**Comunicación entre computadores utilizando socket.**

Este código es un cliente de chat en Python usando la interfaz de usuario Tkinter. Establece una conexión con un servidor utilizando el protocolo TCP, envía el nombre de usuario al servidor y luego inicia un hilo de recepción de mensajes para recibir los mensajes del servidor. El cliente también incluye una interfaz de usuario para que el usuario ingrese un mensaje para enviar al servidor y elija a quien o quienes puede enviar dicho mensaje.

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.

Un reloj digital en la pantalla

Descripción generada automáticamente con confianza media

1. \_\_init\_\_: Esta es la función de inicialización de la clase. Establece los componentes de la interfaz de usuario, establece el nombre del usuario y luego llama a la función

Texto

Descripción generada automáticamente

* 1. Texto

     Descripción generada automáticamenteConstrucción 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 automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamente3.2: Funciones Utilitarias:

Texto, Chat o mensaje de texto

Descripción generada automáticamente

4.connect\_to\_server: Esta función intenta establecer una conexión con el servidor. Si se conecta con éxito, enviará el nombre del usuario al servidor y luego iniciará el hilo de recepción de mensajes, inicia también un temporizador (10 segundos por en este caso) y actualiza botones. En caso de conectarse sin éxito lo volverá a intentar otra vez, hasta un máximo de 3 ocasiones.

Texto

Descripción generada automáticamente

5. send\_warning: Envía un mensaje de advertencia al servidor avisando que está inactivo ejecuta la función reset\_timer. Texto

Descripción generada automáticamente

6. reset\_timer: cuando es llamada esta función, cancela el tiempo actual del temporizador y lo sobrescribe para ser 10 segundo. Luego, lo inicia nuevamente.

Texto

Descripción generada automáticamente

7. funciones de cerrado: estas 3 funciones ejecutan la desconeccion del servidor, pero en distintas ocasiones: handle\_server\_disconnection es llamada cuando no se reciben datos del servidor, on\_close se ejecuta cuando se cierra la GUI y server\_broken se ejecuta cuando el servidor ha sido desconectado. En el caso del on\_close se espera 1 segundo para asegurarse que todo el proceso de desconección pasará sin provocar un error. Texto

Descripción generada automáticamente

8. Disconnect\_from\_server: en caso de haber estado inactivo se quita de la lista de inactivos, entonces intenta desconectarse del servidor enviando “DISCONNECT”. Maneja errores en caso de haberlos, y finalmente actualiza la GUITexto

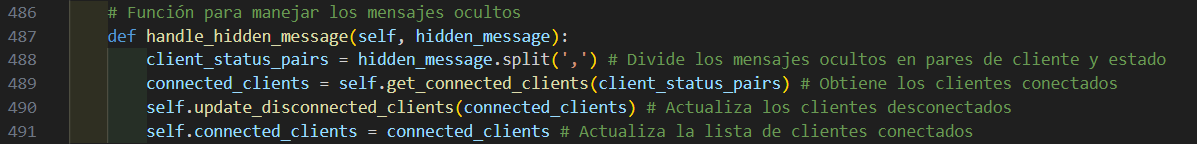
Descripción generada automáticamente

9.Receive\_messges: es una función que escucha los mensajes recibidos del servidor y ejecuta la función handle\_received\_data para procesar los mensajes según corresponda.Texto

