PrÁctica 3 Python

Echo HTTP Server

Andrés Casado García

Adrián Pérez Fernández

Álvaro Gala Martínez

Daniel Montero García

# INTRODUCCIÓN

El objetivo de este ejercicio, al igual que lo fue el ejercicio 2, supone una mejora de la funcionalidad respecto a los ejercicios anteriores. En este caso, la mejora consiste en modificar el código del servidor de tal manera que procese una request de tipo GET realizada por el cliente (que en este caso será el propio navegador web). Dicha request tiene un mensaje el cual deberá ser procesado por el servidor y enviado de vuelta al cliente en la response para que se imprima en la página del navegador.

# SERVIDOR – Nueva funcionalidad

En esta sección se tratará de explicar las diferencias funcionales del código del servidor respecto a la versión que propusimos en el ejercicio 1.

Para empezar en esta nueva nuestro servidor no procesará la request paquete a paquete por lo que el segundo bucle **while**, el cual servía para mantener activo el servidor hasta haber recibido todos los paquetes enviados por el cliente, desaparece. En su lugar simplemente hacemos uso del método **recv** del socket para recibir la request entera.

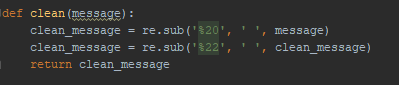


Después, haremos uso del método creado en el ejercicio 2 **requests()**, el cual procesará la request, después de ser decodificada, y le dará el formato requerido en este caso en forma de diccionario. Dicho diccionario lo guardaremos en la variable request.



A continuación, obtenemos del diccionario el mensaje mandado por el cliente y lo limpiamos de caracteres especiales que se generan en al realizar la request. Esto lo realizamos con una función llamada **clean()**, la cual hemos creado nosotros.





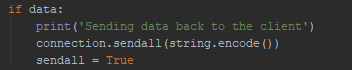
Luego obtenemos el tamaño del mensaje para poder darle un valor apropiado al content-length, un campo de la response que vamos a generar.



Ahora, generamos un string con el formato que debe tener nuestra response. En dicho string se insertarán tanto el mensaje limpio que hemos obtenido del diccionario, como la longitud de dicho mensaje para que aparezca en el campo content-length.

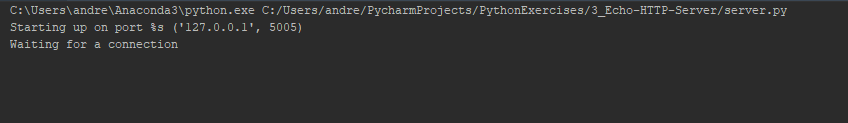


Finalmente, si hemos recibido datos del cliente, le devolvemos la response que hemos creado después de codificarla con el método **encode()**.

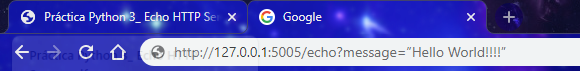


# SALIDAS

Arrancamos el servidor y se queda en espera por una request.



Introducimos la petición en el navegador.



Mostramos por pantalla el diccionario.



Mostramos el mensaje de vuelta en el cliente.

