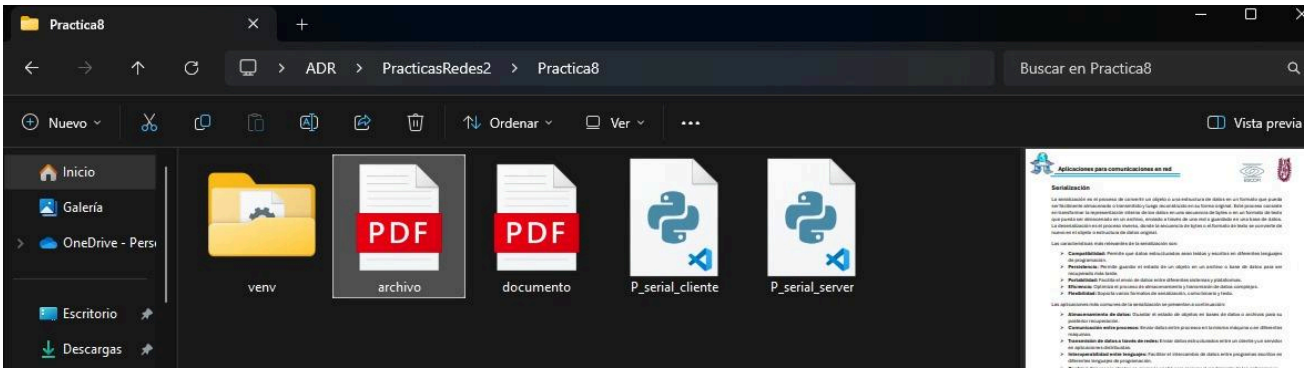
Servidor:



Cliente:

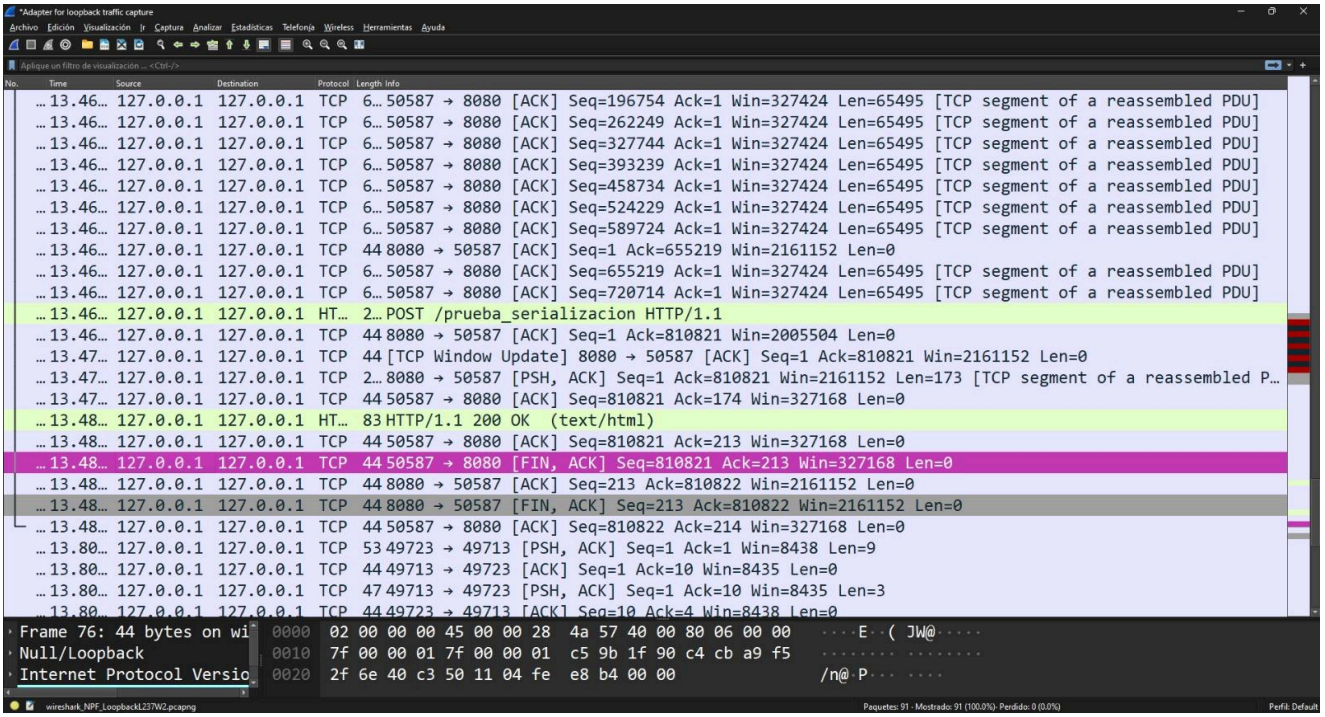


Transmisión del pdf:



Análisis de tráfico con Wireshark

En Wireshark se puede observar que se tiene la respuesta del endpoint:



CONCLUSIONES

Navarrete Becerril Sharon Anette:

Serialización como método clave para el manejo de datos: La serialización mediante pickle se presenta como una herramienta esencial en el manejo y envío de datos complejos. Este método permite que los datos sean empaquetados y transmitidos de forma eficiente entre diferentes partes de una aplicación, destacando su importancia en sistemas distribuidos o en redes que necesitan intercambiar grandes cantidades de información de forma rápida.

BIBLIOGRAFÍA

Stevens, W. Richard. Programación en red con Unix: La API de Sockets. Addison-Wesley Professional, 2003.

Forouzan, Behrouz A. Comunicaciones de Datos y Redes. McGraw-Hill, 2012. Tanenbaum, Andrew S., y David J. Wetherall. Redes de Computadoras. Pearson, 2010.

Comer, Douglas E. Internetworking con TCP/IP Volumen Uno. Prentice Hall, 2006. Kurose, James F., y Keith W. Ross. Redes de Computadoras: Un Enfoque Descendente. Pearson, 2017.

Beazley, David, y Brian K. Jones. Python Cookbook: Recetas para Dominar Python 3. O'Reilly Media, 2013. Begg, A. Sockets TCP/IP en C: Guía Práctica para Programadores. Morgan Kaufmann, 2000