**Comunicación entre computadores utilizando socket.**

Este código es un servidor de chat en Python usando la interfaz de usuario Tkinter. Establece una conexión con uno o más clientes utilizando el protocolo TCP. Recibe conexiones de clientes, envía mensajes a uno o más usuarios a elección, es capaz de recibir mensajes de clientes y sirve como puente para que los usuarios se comuniquen entre sí. Cuando un usuario se conecta, desconecta, o se encuentra inactivo, avisa a todos los clientes para que sus interfaces de usuario se actualicen de forma automática.

Aclaración: es importante desactivar el firewall y conocer la ip del servidor para poder conectarlo

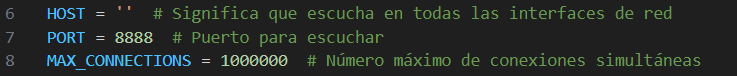
Desarrollo

1. Agregue las siguientes bibliotecas:

Texto

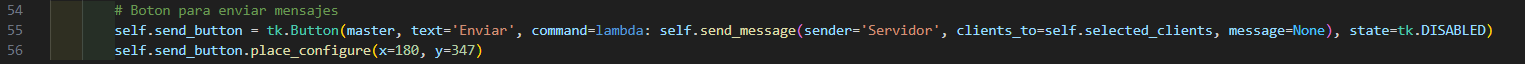
Descripción generada automáticamente

1. Crearemos las variables necesarias para poder conectarnos como servidor.



1. \_\_init\_\_: Esta es la función de inicialización de la clase. Establece los componentes de la interfaz de usuario.Imagen que contiene Interfaz de usuario gráfica

   Descripción generada automáticamente
   1. Construcción de la Interfaz gráfica y las variables a usar: Se construyen los botones de la interfaz grafica junto a todas las funciones que se activarán con cuando se presione junto a estas. Las variables por utilizar son para almacenar distintos valores que serán de utilidad para el funcionamiento correcto del código.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

* 1. Texto

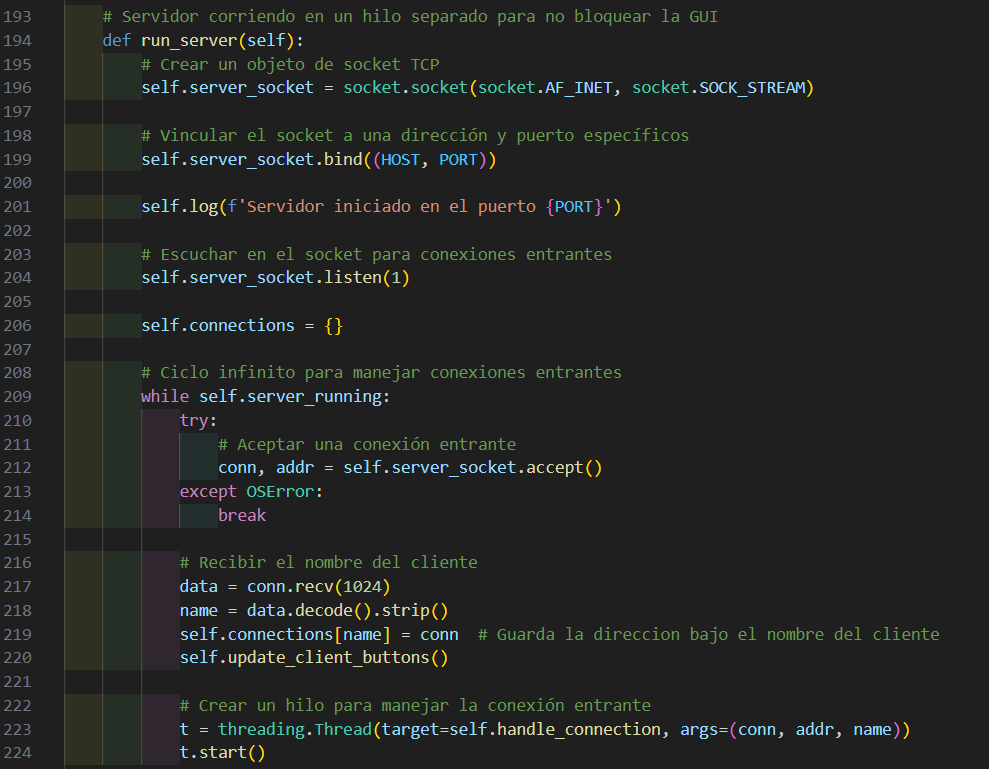
     Descripción generada automáticamenteTexto

     Descripción generada automáticamenteFunciones Utilitarias: Pueden ser citadas en múltiples zonas del código o servir para una cosa, por ese motivo se decidió dejarles su propia zona.

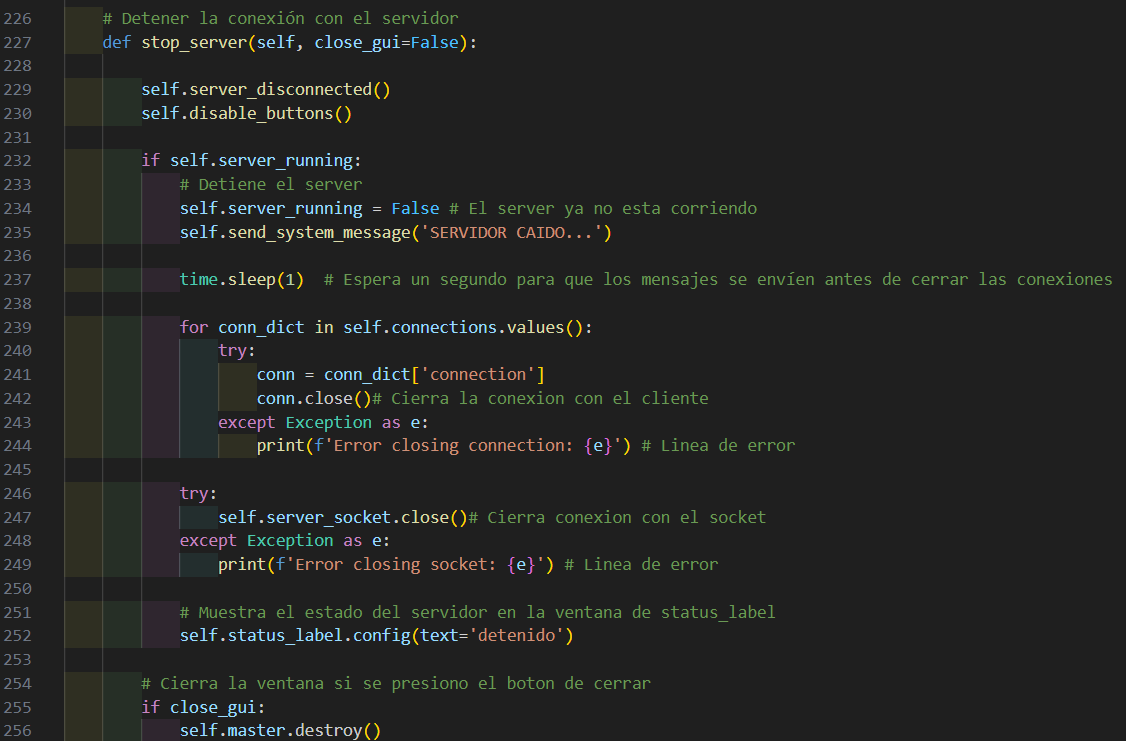
4. Start Server: Esta función habilita y deshabilita botones de la interfaz gráfica, según corresponda e inicia un hilo para saber que el servidor está funcionando, además de iniciar el servido.

Texto

Descripción generada automáticamente

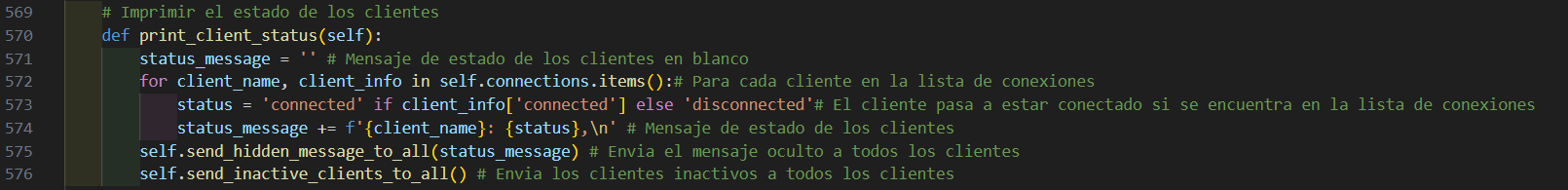
5. run\_server: Esta función inicia el servidor, establece un puerto para escuchar nuevas conexiones, los maneja en hilos diferentes y actualiza la interfaz grafica.

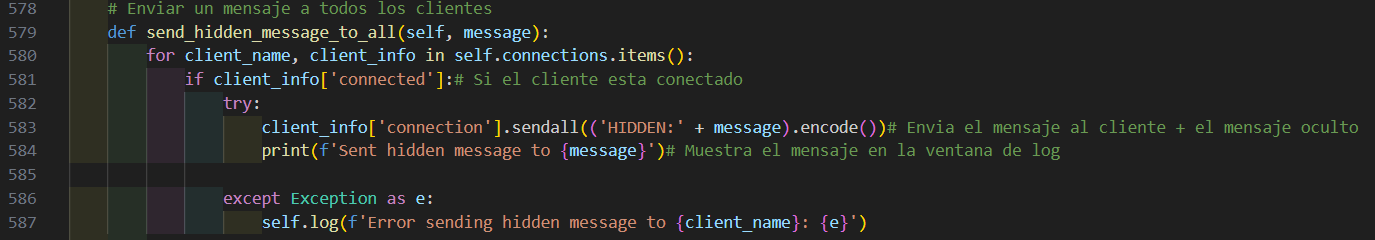
6. stop\_server: detiene el servidor, cierra todas las conexiones activas y el socket del servidor, actualiza la GUI para reflejar el estado detenido y, opcionalmente, cierra la GUI del servidor. Esto es debido a que tanto el botón de cerrar la GUI como al botón de cerrar conexión del servidor ejecutan esta función, por lo que se debe tener en cuenta ambas opciones, sino causaría error. El tiempo de espera es porque el cliente tiene una función que envía mensajes, por lo que puede haber un error si no se tiene el tiempo de espera.



7. handle\_connection: Esta función es activada cada vez que un nuevo cliente es conectado o desconectado. Activa las siguientes funciones en órden.Texto

Descripción generada automáticamente

7.1. Print\_client\_status: Crea un estado de cliente, que guarda como un mensaje y luego ejecuta dos 2 funciones.

7.1.2. send\_hidden\_message\_to\_all: Envia un mensaje a todos los clientes conectados, anteponiendo la palabra ‘HIDDEN:’.

Texto

Descripción generada automáticamente7.1.3. send\_inactive\_clients\_to\_all: toma la lista de clientes inactivos y lo envía a todos los clientes que están conectados.

Texto

Descripción generada automáticamente7.2 initialize\_connection: Envía un mensaje del sistema y un mensaje de registro que indica que el cliente se ha conectado, habilita el botón global, actualiza los botones del cliente y cambia el color de los botones del cliente según si los clientes están activos o inactivos.

Texto

Descripción generada automáticamente7.3. process\_message: Procesa mensajes de un cliente, maneja desconexiones (i el mensaje es ‘DISCONNECT’) y advertencias (si empiezan con ‘ADVERTENCIA: ’). De otra forma, si no es un mensaje vacío, lo separa en las partes necesarias para su procesamiento, maneja clientes reactivados (inactivos que enviaron mensajes nuevamente) y maneja excepciones.

Imagen que contiene Texto

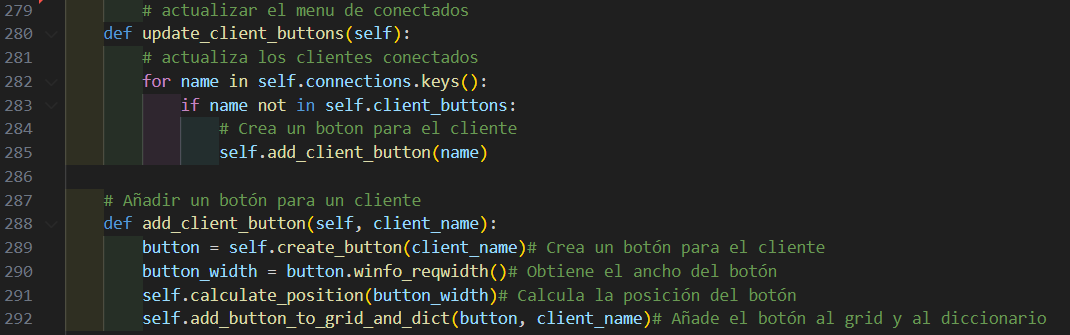
Descripción generada automáticamente7.4 send\_system\_messaage: Envía un mensaje del sistema a todos los clientes conectados, excluyendo a un cliente específico (si se especifica), y maneja los clientes que se desconectan abruptamente.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente7.5 terminate\_connection: Cierra la conexión con un cliente, elimina el cliente del diccionario de conexiones, actualiza los botones del cliente en la GUI del servidor, registra un mensaje que indica que la conexión se ha cerrado e imprime el estado de todos los clientes.

8. Botones: La GUI tendrá una serie de botones de color verde, amarillo y rojo. Estos representan a clientes conectados, inactivos (pero conectados) y desconectados.Captura de pantalla de computadora

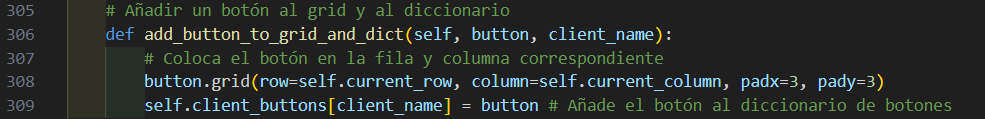
Descripción generada automáticamente

 8.1. update\_client\_buttons: actualiza los botones de los clientes conectados, si estos no existen, entonces los crea. Para crearlos llama la función create\_button, luego se calcula su possición y finalmente se añade al grid.

Texto

Descripción generada automáticamente8.1.1. calculate\_position: Calcula la posición de un nuevo botón en un grid, creando una nueva fila si es necesario.

8.1.2. add\_button\_to\_grid\_and\_dict: agrega un botón en el grid y en el diccionario client\_buttons.



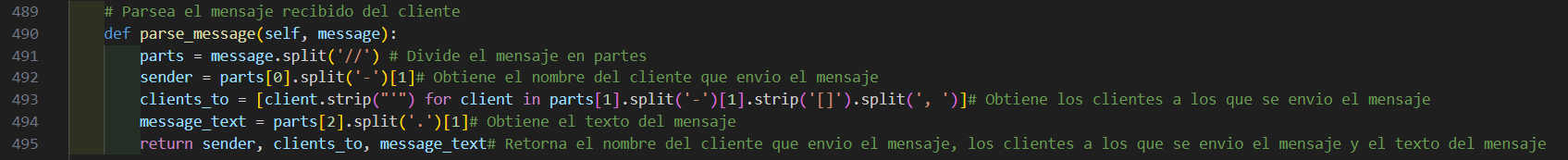
Texto

Descripción generada automáticamente8.2. Remove\_online: Cuando un cliente se encuentre desconectado (botón rojo), esto lo eliminará de la GUI.

Texto

Descripción generada automáticamente9. Receive\_messages: Esta función intenta recibir un mensaje de los clientes, en caso de no recibir datos de este, entonces retorna “DISCONNECT”.

10. parse\_message: Hay que tener en cuenta como el servidor recibe un mensaje. Digamos que el cliente “Pato” quiere enviarle el mensaje “Hola” al cliente “Ganso”. El servidor recibiría este mensaje: ‘SENDER-Pato//CLIENTS\_TO-Ganso//MESSAGE.Hola’.

 Por esto lo separa en 3 partes, teniendo en cuenta los “//” como separadores y ocupa cada parte para identificar quien lo envio, para quien va dirigido y cual es el mensaje que se enviará.

Texto

Descripción generada automáticamente11. handle\_parsed\_messages: En caso de iniciar con la palabra “ADVERTENCIA” se trata de forma separada, separándolos en partes para identificar el cliente que envió la advertencia y se trata aparte, después se retornan los resultados y se deja ahí. Si no empezó con “ADVERTENCIA” se hace una copia de los clientes activos, si “Servidor” estaba en la copia, se remueve. En caso de que “Global” estén los destinatarios o esté vacío, se enviará un mensaje global. Si no, entonces es un mensaje privado a todos los que fueron especificados, concatenándolos con una “y” de la lista original, pero si se envía al cliente “Servidor”, entonces es otra cadena de texto. Finalmente, al final de esto los clientes seleccionados son los mismo que la lista copiada, de otra forma el servidor tendría seleccionado a “Servidor” como cliente destinatario, y el cliente “Servidor” es el propio Server, por eso fue necesario removerlo y hacer una copia en los pasos anteriores.

12. handle\_inactive\_client: En caso de que el cliente se encuentre inactivo, y no lo estaba antes, se añade a la lista de inactivos, se actualiza su botón y aparece la advertencia en la GUI. De otra forma, recuerda que el cliente está inactivo. De cualquier forma, envía un mensaje por sistema que el cliente en cuestión se encuentra inactivo.Texto

Descripción generada automáticamente

13. Send\_global\_message: A todos los clientes conectados les envía el mensaje en cuestión.

Texto

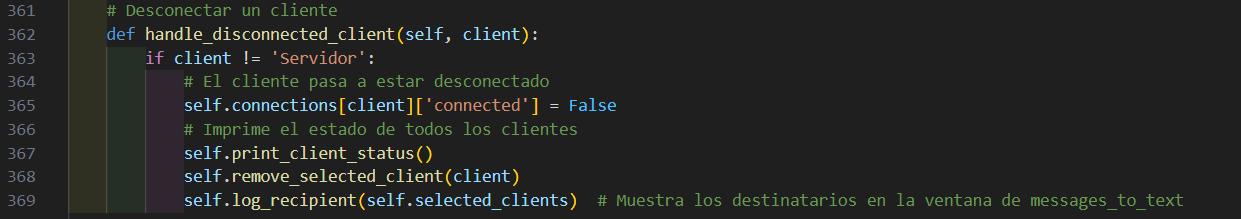
Descripción generada automáticamente

14. Send\_private\_message: Se envía el mensaje privado a quienes corresponda, en caso de que el destinatario esté desconectado se maneja aparte. Una vez eso acaba, se le envia una respuesta a quien envio el mensaje.Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

14.1. send\_message\_to\_client: Es el envío de mensajes, se manejó aparte para hacer mas cómodo el manejo de las excepciones, y en caso de haberlo, este se remueve de los clientes seleccionadosPantalla de computadora

Descripción generada automáticamente con confianza media.

14.2: handle\_disconnected\_client: Si el cliente no es el servidor (para evitar errores), el cliente pasa a estar desconectado, los clientes actualizan sus status, se remueve dee los seleccionados y se actualiza la selección.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media14.3: send\_response\_to\_sender: todos los seleccionados se vuelven una lista separadas por comas, se prepara la respuesta y se envia según corresponde. En caso de que el que lo envio fue el Servidor, este se trata aparte.

Texto

Descripción generada automáticamente15. Select\_client y remove\_selected\_client: Esta función añade o remueve los clientes seleccionados presionando en su respectivo botón para enviarles un mensaje, teniendo en cuenta todos las posibles combinaciones.

Texto

Descripción generada automáticamente16. send\_message: Esto se activa cuando se presiona el botón “Enviar” en la GUI. Si no hay un mensaje, lo extrae de la caja donde uno lo escribe. Si está vacío, retorna y se acaba la función. Si no lo está, prosigue y verifica a quien está dirigido. Si es global envía un mensaje global, si es un mensaje privado, entonces envía un mensaje privado.

17. Este último fragmento de código crea una instancia de la clase ServerGUI y luego ejecuta el bucle principal de Tkinter, que es responsable de manejar todos los eventos de la interfaz de usuario.

