Práctica 1.3

Silviu Constantin Sofrone

- **4servidorWeb.py**

```
!/usr/bin/env python3
# -*- coding: cp1252 -*-

import sys
import socket
import select
import re #expresiones regulares
import os
import signal
import time
import requests
import pathlib

# Manejador de la señal SIGALARM
def timer(signum, stack):
        print('Servidor web inactivo')

if len(sys.argv) != 3:
        print('Usage:', sys.argv[0], '<Modo> <Server Port>\n')
        exit(1)

mode = int(sys.argv[1]) # El segundo parámetro establece el modo en el que va a trabajar el servidor (1 para el modo persistente, y 0 para no persistente)

if mode == 1:
        print('Modo persistente\n')
```

```
elif mode == 0:
        print('Modo no persistente\n')
else:
        print('Valor introducido inválido:\r')
        print('Modo persistente = 1\r')
        print('Modo no persistente = 0\n')
        exit(1)

ServPort = sys.argv[2]

# Socket TCP del servidor
ServSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Configurarlo para que no se bloquee
ServSock.setblocking(0)

# Unimos (bind) socket al puerto
ServSock.bind(('', int(ServPort)))

# Escucha conexiones entrantes
ServSock.listen(10)

# Sockets que van a ready_to_read
entrantes = [ServSock]

# Sockets que van a ready_to_write
salientes = []

# Establecemos un manejador para la señal SIGALRM. La función que maneja la señal es timer
signal.signal(signal.SIGALRM, timer)

# Cuando pasen 10 segundos se solicita que SIGALARM envié una señal a su manejador
```

```
# Utilizando el timer de la función select, el texto de servidor inactivo se muestra cada 10 segundos llenando la pantalla con demasiada información. Por ello he utilizado la señal de alarma, en vistas a mostrar el mensaje solo cuando haya inactividad

signal.alarm(10) # Inicializamos la alarma aquí, ya que el servidor web puede estar corriendo pero no recibir ninguna petición


while entrantes:

        ready_to_read, ready_to_write, error = select.select(entrantes, salientes, entrantes)

        #if ready_to_read == []:

                #print('Servidor web inactivo\n')

        for s in ready_to_read:

                if s is ServSock: #Si s es el socket del servidor aceptamos las conexiones de los clientes que entran al chat

                        connection, clntAddr = ServSock.accept()

                        print( 'Conexión con:', clntAddr)

                        print()

                        entrantes.append(connection) # Añadimos a la lista de entrantes el nuevo cliente

                else:

                        recvdata = s.recv(5000).decode()

                        if recvdata:

                                signal.alarm(0) #En caso de recibir datos (una petición) entendemos que el servidor ya no va a estar inactivo (tiene trabajo que hacer)-> quitamos la alarma para que no siga corriendo desde donde se ha quedado

                                print(recvdata) # Muestro la petición recibida

                                pieces = recvdata.split() # Separo la petición por espacios


                                metodo = pieces[0] # El primer elemento tras hacer split() es el método (GET, POST,...)

                                solicitud = pieces[1] # Luego va el objeto solicitado

                                solicitud = solicitud.lstrip('/') #Elimina el "/" inicial

                                version = pieces[2] # Y la versión de http empleada


                                print()
```

```
                print('Solicitud de ',s.getpeername())


                # Si el método empleado es GET:
                if metodo=='GET':
                    if(solicitud == ''):
                        solicitud = 'index.html' # Se solicita el archivo index.html por defecto


                    if '.jpg' in recvdata: # Si encuentro '.jpg' dentro de la petición identifico que se ha solicitado una imagen
                        tipo = 'image/jpg' # Por tanto, el tipo de recurso será image/jpg
                        print('Solicitud:', solicitud)
                    # Realizo un procedimiento parecido para la identificación del recurso solicitado mediante expresiones regulares
                    else:
                        if re.findall('[.]txt$', solicitud):
                            tipo = 'plain/text'
                            print('Solicitud', solicitud)
                        else:
                            tipo = 'text/html'
                            if re.findall('[.]html$', solicitud):
                                print('Solicitud:', solicitud)
                            else:
                                solicitud += '.html' # En caso de que el recurso solicitado no tenga una extensión reconocida con anterioridad, se da por hecho que se trata de un html
                                print('Solicitud:', solicitud)


                    # Me aseguro de no haber cometido ningún error, y en caso de que el recurso solicitado sea un archivo de texto, pero le haya añadido de forma errónea '.html' como extensión, borro dicha extensión
                    if '.txt' in recvdata:
                        tipo = 'text/plain'
```

```
                    if re.findall('[.]html$', solicitud):
                        solicitud = pathlib.Path(solicitud).with_suffix('.html')
                        print('Solicitud:', solicitud)

                try: # Intentamos abrir el recurso solicitado y enviar su información contenida
                    f = open(solicitud, 'rb')
                    bytes_f = f.read() # Leemos el archivo y codificamos su información para ser enviada
                    f.close()

                    tam = os.stat(solicitud).st_size # Tamaño del archivo
                    # Creación de las cabeceras de respuesta
                    data =  version + ' 200 OK\r\n' # Indicamos la versión obtenida de la solicitud y el código de respuesta