Descripción generada automáticamente

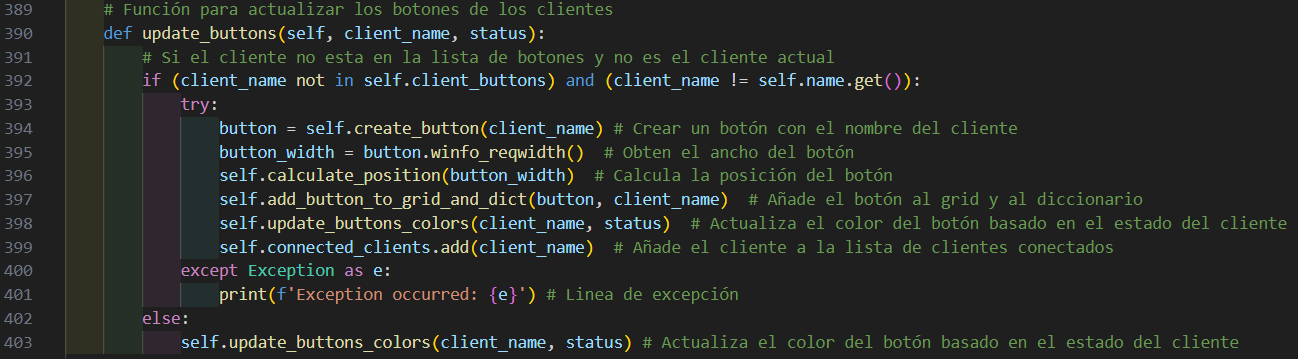
Texto

Descripción generada automáticamente con confianza media9.1. Procesa la data recibida según corresponda, mandando a cerrar el servidor y ejecutar otras funciones dependiendo de como fue recibido el mensaje.

10. Handle\_hidden\_message: es procesado sin la palabra “HIDDEN:”, se obtiene su status, separandolos por “,”, luego se ejecuta la función get\_connected\_clients y después se ejecuta la función update\_disconnected\_clients. Finalmente se actualiza la lista de clientes que están conectados.

10.1. get\_connected\_clients: connected\_clients es iniciado y en el ciclo for se obtienen todos los clientes junto a su estado actual, luego se actualizan los botones y finalmente se añade al conjutno de clientes conectadosTexto

Descripción generada automáticamente

10.1.1. update\_buttons: Primero, se asegura que el cliente no tenga un boton ya creado ni sea él mismo (para no crear un botón con su propio nombre), los que ingresan en esta parte tienen un status de “connected”. En caso de no cumplir dichas condiciones, ejecutará la función update\_buttons\_colors, siendo el estado de quienes no cumplen esas condiciones “disconnected” por defecto, ya que así son enviados desde el servidor.

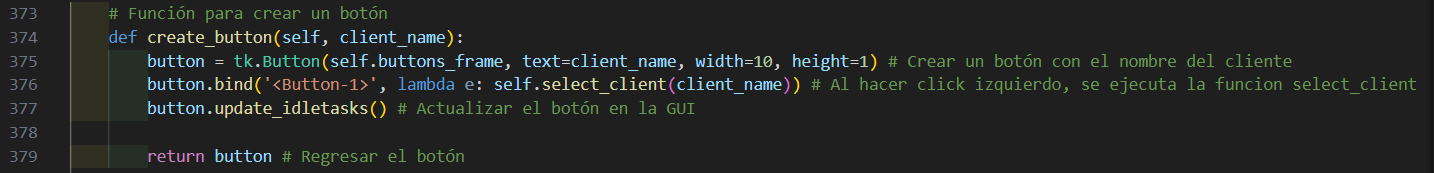
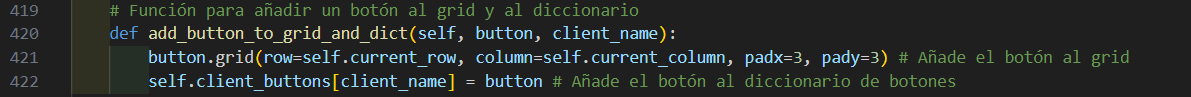
10.1.1.1. create\_button: Esta función crea un botón sin color, el cual tiene la función “select\_client” ligada a este. Luego, retorna el button para usarse en otra parte de la función.

