

## TRABAJO SERVIDOR UDP

```
31 def __init__(self, *args, **kwargs):
32     self.file = None
33     self.fingerprints = set()
34     self.logdupes = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, 'requests.log'), 'a')
39         self.file.seek(0)
40         self.fingerprints.update([e.rstrip() for e in self.file.readlines()])
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('SUPERFILTER_DEBUG')
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)
```

**Participantes:** Pablo Garcia, Tobias Vidal y Youssef Khattala.

### **Clase Clientes:**

En la clase clientes comenzamos importando todo lo que necesitamos, seguido de eso podemos ver una función llamada 'enviar' la cual se encargará de enviar los mensajes que el cliente desee al servidor, junto a su dirección, debajo de dicha función nos encontramos con otra función llamada 'recibir', esta a diferencia de la otra recibirá todos los mensajes que se envíen desde el servidor. Luego de eso nos encontraremos con la ip, el puerto y con la creación del el host y por último la creación del socket.

Ya terminando vemos un humilde menu de navegacion que te permite loguearte al servidor o crear usuario, donde si el cliente selecciona la opción 'y', accedera a la opcion de creacion de usuario donde se le pedirá un nombre para el mismo, una vez el cliente haya puesto un nombre se le enviará al servidor un mensaje con la finalidad de que el servidor sepa lo que estamos haciendo y le genera un token al usuario, si el cliente selecciona la opción 'n' el programa le pedirá que ingrese su token de usuario ya existente para poder conectarse al servidor, por último si no selecciona ninguna de las anteriores saldrá un mensaje poniendo que los datos han sido erróneos y al final del código se ven la creación de los hilos y los .start de los mismos.

## CLASE SERVIDOR:

En la clase servidor al igual que en la de clientes comenzamos importando todo lo que vayamos a necesitar, luego creamos una variable de tipo file y la asociamos a la ruta donde se deberá crear o ir a buscar nuestro archivo .json, siguiendo el código veremos la primer función llamada 'buscarUsuario' esta recibe un token y comprueba si existe en el archivo JSON. Debajo de la misma nos encontramos con otra función llamada 'logueo', esta recibe datos del cliente a través del socket del servidor.

Token contiene el token enviado por el cliente y la dirección contiene la dirección del cliente. Luego en el bloque try except intentamos obtener el cliente con el token asociado. Si ocurre alguno de los errores especificados en el except, como ConnectionError, JSONDecodeError, o FileNotFoundError, #se manejan en el bloque except, y la función retorna None.

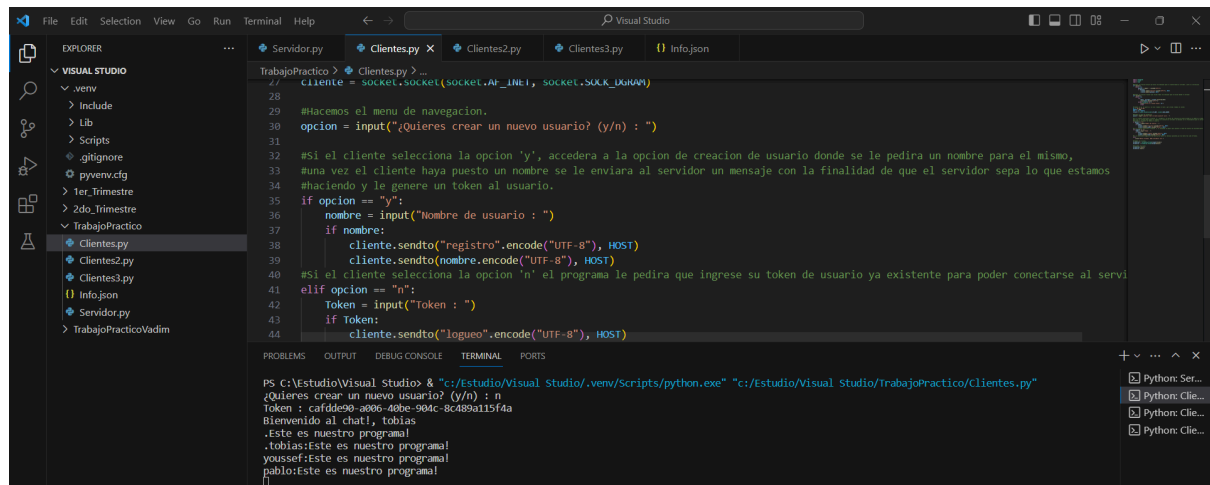
Por último, se verifica si el nickname obtenido no está presente en los valores del diccionario clientes. Si el nickname no está en clientes, significa que el cliente no está registrado, por lo que se agrega el nickname y el token asociado a los diccionarios clientes y tokens, respectivamente y se le envía al usuario un mensaje de bienvenida.

