```
CHAT {
  [MODELO
  CLIENTE-SERVIDOR]
     < Antonio González López >

}
```

Objetivos

Comunicación

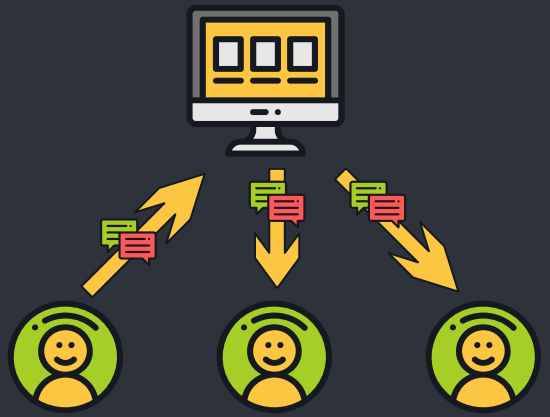
Chat funcional para dos o más usuarios

Privacidad

Encriptación simétrica en la comunicación

Aplicación distribuida

Usando un modelo cliente-servidor



Código aplicación servidor{

Creación de socket

```python
import socket
import threading as threading


host=socket.gethostbyname(socket.gethostname())         #IP PRIVADA DEL SERVIDOR
port=8000                                               #PUERTO DEL SERVIDOR DONDE SE ATIENDEN PETICIONES

socketserver=socket.socket(socket.AF_INET,socket.SOCK_STREAM)       #SE CREA EL SOCKET
socketserver.bind((host,port))                                      #SE VINCULA EL SOCKET A UNA IP Y UN PUERTO
socketserver.listen()                  #EL SERVIDOR EMPIEZA A ESCUCHAR DESDE EL PUERTO PARA ATENDER LAS CONEXIONES DEL CLIENTE

print(f"El servidor se encuentra en: {host}")      #SE DA IP DEL SERVIDOR

print("Esperando conexión con clientes...")

clientes=[]             #LISTA DE CLIENTES
```

}

Código aplicación servidor{

Broadcast

```
19  def broadcast(msj,remitente):           #FUNCIÓN QUE ENVÍA MENSAJE A TODOS LOS CLIENTES MENOS A REMITENTE
20      for client in clientes:
21          if client != remitente:         #SI EL CLIENTE NO ES EL REMITENTE
22              try:
23                  client.sendall(msj.encode("utf-8"))         #SE ENVÍA EL MENSAJE
24              except:
25                  clientes.remove(client)                     #SI HAY ALGÚN ERROR, SE ELIMINA AL CLIENTE DE LA LISTA DE CLIENTES
26                  print(f"Desconectando a {address} por un error en la conexión")
```

}

Código aplicación servidor{

Distribución de mensajes

```python
def recibiryenviar(client,address):     #FUNCIÓN QUE ESTABLECE CONEXIÓN CON UN CLIENTE Y SE ENCARGA DE DISTRIBUIR EL TRÁFICO
    clientes.append(client)             #SE AÑADE EL NUEVO CLIENTE A LA LISTA DE CLIENTES
    print(f"Conexión establecida desde {address}")

    while True:                         #BUCLE INFINITO
        try:
            msj=client.recv(1024).decode("utf-8")           #SE RECIBEN LOS MENSAJES ENCRIPTADOS DEL CLIENTE (MÁXIMO 1024 BYTES)
            if msj:                                         #SI HAY UN MENSAJE
                print(f"Mensaje recibido desde {address}: {msj}")       #SE IMPRIME EN PANTALLA EL MENSAJE ENCRIPTADO
                broadcast(msj,client)                       #EL MENSAJE RECIBIDO POR EL SERVIDOR SE ENVÍA A TODOS LOS CLIENTES
        except:
            broadcast(encriptar(f"{address} se ha desconectado",10), client)
            print(f"Conexión cerrada desde {address}")
            clientes.remove(client)         #EN CASO DE ERROR SE AVISA DE LA DESCONEXIÓN Y SE ELIMINA AL CLIENTE DE LA LISTA
            break                           #SE ACABA EL BUCLE
```

}

Código aplicación servidor{

Creación de hilo

```
57  while True:                 #POR SIEMPRE
58      client,address=socketserver.accept()            #ACEPTAR LAS CONEXIONES QUE LLEGAN AL SOCKET
59      hilo = threading.Thread(target=recibiryenviar,args=(client,address))        #SE CREA UN HILO POR CADA CLIENTE QUE EJECUTA
60                                                                                  #LA FUNCIÓN RECIBIRYENVIAR()
61      hilo.start()            #SE EJECUTA EL HILO
```