                    data += 'Content-type: ' + str(tipo) + '\r\n' # En esta cabecera indicamos el tipo del recurso solicitado
                    data += 'Content-length: ' + str(tam) + '\r\n' # Importante la longitud ya que el navegador debe saber cuando has acabado de mandar información, sino no se podrá implementar el modo persistente
                    data += '\r\n'

                # En caso de que el recurso solicitado no se encuentre en el servidor, se envía el archivo 'error_404.html', que indica que la página solicitada no se encuentra
                except FileNotFoundError:
                    f = open('error_404.html', 'rb')
                    bytes_f = f.read()
                    f.close()

                    tam = os.stat('error_404.html').st_size
                    data = version + ' 404 Not Found\r\n'
```

```
                                        data += 'Content-type: text/html\r\n'
                                        data += 'Content-length: ' + str(tam) + '\r\n'
                                        data += '\r\n'

                                    print()
                                    print(data) # Mostramos las cabeceras de respuesta en el servidor

                                    # Finalmente codificamos las cabeceras, añadimos al final los datos del archivo leído y mandamos todo al cliente
                                    final_data = data.encode()
                                    final_data += bytes_f
                                    s.sendall(final_data)

                                if metodo=='POST':

                                    # En este caso, solo trataré los datos datos codificados en tuplas llave-valor
                                    if 'application/x-www-form-urlencoded' in recvdata:
                                        tipo = 'text/html'
                                        if re.findall('[.]html$', solicitud):
                                            print('Solicitud:', solicitud)

                                    # Recojo los datos enviados por el método POST ubicados al final de la petición y los separo haciendo un split() con '&', dado que es el separador que utiliza dichos datos
                                    datapost = 'Datos: '
                                    for i in range (0, len(pieces)):
                                        if i == (len(pieces) - 1):
                                            print('Datos POST:' + str(pieces[i]) + '\n')
                                            postdata = str(pieces[i])
                                            postdata = postdata.split('&')
```

```
                print('Separado &:', postdata)
                for i in range (0, len(postdata)):
                    datapost += postdata[i] + '\n'

            # Guardo dichos datos en un archivo de texto para luego mostrarlos mediante el programa 'respuesta.html'
            # Los formularios html albergados en el servidor envían los datos introducidos por el usuario a 'respuesta.html', que a su vez mostrará dichos datos
            # Mediante este procedimiento, trato de realizar una funcionalidad que sirva para múltiples casos independientemente de su naturaleza. Es decir, independientemente del tipo de formulario el resultado será mostrar sus datos
            try:
                f = open('datos.txt', 'w')
                f.write(datapost)
            except IOError:
                print('Fichero no accesible\n')
            finally:
                f.close()

            # Procedimiento análogo para enviar los datos al cliente
            try:
                f = open(solicitud, 'rb')
                bytes_f = f.read()
                f.close()

                tam = os.stat(solicitud).st_size
                data =  version + ' 200 OK\r\n'
                data += 'Content-type: ' + str(tipo) + '; charset=UTF-8\r\n'
                data += 'Content-length: ' + str(tam) + '\r\n'
                data += '\r\n'
```

```
                except FileNotFoundError:
                    f = open('error_404.html', 'rb')
                    bytes_f = f.read()
                    f.close()

                    print('Solicitud:', solicitud)

                    tam = os.stat('error_404.html').st_size
                    data = version + ' 404 Not Found\r\n'
                    data += 'Content-type: text/html\r\n'
                    data += 'Content-length: ' + str(tam) + '\r\n'
                    data += '\r\n'

                print()
                print(data)

                final_data = data.encode()
                final_data += bytes_f
                s.sendall(final_data)


            if(mode == 0): # En caso de que el modo seleccionado sea no persistente, se cierra la coneción con el cliente en cuestión y se le elimina de la lista de entrantes
                print('Modo no persistente, cierro conexión\n')
                entrantes.remove(s)
                s.close()
            signal.alarm(10) #Una vez terminado su trabajo, volvemos a establecer la alarma a 10 segundos (la reiniciamos ya que antes la pusimos a 0)
        else:
            if s != ServSock:
                print('Ya no hay conexión con:',s.getpeername())
```

```
                                        entrantes.remove(s) #En caso de perderse la conexión con algún cliente se le expulsa de la lista
                                        s.close() #Y se cierra su socket asociado
```

- **Ejemplo de formulario**

```
<html>
    <body>

        <form action="respuesta.html" method="post">
            Nombre: <input type="text" name="nombre"><br>
            Grupo: <input type="text" name="grupo"><br>
            <input type="submit">
        </form>

    </body>
</html>
```

- **Respuesta.html**

```
<html>
    <body>

            <h3>Datos del formulario</h3>
            <p><iframe src="datos.txt" frameborder="0" height="400" width="95%"></iframe></p>

    </body>
</html>
```