Debajo nos encontraremos con la función 'registro' esta espera un mensaje y una dirección. El mensaje viene desde la clase clientes, que primero envía un mensaje para que el servidor sepa a qué función deseamos acceder y luego el otro mensaje será el valor del diccionario token, mientras que la dirección será la clave del diccionario clientes. También esta función mediante una librería genera el token, enviando un mensaje al cliente informando de su token, si no existe el file lo crea automáticamente y registra el nuevo usuario.

La última función que nos encontraremos se llama 'mensaje' esta espera un mensaje y una dirección, el mensaje viene desde la clase clientes, que primero envía un mensaje para que el servidor sepa a qué función deseamos acceder y luego el otro mensaje sera el que mediante el bucle for, que recorre todo el diccionario de clientes, se envía a todos los clientes del servidor.

Luego se ve la creación de la ip y del puerto, la creación de los 2 diccionarios, la creación del socket y la asignación y un bucle while true que gestiona el funcionamiento de la clase en sí.

Imagen del correcto funcionamiento del servidor/chat,



The screenshot shows the Visual Studio IDE with a project named 'TrabajoPractico'. The Explorer pane on the left shows the file structure, including a 'TrabajoPractico' folder with files like 'Clientes.py', 'Clientes2.py', 'Clientes3.py', 'Info.json', 'Servidor.py', and 'TrabajoPracticoVadim'. The main editor displays the code for 'Clientes.py', which is a Python script for a chat client. The code includes comments in Spanish and implements a menu-driven interface for creating a new user or logging in. The code is as follows:

```
TrabajoPractico > Clientes.py > ...
27 cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
28
29 #Hacemos el menu de navegacion.
30 opcion = input("¿Quieres crear un nuevo usuario? (y/n) : ")
31
32 #Si el cliente selecciona la opcion 'y', accedera a la opcion de creacion de usuario donde se le pedira un nombre para el mismo,
33 #una vez el cliente haya puesto un nombre se le enviara al servidor un mensaje con la finalidad de que el servidor sepa lo que estamos
34 #haciendo y le genere un token al usuario.
35 if opcion == "y":
36     nombre = input("Nombre de usuario : ")
37     if nombre:
38         cliente.sendto("registro".encode("UTF-8"), HOST)
39         cliente.sendto(nombre.encode("UTF-8"), HOST)
40 #Si el cliente selecciona la opcion 'n' el programa le pedira que ingrese su token de usuario ya existente para poder conectarse al servi
41 elif opcion == "n":
42     Token = input("Token : ")
43     if Token:
44         cliente.sendto("logueo".encode("UTF-8"), HOST)
```

The TERMINAL pane at the bottom shows the command prompt output, indicating that the client is running successfully and interacting with the server. The output is as follows:

```
PS C:\Estudio\Visual Studio> & "c:/Estudio/Visual Studio/.venv/scripts/python.exe" "c:/Estudio/Visual Studio/TrabajoPractico/Clientes.py"
¿Quieres crear un nuevo usuario? (y/n) : n
Token : carfdes9 a206-40be-904c-8c489a115f4a
Bienvenido al chat!, tobias
.Este es nuestro programal
.tobias:Este es nuestro programal
.youssef:Este es nuestro programal
.pablo:Este es nuestro programal
```

**FUENTES:** Trabajo anterior de TCP, PDF de la unidad 3 ‘Sockets’, Información y videos de internet e inteligencia artificial.