}

Código aplicación cliente{

Creación de socket

```python
import socket
import threading

ipserver=input("Escribe la IP del servidor: ")
port=8000                    #PUERTO DEL SERVIDOR QUE ATENDERÁ PETICIONES


#SE CREA CONEXIÓN TCP/IP ENTRE CLIENTE Y SERVIDOR
client=socket.socket(socket.AF_INET,socket.SOCK_STREAM)

client.connect((ipserver,port))


nombre=input("Escribe tu nombre de usuario: ")      #SE PREGUNTA A USUARIO SU NOMBRE
```

}

Cliente.py

Servidor.py

Código aplicación cliente{

Cifrado César para mensajes

```python
def encriptar(mensaje, desplazamiento):          #ENCRIPTACIÓN CIFRADO CÉSAR
    mensaje_encriptado = ""
    for caracter in mensaje:
        if str(caracter).isalpha():
            ascii_inicial = ord('a') if str(caracter).islower() else ord('A')
            ascii_encriptado = (ord(str(caracter)) - ascii_inicial + int(desplazamiento)) % 26 + ascii_inicial      #FÓRMULA ENCRIPTADO
            caracter_encriptado = chr(ascii_encriptado)
            mensaje_encriptado += caracter_encriptado
        else:
            mensaje_encriptado += str(caracter)
    return mensaje_encriptado
```

}

Código aplicación cliente{

Desencriptación

```python
def decrypt(mensaje_encriptado, desplazamiento):    #DESENCRIPTACIÓN CIFRADO CÉSAR
    mensaje_desencriptado = ""
    for caracter in mensaje_encriptado:
        if caracter.isalpha():
            ascii_inicial = ord('a') if caracter.islower() else ord('A')
            ascii_desencriptado = (ord(caracter) - ascii_inicial - desplazamiento) % 26 + ascii_inicial
            caracter_desencriptado = chr(ascii_desencriptado)
            mensaje_desencriptado += caracter_desencriptado
        else:
            mensaje_desencriptado += caracter
    return mensaje_desencriptado
```

}

Código aplicación cliente{

Recepción de mensajes

```
42  def recibir():          #FUNCIÓN PARA RECIBIR MENSAJES DESDE EL SERVIDOR
43      while True:
44          try:
45              msj=client.recv(1024).decode('utf-8')       #SE RECIBEN MENSAJES DE MÁXIMO 1024 BYTES Y SE DECODIFICAN USANDO UTF-8
46              print(decrypt(msj,10))                      #SE DESENCRIPTA CON CIFRADO CÉSAR
47
48          except:                                         #EN CASO DE QUE HAYA ALGÚN ERROR...
49              print("Desconectando del servidor...")
50              client.close()
51              break
52
53
54  hilorecibir = threading.Thread(target=recibir)          #UN HILO SE EJECUTA SIEMPRE Y SE ENCARGA DE RECIBIR MENSAJES
55  hilorecibir.start()                                     #SE INICIA EL HILO
```

}

Código aplicación cliente{

Envío de mensajes

```
58  while True:                         #CLIENTE ESCUCHANDO SIEMPRE A USUARIO PARA MANDAR MENSAJES AL SERVIDOR
59      msj=input('>')                  #MENSAJE QUE ESCRIBE EL USUARIO
60      if msj:                         #EN CASO DE QUE EL USUARIO HAYA ESCRITO UN MENSAJE
61          if msj.upper()=="FIN":      #SI USUARIO ESCRIBE FIN --> SE TERMINA LA CONEXIÓN
62              client.close()
63              break
64
65          else:
66              msjsinencriptar=f"{nombre}: {msj}"      #SE CONCATENA EL NOMBRE DE USUARIO CON EL MENSAJE
67              msjencriptado=encriptar(msjsinencriptar,10).encode("utf-8")     #SE ENCRIPTA EL MENSAJE USANDO CIFRADO CÉSAR CON DESPLAZAMIENTO
68          #    print(msjencriptado)        #COMPROBACIÓN DE QUE SÍ ENCRIPTA
69              client.sendall(msjencriptado)           #SE ENVÍA EL MENSAJE ENCRIPTADO AL SERVIDOR
```

}

```
1
2  Resultados obtenidos{
3
4
5
6          [demo]
7
8
9
10
11
12         }
13
14
```

Problemas

1. Librerías y documentación

2. Puertos del servidor

3. Cifrado de mensajes

4. Transmisión de mensajes como bytes

5. Firewall

Futuras mejoras

1. GUI (librería tkinter)

2. IP pública y permanente del servidor

3. Envío de archivos e imágenes

4. Encriptación más segura (AES)

5. Salas de chat y mensajes privados

6. Base de datos con usuarios y contraseñas