Imagen que contiene Interfaz de usuario gráfica

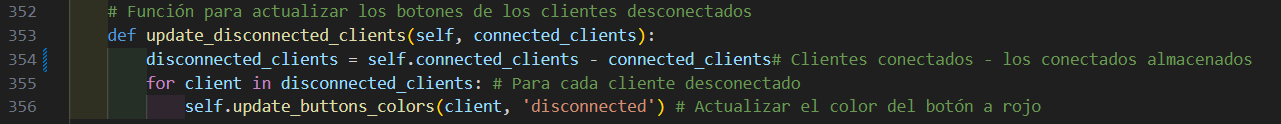
Descripción generada automáticamente10.1.1.2. Calculate\_ position: calcula la posicion del boton, empezando del segundo valor del grid en la primera fila, ya que un botón es creado por defecto en la GUI: el botón “Servidor”.

10.1.1.3: añade el botón en el grid con sus medidas correspondientes y el nombre de su respectivo cliente.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media10.1.2. update\_buttons\_colors: esta función actualiza el color de los botones ya creados de acuerdo a los estados que estos tengan.

10.2. update\_disconnected\_clients: lo anterior solo contempla las nuevas conecciones, para actualizar las ya existentes usamos esta función. Sabe cuales son los conetados, ya que compara la lista recibida con la actual, generando una nueva y actualizarlo con la función update\_button\_colors.



11. handle\_inactive\_clients\_from\_server: procesa los datos recibidos del servidor, ya se le ha quitado la parte de “INACTIVES:”. Separa los datos usando “,” e ignora los vacíos, de haberlos. Para cada cliente en la data se ejecuta la función verify\_inctive\_clients.Texto

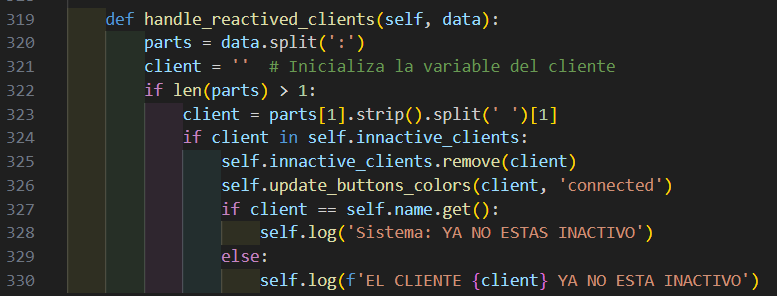
Descripción generada automáticamente

11.1. verify\_inactive\_clients: Si el cliente en cuestión no estaba inactivo, se añade a la lista como uno, después se cambia su colorTexto

Descripción generada automáticamente usando la función update\_buttons\_colors.

12. handle\_inactive\_clients: Esta función es ejecutada cuando se recibe un mensaje que empieza con “Sistema: ADVERTENCIA:”. Lo separa en partes usando “:” como medio de separación, y como habrá más partes, se toma la tercera y se ignoran los espacios en blanco (el servidor envía algo similar a esto: “Sistema: ADVERTENCIA: Pato”, por eso la tercera parte siempre es el cliente). Entonces considera varias situaciones, primero: si el que está inactivo es el propio usuario, primero verifica si ya se clasifica como inactivo, de estarlo avisa que sigue inactivo, si no lo estaba antes da el primer aviso. En caso de ser otro el cliente avisa cual se encuentra inactivo, finalmente, después de todo eso, se ejecuta la función verify\_inactive\_clients.Texto

Descripción generada automáticamente

13. handle\_reactived\_clients: Esta función es ejecutada cuando se recibe un mensaje que empieza con “Sistema: Cliente”. Lo divide en partes y compara si el cliente en cuestión ya estaba desconectado, de estarlo, lo elimina de esa lista, después actualiza su botón y finalmente verifica si dicho cliente era él mismo, de serlo imprime en la GUI que ha sido reactivado, si era otro entonces da aviso que dicho cliente ya no se encuentra inactivo. 

14. Select\_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 automáticamente

15. Esta función se ejecuta cuando es presionado el botón “Enviar” en la GUI, al hacerlo, extrae el mensaje de la casilla message\_text. Reinicia el temporizador, si se está conectado envía el mensaje con su formato respectivo y, en caso de haber algún error, este aparece en pantalla.Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente