“Año del Fortalecimiento de la Soberanía Nacional”

UPC – Universidad Peruana de Ciencias Aplicadas

Proyecto Final -Parte 1 Chat Serial

Proyecto Final -Parte 2 Chat TCP/IP

Curso:

Programación Avanzada de Computadoras

Alumno:

Carbajal Jordán,Carlos Christopher

Código:

U201712774

Profesor:

Luis Antonio Muñoz Alfaro

Sección:

EL184-LS6B

Fecha:

30/11/2022

Ciclo: 2022-02

Proyecto Final -Parte 1 Chat Serial

Comunicación de datos

La comunicación de datos es la transmisión de mensajes digitales a dispositivos externos. Es decir, el intercambio de comunicación reside en enviarse bytes de un dispositivo a otro. El proceso de transmitir un mensaje ocurre millones de veces al día y sin que ninguno de nosotros sea consciente de ello. (Techlandia,2018)

Periféricos de comunicación

Los periféricos son dispositivos que no forman parte de la estructura central de un ordenador y además, no ocupan espacio en su memoria. Hay una gran variedad y tienen la función de aumentar sus prestaciones mejorando la experiencia del usuario. Asimismo, estos dispositivos sirven para establecer una transmisión de datos a distancia entre un computador y otro, o entre un computador y otro periférico remoto. Dicha comunicación puede darse de manera alámbrica o inalámbrica. (Enciclopedia de Ejemplos, 2015)

Los dispositivos o periféricos de entrada son aquellos aparatos que sirven para ingresaro proporcionar datos y señales de control a la unidad central de procesamiento de un sistema.

Los dispositivos de salida son aquellos aparatos que forman parte del hardware y que transmiten la información proveniente de una computadora hacia al usuario u otra computadora o red.

¿Cuáles son los dispositivos periféricos de comunicación?

Según (PCAcademia,2021) entre los más importantes serian lo siguientes:

- **Enrutadores de Red**

  Este dispositivo tiene la capacidad de decodificar toda la información de la red de internet y transformarla en datos, para que sean utilizados por los diferentes equipos tecnológicos. Este tipo de equipos puede ser utilizado de forma alámbrica o inalámbrica y es el tipo de conexión más utilizada entre los usuarios, por su alta eficiencia de comunicación.

- **Módems**

  Suelen ser equipos que reciben, registran y codifican la señal de un proveedor de internet a través de la conexión de un cable o de una fibra óptica. Tiene como objetivo traducir las señales que se emiten entre el equipo que emite la señal, y losque envían y reciben la información.

- **Repetidor de Señal**

  Se utilizan cuando necesitan mejorar las condiciones de la conexión de internet. Se utilizan cuando se tienen conexiones inalámbricas y estas por naturaleza, tienen un alcance un poco limitado. Son ideales para ampliar y mejorar de manera considerable la señal. No alteran ninguna condición de la red de Wi-Fi.

- **Conmutador de Red**

  Permite realizar la conexión entre dispositivos a través de una red que se le conoce como “Sistema de estrella”. Se dice que el Conmutador es el área central y representativa de toda la conexión entre los diferentes equipos, u ordenadores que conformarán la red.

- **Tarjetas de Red (NIC)**

  Son periféricos de comunicación que se pueden o no integrar a la placa base del ordenador. Intercambian la información entre los dispositivos que se encuentran conectados.

- **Concentrador**

  Identificado con el nombre de “Hubs”, este es un dispositivo que tiene como función ampliar la conexión en una red que tenga sistema de cableado. Son muy utilizadosen las estructuras de un servidor web.

- **Dispositivos Wi-Fi**

  En la actualidad este es un periférico de comunicación muy utilizado por su potenciaal establecer las conexiones de red. Además, su sistema de comunicación puede ser alámbrica o inalámbrica, según las preferencias del usuario.

Protocolos

Los protocolos de comunicación son un conjunto de reglas y pautas que permiten a los usuarios de la informática enviar un bloque de datos de una ubicación a otra. Se utilizan fundamentalmente para determinar el formato y la transmisión correcta de los datos, tanto en las comunicaciones analógicas como digitales. (lifeder,2020)

Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación. (openwebinars,2022)

Fundamentos teóricos Parte 1 Chat Serial:

Transmisión de datos Serial

La comunicación en serie es un método comúnmente utilizado para intercambiar datos entre ordenadores y dispositivos periféricos. La transmisión serie entre el emisor y el receptor está sujeta a protocolos estrictos que proporcionan seguridad y fiabilidad y han llevado a su longevidad. Muchos dispositivos, desde ordenadores personales hasta dispositivos móviles, utilizan la comunicación en serie. Echemos un vistazo más de cerca a sus fundamentos. El baud rate indica el número de bits por segundo que se transfieren, y se mide en baudios. Las velocidades de transmisión más comunes son de 115200, 9600, y 4800.(Serialportmonitor,2020)

Los bits de datos se refieren a la cantidad de bits (palabra) en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits.

- COM

  La mayoría de las computadoras incluyen puertos seriales. Actualmente puertos USB, aunque aún se encuentran algunas con puerto serial RS-232.

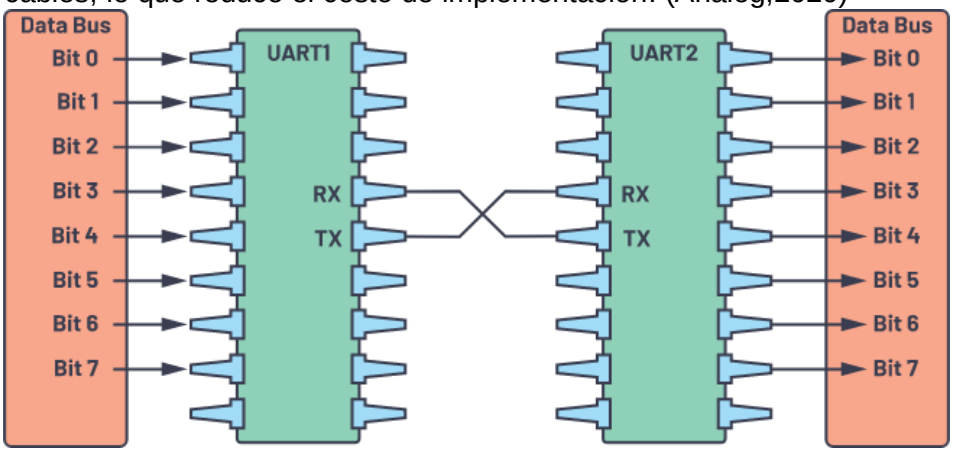
  RS-232 (Estándar ANSI/EIA-232) es el conector serial hallado en las PCs IBM y

compatibles. Está limitado a comunicaciones de punto a punto entre los dispositivosy el puerto serial de la computadora. (Instituto Tecnológico de Querétaro, 2021)

RS-485 (Estándar EIA-485) es una mejora sobre RS-422, ya que incrementa el número de dispositivos que se pueden conectar (de 10 a 32) y define las características necesarias para asegurar los valores adecuados de voltaje cuando se tiene la carga máxima. La capacidad de tener una inmunidad al ruido, hacen queeste tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan dispositivos distribuidos en red conectados a una PC u otro controlador para la colección de datos, HMI, u otras operaciones. (Instituto Tecnológico de Querétaro, 2021)

- UART

UART, o receptor-transmisor asíncrono universal, es uno de los protocolos de comunicación dispositivo a dispositivo más utilizados. Este artículo muestra cómo usar UART como un protocolo de comunicación de hardware siguiendo el procedimiento estándar.
Cuando se configura correctamente, UART puede funcionar con muchos tipos diferentes de protocolos en serie que implican la transmisión y recepción de datos en serie. En la comunicación en serie, los datos se transfieren bit a bit utilizando una sola línea o cable. En la comunicación bidireccional, usamos dos cables para una transferencia de datos en serie exitosa. Dependiendo de la aplicación y los requisitos del sistema, las comunicaciones en serie necesitan menos circuitos y cables, lo que reduce el costo de implementación. (Analog,2020)



Envío y Recepción de datos

La transmisión de datos es la transferencia física de datos, o sea, un flujo digital de bits, por un canal de comunicación punto a punto o punto a multipunto. Ejemplos de estos canales son cables de par trenzado, fibra óptica, los canales de comunicación inalámbrica y medios de almacenamiento. El envío y recepción de datos es solamente la programación de cómo se va a transmitir la información y a través de que puertos por ejemplo el USB. (Prudente, Jorge, 2019)

Threading

Este módulo construye interfaces de hilado de alto nivel sobre el módulo de más bajo nivel. En CPython, debido al Candado de intérprete global, solo un hilo puede ejecutar código Python a la vez (aunque ciertas bibliotecas orientadas al rendimiento pueden superar esta limitación). (Python, 2022)

ScrolledText

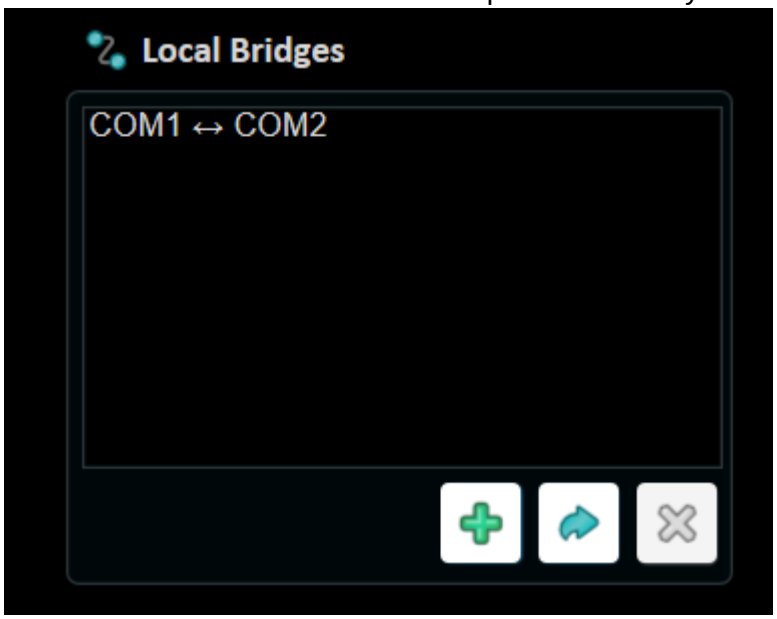
El módulo tkinter.scrolledtext proporciona una clase del mismo nombre que implementa un widget de texto básico que tiene una barra de desplazamiento vertical configurada para hacer «lo correcto». Usar la clase ScrolledText es mucho más fácil que configurar un widget de texto y una barra de desplazamiento directamente. (Python, 2022)

Requerimiento del TF1-Parte1:

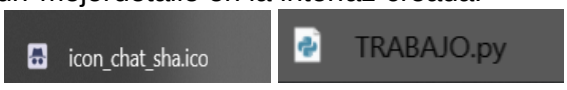
Realizar un programa en tkinter de Chat Serial que permita establecer comunicación entre dos equipos utilizando un NULL MODEM, o con dos aplicaciones ejecutadas de forma simultánea y conectadas a un Serial Bridge habilitado con el emulador de puerto serial Virtual Serial Ports.

Explicación del funcionamiento del programa realizado TF-Parte1:

1. Establecer la comunicación de puertos a través del programa HHD Virtual Serial Port. En este caso se establece los puertos COM1 y COM2.

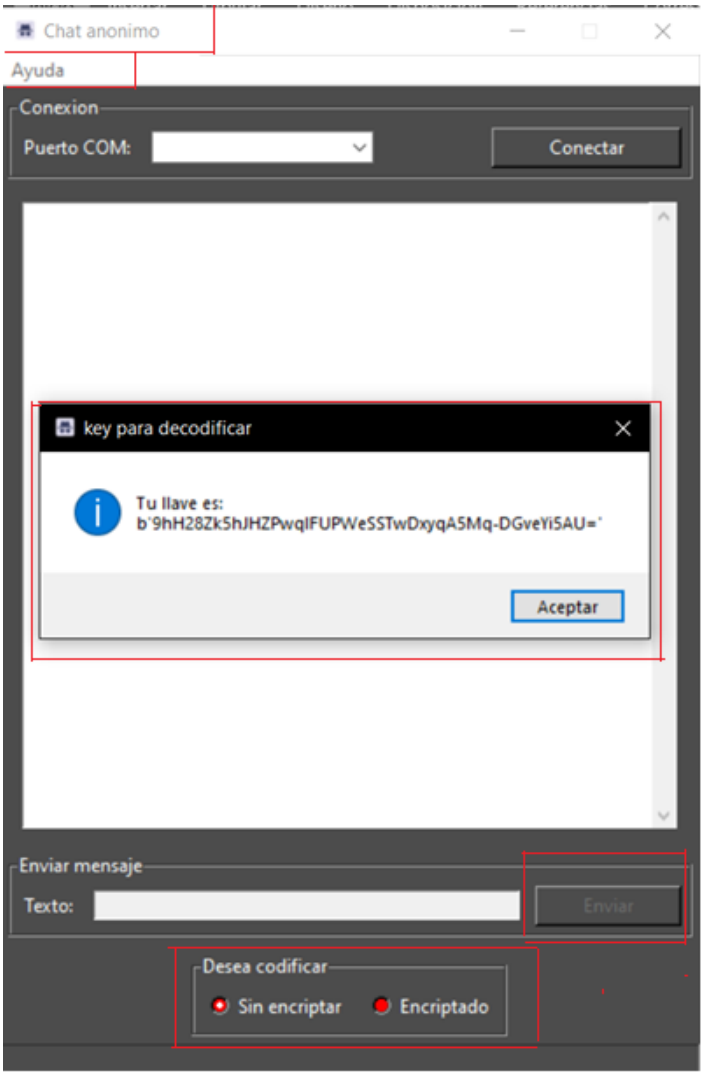


2. Carpeta donde se encuentra el archivo .py y el icono utilizado para un mejordetalle en la interfaz creada.

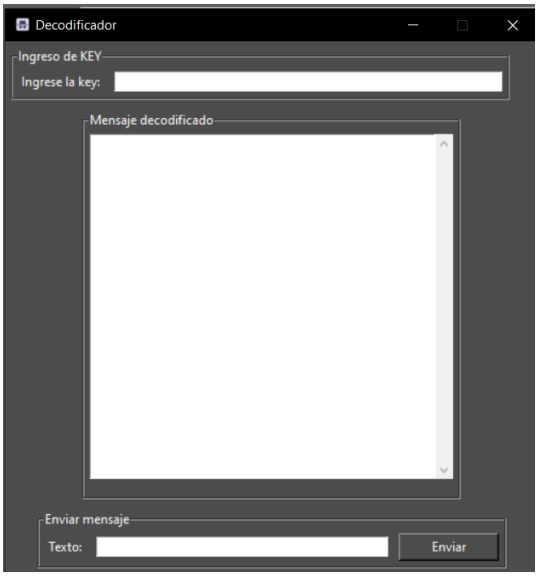


3. Ejecutar el programa .py a través de la consola. En este caso se ejecuta 2 veces para tener 2 chats seriales y demostrar la simulación.

4. Al iniciar se puede observar un icono, en este caso de Incognito. Asimismo, una ventana de visualización de la clave para decodificar las palabras que valla a codificar a través de la aplicación. Además, se observauna ventana de ayuda. Adicionalmente, se observa que desde un principio el Entry del texto está deshabilitado, al igual que el botón enviar.

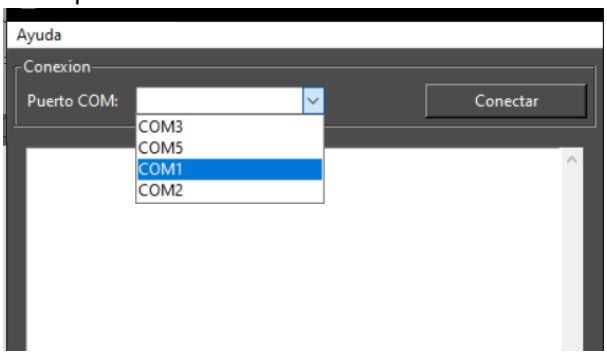


5. Además, en el Menú Ayuda podemos visualizar tiene opciones para cambiar de nombre y la opción de Decodificar para traducir la información codificada

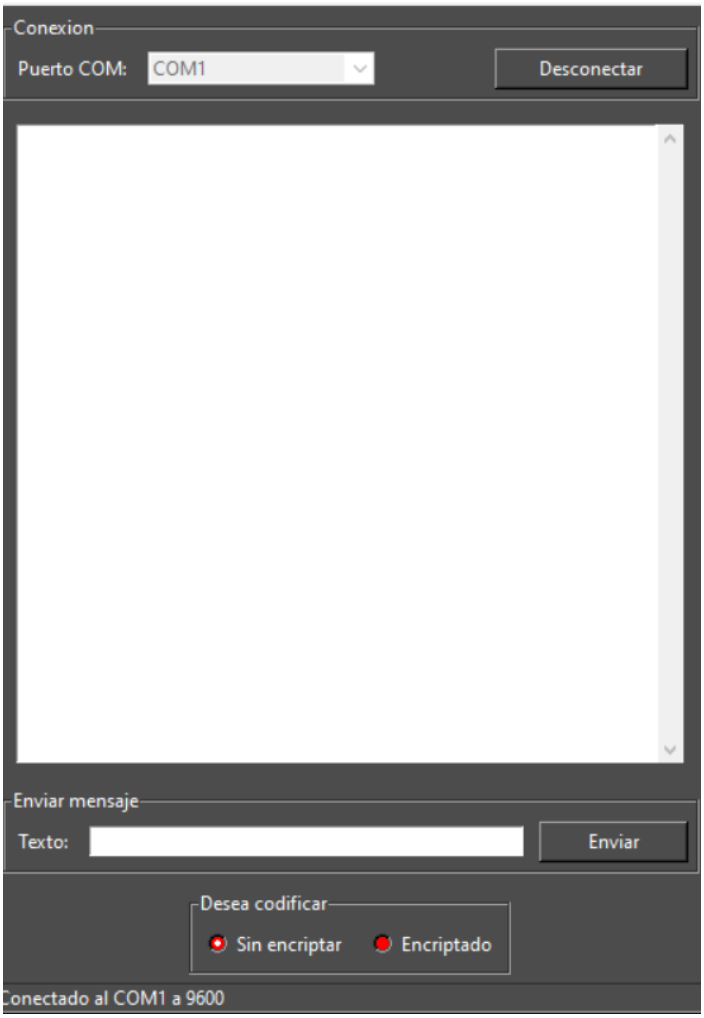


6. Cuando no se escoge ningún puerto y se le da al botón conectar, en el statusBar aparece un mensaje de “Error: No hay puerto seleccionado”

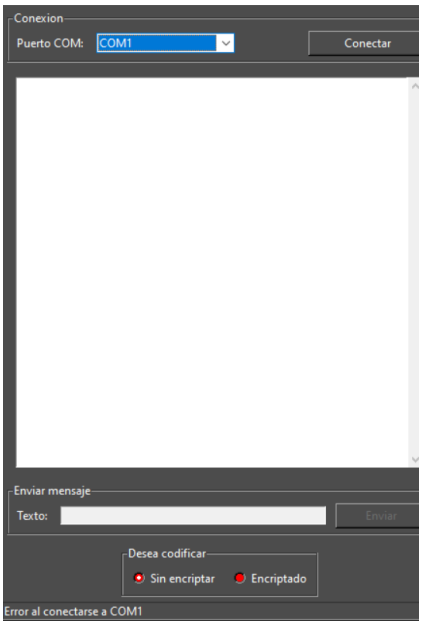
7. En el ComboBox, donde se muestran los puertos instalados en el sistema, no se puede escribir encima.



8. En ambas ventanas se escoge los puertos, uno con COM2 y otro con COM3. Al conectaren ambos casos, aparece el mensaje en el StatusBar que dice “Conectado al {Puerto} a 9600 de velocidad de transmisión”. Adicionalmente, al estar conectado se deshabilita parte del frame 1, y el botón conectar pasa a desconectar.

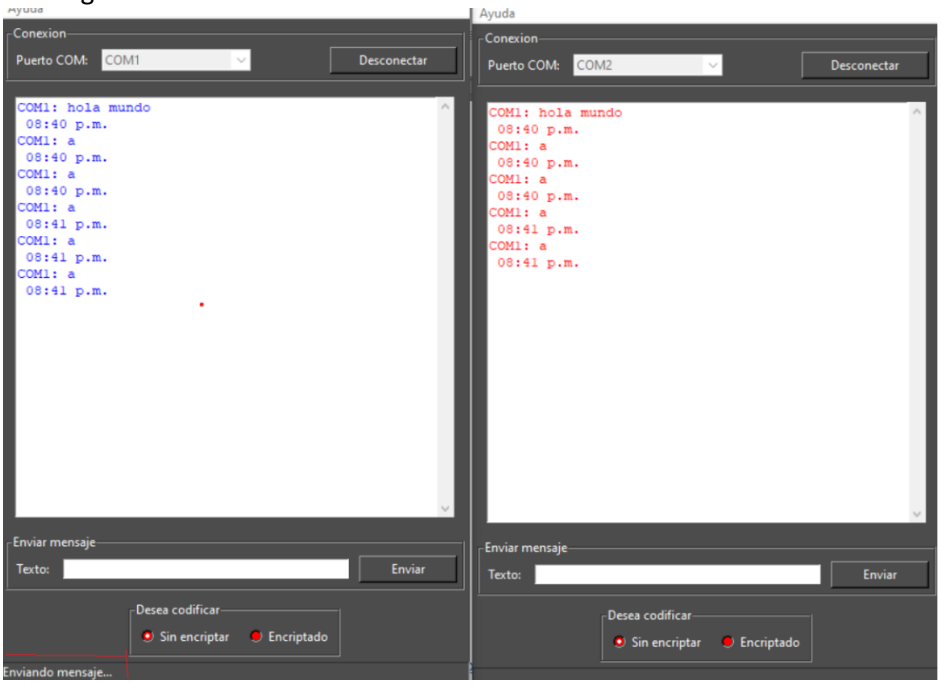


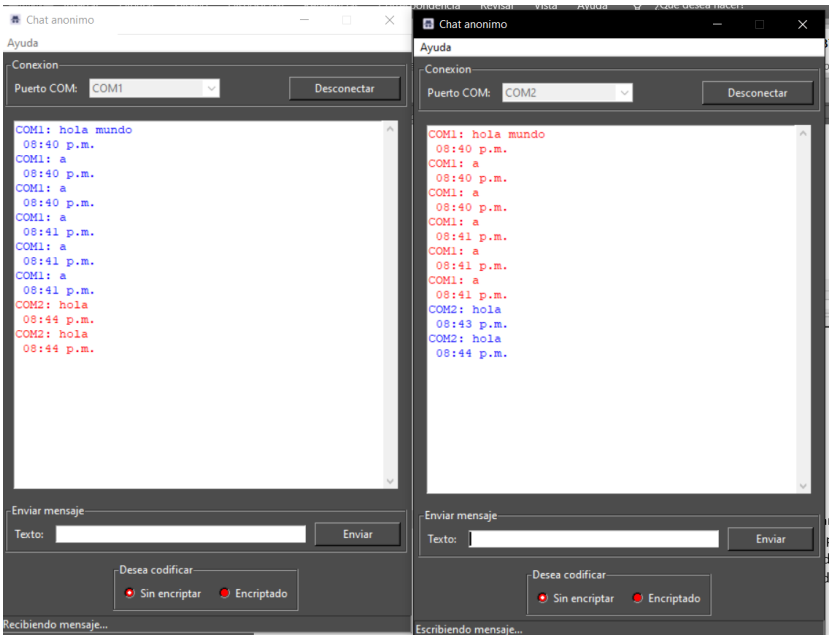
9. Cuando en el segundo chat se escoge el mismo puerto que el primer chat, saldrá “Error al conectarse al puerto”, ya que el puerto del chat1 ya está siendo utilizado.



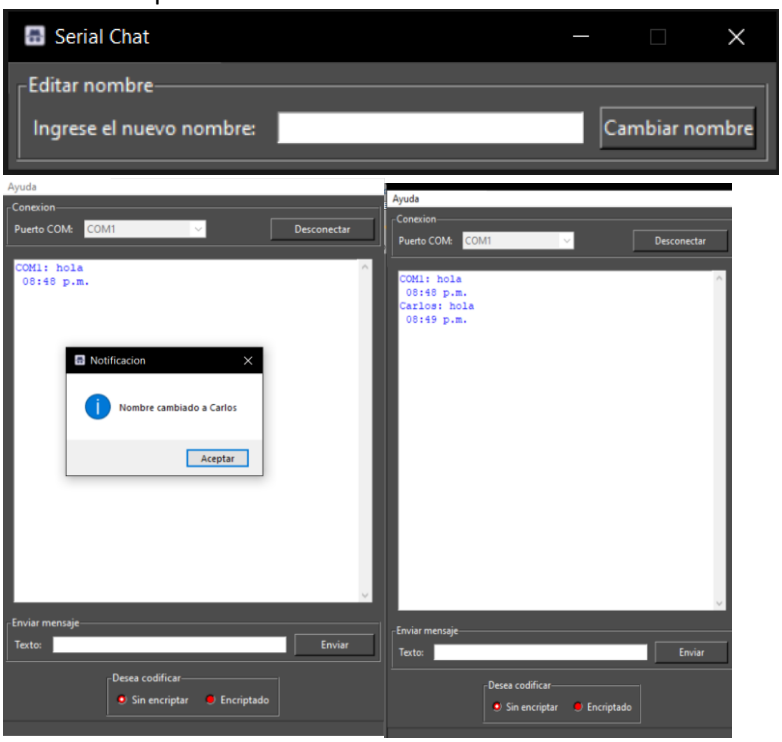
10. Al momento de escribir el mensaje, aparece en el STATUSBAR un mensaje de “Escribiendo mensaje…”. Se puede enviar tanto dando clic en el botón enviar como conla tecla ENTER del teclado.

11. Al enviar el mensaje, se observa que el color azul es el color de la persona que envia y el color rojo es para el que recepciona el mensaje. Además, se observa la hora de envíoen el chat. Asimismo, se observa los mensajes en el STATUSBAR de “Enviando mensaje”y “Recibiendo mensaje” luego de 1segundo de envio.

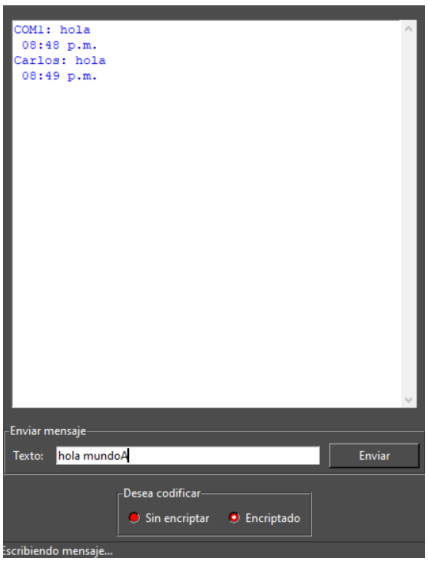


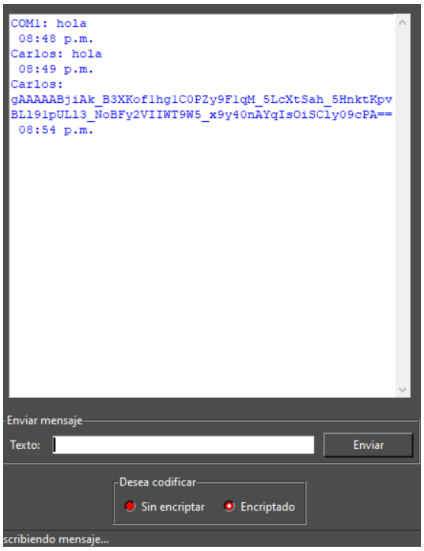


12. Un adicional, es que la persona puede cambiar su nombre de usuario a través de un comando, por si no quiere tener el nombre de COM2 y COM3. Esto lo podemos observaren la ventana de “Ayuda”, donde saldrá una ventana para modificar en nombre deseado.

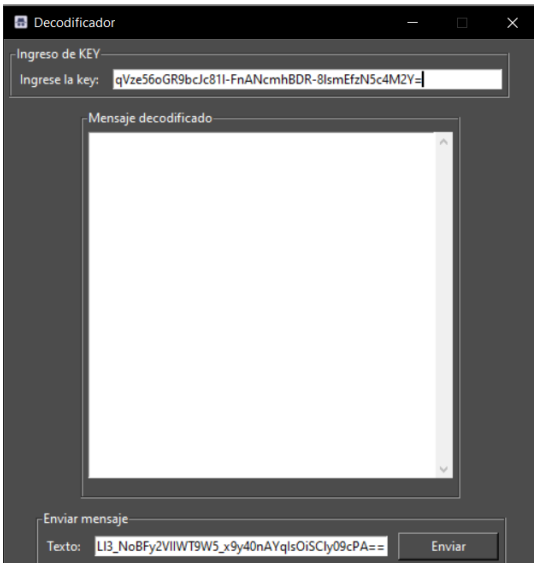


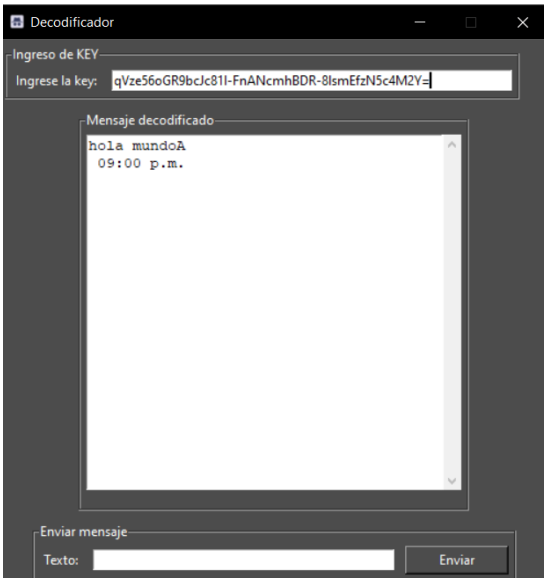
13. Otro adicional es el la encriptación de información que se envia a través del chat



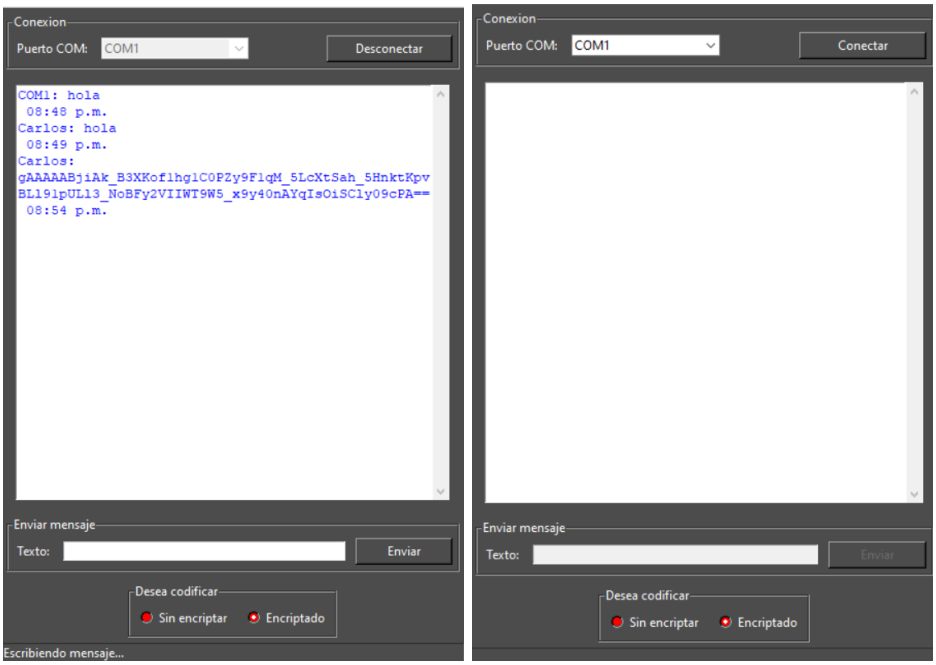


En el menu hay se encuentra la el decodificador que permite traducir la informacion





14. Finalmente, cuando se de en el botón desconectar, así haya texto tanto en el Entry del frame 3 como en el scrolledtext del frame 2, estos se borrarán.



Código del funcionamiento del programa realizado TF-Parte1

```
#TRABAJO FINAL - Carlos Carbajal Jordan
#Se importan la librerias a utilizar
import threading #ejecución varios procesos a la vez: un proceso para
recibir y otro para enviar message
import time
import tkinter as tk
import tkinter.ttk as ttk
import serial
import serial.tools.list_ports
from tkinter.scrolledtext import ScrolledText
from tkinter.messagebox import showinfo
from datetime import datetime
from collections import OrderedDict
from cryptography.fernet import Fernet


class SerialChat:
    def __init__(self, master):

        self.master = master #padre de la ventana. Contiene todo.
        self.puerto=0 #Valor inicial de puerto
        self.puerto_conectado=False #Estado de la conexion
        self.master.title("Chat anonimo")
        self.master.geometry("+50+50")
        self.master.resizable(0, 0)
```

```
        self.master.iconbitmap('icon_chat_sha.ico') #icono del chat
        #####Insertar el menu de información para cambiar nombre de
usuario######
        main_menu = tk.Menu(self.master)

        self.master.config(menu=main_menu,bg="#4c4c4c")
        #mensaje con la key del cifrado (inicia un message box)
        self.inicio=True
        # Se definen los menus (submenus) del menu principal
        info_comandos = tk.Menu(main_menu, tearoff=False)
        #info_comandos.add_command(label="Información para cambiar
nombre", command=self.infoNombre)
        main_menu.add_cascade(label="Ayuda", menu=info_comandos)
        info_comandos.add_command(label="decodificar",command=self.decodi
fica)
        info_comandos.add_command(label="editar_nombre",command=self.camb
iar_nombre)

        ##############Nombre inicial del
boton#########################################

        self.conexion = tk.StringVar() #Texto del boton Conectar
        self.conexion.set("Conectar") #.set, método para cambiar el texto
del StringVar()
        self.var_encrip = tk.IntVar() #para los radio button
        # ---------------------- SERIAL PORT -------------------------
        #Se obtiene información de los puertos disponibles
        ports = serial.tools.list_ports.comports()
        print("SE TIENEN LOS SIGUIENTES PUERTOS A UTILIZAR:\n")
        for port in ports: #Puertos disponibles
            print(port)

        self.puertos=[]

        for port in ports:
                p1,p2 = str(port).split('-')
                p1=p1.replace(' ','')
                self.puertos.append(p1)
        self.puertos=list(OrderedDict.fromkeys(self.puertos))



        # ------------------------ FRAMES ----------------------------
        frm1 = tk.LabelFrame(self.master,
text="Conexion",bg="#4c4c4c",fg="#ffffff")
        self.frm2 = tk.Frame(self.master,bg="#4c4c4c")
        frm3 = tk.LabelFrame(self.master, text="Enviar
mensaje",bg="#4c4c4c",fg="#ffffff")
```

```
        frm4 = tk.LabelFrame(self.master, text="Desea
codificar",bg="#4c4c4c",fg="#ffffff")
        frm1.pack(padx=5, pady=5, anchor=tk.W)
        self.frm2.pack(padx=5, pady=5, fill='y', expand=True)
        frm3.pack(padx=5, pady=5)
        frm4.pack(padx=5, pady=5)

        # ----------------------- FRAME 1 ---------------------------
        self.lblCOM = tk.Label(frm1, text="Puerto
COM:",bg="#4c4c4c",fg="#ffffff")
        #values obtenido en ports = serial.tools.list_ports.comports()
        self.cboPort = ttk.Combobox(frm1, values=self.puertos, state =
'readonly') #readonly para que no se edite el Combobox
        self.lblSpace = tk.Label(frm1, text="",bg="#4c4c4c",fg="#ffffff")
        self.btnConnect = tk.Button(frm1,
textvariable=self.conexion,command=self.seleccionando_puertosCOM,
width=16,bg="#4c4c4c",fg="#ffffff")

        self.lblCOM.grid(row=0, column=0, padx=5, pady=5)
        self.cboPort.grid(row=0, column=1, padx=5, pady=5)
        self.lblSpace.grid(row=0,column=2, padx=30, pady=5)
        self.btnConnect.grid(row=0, column=3, padx=5, pady=5)

        # ----------------------- FRAME 2 ---------------------------
        self.txtChat = ScrolledText(self.frm2, height=25, width=50,
wrap=tk.WORD, state='disable')
        self.txtChat.grid(row=0, column=0, columnspan=3, padx=5, pady=5)

        # ----------------------- FRAME 3 ---------------------------
        self.lblText = tk.Label(frm3,
text="Texto:",bg="#4c4c4c",fg="#ffffff")
        self.inText = tk.Entry(frm3, width=45, state='disable')


        self.inText.bind("<Return>", self.enviar_mensaje) #Tecla ENTER
para enviar mensaje
        self.inText.bind("<Key>", self.actualizandoStatusBar) #detectar
caracteres para "Escribiendo mensaje..."

        #command=lambda: self.enviar_mensaje(None) -> Para darle el
evento que pide.(en este caso None porque no hay evento)
        self.btnSend = tk.Button(frm3, text="Enviar", width=12,
state='disable',command=lambda:
self.enviar_mensaje(None),bg="#4c4c4c",fg="#ffffff")
        #-------------------------------------fmr4----------------------
---
        #---------------------------Selecion  de encriptacion (si o no)
        self.sinconversion = tk.Radiobutton(frm4, text="Sin
encriptar",bg="#4c4c4c", variable=self.var_encrip, value=0,
```

```
                                            fg='white',
activebackground='#4c4c4c',selectcolor='red',activeforeground='white')
        self.conconversion = tk.Radiobutton(frm4,
text="Encriptado",bg="#4c4c4c", variable=self.var_encrip, value=1,
                                            fg='white'
,activebackground='#4c4c4c',selectcolor='red',activeforeground='white')

        self.lblText.grid(row=0, column=0, padx=5, pady=5)
        self.inText.grid(row=0, column=1, padx=5, pady=5)
        self.btnSend.grid(row=0, column=2, padx=5, pady=5)
        self.sinconversion.grid(row=0, column=0, padx=5,
pady=5,sticky=tk.W)
        self.conconversion.grid(row=0, column=1, padx=5,
pady=5,sticky=tk.W)

        # --------------------------- StatusBar ----------------------
        self.statusBar = tk.Label(self.master, bd=1, relief=tk.SUNKEN,
anchor=tk.W,bg="#4c4c4c",fg="#ffffff")
        self.statusBar.pack(side=tk.BOTTOM, fill=tk.X)
        #----------------------------mensaje de entrada de la  key-------
----------------------
        if (self.inicio == True):
            self.KEY = Fernet.generate_key()
            #self.clave = Fernet(self.KEY)
            showinfo(title="key para decodificar", message=f"Tu llave es:
{self.KEY}")
            self.cambionombre=False
            self.inicio = False
        print(f"clave: {self.KEY}")

        # ------------- Control del boton "X" de la ventana -----------
        self.master.protocol("WM_DELETE_WINDOW", self.cerrar_puertos)

        #------------- recibe el mensaje del emisor -----------
        self.master.after(1000,self.recibir_mensaje()) #espera que pase
1seg y llama a recibir_mensaje y dentro hay otro after y allí se hace el
bucle

    #Cuando se cierra la ventana se cierra las comunicaciones y se cierra
la ventana. Solo cuando se clickea en la 'X'
    def cerrar_puertos(self):
        try:
            self.serial.close()
        except:
            pass
        self.master.destroy()
    #El mensaje que aparece en el status cuando se detecta caracteres en
el entry del frame3
    def actualizandoStatusBar(self,event):
```

```
        self.statusBar.config(text='Escribiendo mensaje...')



    ###--Proceso de seleccionar puertos-----
    def seleccionando_puertosCOM(self):
        # Uso get para recibir el valor del combobox
        self.puerto= self.cboPort.get()

        #PARTE DE CONEXION
        #Estado desconectado  (habilitacion de botones e ingreso de info)
        if(self.puerto_conectado==False):# si el puerto está desconectado
            self.puerto_conectado=True #entonces se vuelve conectado
            self.cboPort.config(state='disabled')
            self.conexion.set("Desconectar")
            self.inText.config(state='normal') #La entrada de texto se
activa
            self.btnSend.config(state='normal') #El boton de enviar se
activa

        #Estado de conexion (deshabilita los botones y borrar  info)
        elif(self.puerto_conectado==True):# si el puerto está conectado.
Este elif entra cuando se está desconectando.
            self.conexion.set("Conectar") #el botón de pone en conectar
            self.puerto_conectado=False #puerto desconectado
            self.puerto=0
            self.ser.close() #se cierran las conexiones. 'ser' es el
objeto que representa la conexion serial.
            self.inText.delete(0, 'end') #Para borrar los caracteres en
el entry del frm3 cuando se desconecta.
            ##
            #Creando uno nuevo scrolledText. Simula la limpieza de chat.
            self.txtChat = ScrolledText(self.frm2, height=25, width=50,
wrap=tk.WORD, state='disable')
            self.txtChat.grid(row=0, column=0, columnspan=3, padx=5,
pady=5)
            ##

            #Como es para desconectar la entrada de texto y el boton se
deshabilitan.
            self.inText.config(state='disabled')
            self.btnSend.config(state='disabled')
            self.cboPort.config(state='readonly') #Con esto se habilita
el combobox donde están los puertos COM
            self.statusBar.config(text="") #Para limpiar el StatusBar
cuando se desconecta
            return
        #Lo que está debajo es para conectar.
```

```
        ##Esto se hace cuando se está conectando######
        try:
            #Estableciendo comunicación con puerto serial {PORT}
            self.statusBar.config(text="Conectado al "+str(self.puerto)+"
a 9600")
            self.ser = serial.Serial(port=self.puerto,
                                     baudrate=9600,
                                     bytesize=8,
                                     timeout=2,
                                     stopbits=serial.STOPBITS_ONE)

        #Si no se pudo conectar...
        except:
            self.conexion.set("Conectar") #se pone en conectar el botón
            self.puerto_conectado=True #Puerto desconectado

            self.inText.config(state='disabled') #Entrada de texto se
deshabilitan
            self.btnSend.config(state='disabled') #El boton de enviarse
se deshabilitan
            self.cboPort.config(state='readonly') #El combobox se activa
para seleccionar un puerto. No se puede escribir allí.
            #print(self.puerto)
            if self.puerto == '': #Cuando no se ha seleccionado un
puerto...
                self.statusBar.config(text="Error: No hay puerto
seleccionado")
            else: #Cuando ya esté usado ese puerto...
                self.statusBar.config(text="Error al conectarse a
"+str(self.puerto))
            self.puerto=0 #se resetea el puerto, porque no se ha
conectado nada. Puerto al que se está conectando, pero no se pudo

    #Método para insertar un texto en el ScrolledText. Tanto los mensajes
de los usuarios junto con la hora de envío.
    def insertar_texto(self,mensaje,colormsj): #Argumentos (mensaje,
color del mensaje)
        fecha=f"{datetime.strftime(datetime.now(),'%H'+':'+'%M')}"
        #print(fecha)

        hora = int(fecha[:2])
        minutos = int(fecha[-2:])

        if hora>12:
            self.txtChat.insert(tk.INSERT,str(mensaje)+f"\n {hora-
12:02d}:{minutos:02d} p.m.\n",colormsj)
        else:
            self.txtChat.insert(tk.INSERT,str(mensaje)+f"\n
{hora:02d}:{minutos:02d} a.m.\n",colormsj)
```

```
        self.txtChat.yview(tk.END)

    #---------MÉTODO PARA ENVIAR MENSAJE----------
    def enviar_mensaje(self,event): #el evento es NONE. Se envia con
click en Enviar o con el ENTER del teclado

        # Texto del mensaje enviado
        texto=self.inText.get()
        #--------------Para  seleccionar el nombre de usuario
        if(self.cambionombre == True):
            self.nombreUsuario = self.nombrenuevo
        elif(self.cambionombre == False):
            self.nombreUsuario = self.cboPort.get()

        if (self.var_encrip.get() == 1):
            fernet = Fernet(self.KEY)
            texto_cod=str(fernet.encrypt(texto.encode()))
            texto_cod=texto_cod[2:]
            texto_cod=texto_cod[:-1]

        elif (self.var_encrip.get() == 0):
            texto_cod=str(texto)

        texto=str(self.nombreUsuario)+": "+ texto_cod
        #Se establece cómo se puede cambiar el nombre del usuario a
través de un comando explicado en tk.Menu
        texto_encode=texto.encode("utf-8")
        self.ser.write(texto_encode) #se envia el mensaje hacia el otro
COM.
        self.txtChat.config(state='normal') #ScrolledText estado normal
(habilitado)

        #Al enviar mensaje, se tendrá texto de color azul
        self.insertar_texto(texto,'azul')

            # Threading
        #En contador para borrar el statusBar.Después de un 1seg borra el
mensaje de escribiendo y recibiendo mensaje.
        #Aquí se crea el hilo, que ejecuta el método tiempo_empleado.
        th1sec = threading.Thread(target=self.tiempo_empleado,
args=(1,0,), daemon=True) #1,0 enviando mensaje. (tiempoespera,envio)
        th1sec.start() #Aquí comienza el hilo y comienza a contar 1seg,
por el método tiempo_empleado.

            # Color de letra a azul en el chat
        self.txtChat.tag_config('azul', foreground='blue')
        self.inText.delete(0, 'end') #Luego de enviarse el mensaje debe
de borrarse en el entry.
```

```
        self.txtChat.config(state='disabled') #Deshabilita el chat para
evitar escribir.
        return

    #---------METODO DE RECIBIR MENSAJE--------------
    def recibir_mensaje(self):

        if(self.puerto in self.puertos): #Si el COM está dentro de los
puertos.

            if self.ser.in_waiting > 0: #Cuando hay un mensaje
esperando... (datos esperando)

                mensaje= self.ser.readline() #se lee ese mensaje
                mensaje = mensaje.decode('utf-8') #se decodifica
                self.txtChat.config(state='normal')

                # Threading
                #Se crea un hilo para contar 1seg. Luego de ello se va a
cambiar el texto del statusBar.
                th2sec = threading.Thread(target=self.tiempo_empleado,
args=(1,1,), daemon=True) #1,1 recibiendo mensaje
(tiempoespera,recibiendo)
                th2sec.start()

                #Al recibir mensaje, se tendrá texto de color rojo
                self.insertar_texto(mensaje,'rojo')

                self.txtChat.tag_config('rojo', foreground='red')
                self.txtChat.config(state='disabled')

        self.master.after(1000,self.recibir_mensaje)


    def tiempo_empleado(self,delay,i):
        # El tiempo lo cambias en threading
        if(i==0):
            self.statusBar.config(text="Enviando mensaje... ")
            time.sleep(delay)  # Tiempo de 1s
            self.statusBar.config(text="") #Luego de un 1segundo lo borra

        elif(i==1):
            self.statusBar.config(text="Recibiendo mensaje... ")
            time.sleep(delay)  # Tiempo de 1s
            self.statusBar.config(text="") #Luego de un 1segundo lo borra

        return

    def decodifica(self):
```

```
        #Abre ventana nueva
        self.codigo = tk.Toplevel(self.master)

        self.codigo.title("Decodificador")
        self.codigo.geometry("500x500+100+100")
        self.codigo.resizable(0, 0)
        self.codigo.iconbitmap('icon_chat_sha.ico')
        self.codigo.config(bg="#4c4c4c")

        #----------subventana frames---------------------------------
        self.frm11=tk.LabelFrame(self.codigo,text= "Ingreso de
KEY",bg="#4c4c4c",fg="#ffffff")
        self.frm22 = tk.LabelFrame(self.codigo,text="Mensaje
decodificado",bg="#4c4c4c",fg="#ffffff")
        self.frm33 = tk.LabelFrame(self.codigo, text="Enviar
mensaje",bg="#4c4c4c",fg="#ffffff")
        self.frm11.pack(padx=5, pady=5,anchor=tk.W)
        self.frm22.pack(padx=5, pady=5, fill='y', expand=True)
        self.frm33.pack(padx=5, pady=5)

        #-------------------frm subventana1-------------------------------
------
        #-------------------frame 1-------------------------------------
---------
        self.lblText1 = tk.Label(self.frm11, text="Ingrese la
key:",bg="#4c4c4c",fg="#ffffff")
        self.inText1 = tk.Entry(self.frm11, width=60)
        self.lblText1.grid(row=0, column=0, padx=5, pady=5)
        self.inText1.grid(row=0, column=1, padx=5, pady=5)

        # ------------------------ FRAME 2 ---------------------------
        self.txtChat1 = ScrolledText(self.frm22, height=20, width=40,
wrap=tk.WORD)
        self.txtChat1.grid(row=0, column=0, columnspan=3, padx=5, pady=5)
        #--------------------------FRAME 3-------------------------------
------------
        self.lblText2 = tk.Label(self.frm33,
text="Texto:",bg="#4c4c4c",fg="#ffffff")
        self.inText2 = tk.Entry(self.frm33, width=45)

        self.inText2.bind("<Return>", self.enviar_mensaje_key) #Tecla
ENTER para enviar mensaje
        self.inText2.bind("<Key>", self.actualizandoStatusBar) #detectar
caracteres para "Escribiendo mensaje..."

        self.btnSend1 = tk.Button(self.frm33, text="Enviar",
width=12,command=lambda:
self.enviar_mensaje_key(None),bg="#4c4c4c",fg="#ffffff")
```

```
        self.lblText2.grid(row=0, column=0, padx=5, pady=5)
        self.inText2.grid(row=0, column=1, padx=5, pady=5)
        self.btnSend1.grid(row=0, column=2, padx=5, pady=5)

    def enviar_mensaje_key(self,event):
        #agarra la llave que se puso en el entry para decodificar

        self.KEY_DEC=self.inText1.get()
        fernet_dec = Fernet(self.KEY_DEC)
        #agarra  el dato a desencriptar
        mensaje_cifrar=self.inText2.get()
        #proceso de desencriptar
        mensaje_dec = fernet_dec.decrypt(mensaje_cifrar).decode()
        #fecha
        fecha=f"{datetime.strftime(datetime.now(),'%H'+':'+'%M')}"
        hora = int(fecha[:2])
        minutos = int(fecha[-2:])
        if hora>12:
            self.txtChat1.insert(tk.INSERT,str(mensaje_dec)+f"\n {hora-
12:02d}:{minutos:02d} p.m.\n")

        else:
            self.txtChat1.insert(tk.INSERT,str(mensaje_dec)+f"\n
{hora:02d}:{minutos:02d} a.m.\n")
        self.txtChat.yview(tk.END)
        self.inText2.delete(0, 'end')

    def cambiar_nombre(self):
        self.editar_nombre = tk.Toplevel(self.master)

        self.editar_nombre.title("Serial Chat")
        self.editar_nombre.geometry("+100+100")
        self.editar_nombre.resizable(0, 0)
        self.editar_nombre.iconbitmap('icon_chat_sha.ico')
        self.editar_nombre.config(bg="#4c4c4c")
        #----------------------------Frame-----------------------------
----------------------
        self.frm21=tk.LabelFrame(self.editar_nombre,text= "Editar
nombre",bg="#4c4c4c",fg="#ffffff")
        self.frm21.pack(padx=5, pady=5,anchor=tk.W)

        #--------------------------info-----------------------------------
----------
        self.lblTextnombre = tk.Label(self.frm21, text="Ingrese el nuevo
nombre:",bg="#4c4c4c",fg="#ffffff")
        self.inTextnombre = tk.Entry(self.frm21, width=30)
        self.lblTextnombre.grid(row=0, column=0, padx=5, pady=5)
        self.inTextnombre.grid(row=0, column=1, padx=5, pady=5)
```

```
        self.btnSendnombre = tk.Button(self.frm21, text="Cambiar nombre",
width=12,command=lambda:
self.usuariocambiado(None),bg="#4c4c4c",fg="#ffffff")
        self.btnSendnombre.grid(row=0, column=2, padx=5, pady=5)
        #---------------------------Editar nombre----------------------
-----
    def usuariocambiado(self,event):
        self.nombrenuevo=self.inTextnombre.get()
        self.cambionombre=True
        showinfo(title="Notificacion",message=f"Nombre cambiado a
{self.inTextnombre.get()}")




root = tk.Tk()
app = SerialChat(root)
root.mainloop()
```

Proyecto Final -Parte 2 Chat TCP/IP Fundamentos teóricos Parte 2 Chat TCP/IP: Transmisión de datos Socket

Un Socket es el punto final en una conexión, es decir, un dispositivo o elemento electrónico que se genera gracias al sistema operativo y que permite el envío e información de otros procesos que también hagan uso de estos. Puede definirse también como un mecanismo ideado para la comunicación entre un programa del servidor y un programa del cliente en una red.

Además, se caracteriza por su interconectividad, ya que permite enviar y recibir información entre procesos, incluso cuando estos se encuentran ubicados en diferentes máquinas.

Dentro de los principios básicos que rigen el funcionamiento de un socket, se puede encontrar que para que ocurra el proceso de comunicación entre dos programas, es necesario que uno de los programas pueda localizar o ubicar al otro; igualmente, ambos programas deben contar con la capacidad de intercambiar entre sí. (Keepcoding, 2022)

- **Ethernet**

Ethernet es una tecnología para redes de datos por cable que vincula software y/o hardware entre sí. Esto se realiza a través de cables de redes LAN, de ahí que Ethernet sea concebido habitualmente como una tecnología LAN. Así, Ethernet permite el intercambio de datos entre terminales como, por ejemplo, ordenadores, impresoras, servidores, distribuidores, etc. Conectados en una red local, estos dispositivos establecen conexiones mediante el protocolo Ethernet y pueden intercambiar paquetes de datos entre sí. El protocolo actual y más extendido para ello es IEEE 802.3. (IONOS, 2018)

- **WiFi**

Es una tecnología de red inalámbrica que permite que dispositivos tales como computadoras (portátiles y de escritorio), dispositivos móviles y otros equipos (impresoras y cámaras de video) interactúen con Internet. Permite que estos dispositivos, y muchos más, intercambien información entre sí, creando una red. La conectividad a Internet se produce a través de un enrutador inalámbrico. Cuando se accede a Wi-Fi, se está conectando a un enrutador inalámbrico que permite que sus dispositivos compatibles con Wi-Fi interactúen con Internet. (Cisco, 2022)

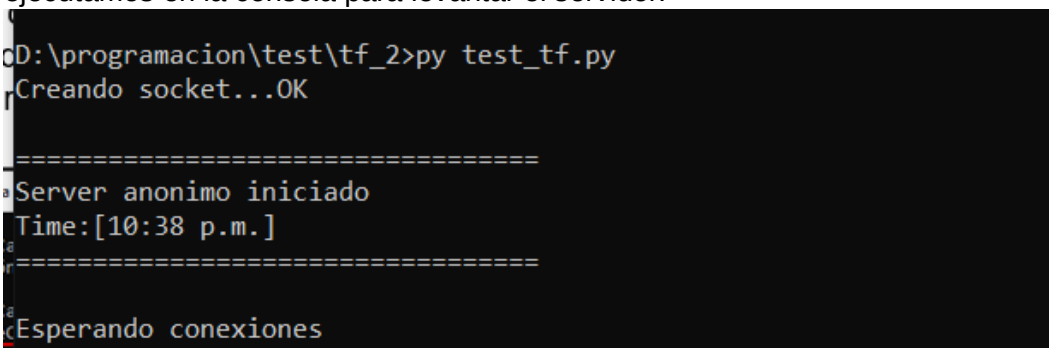
Servidor
Es un aparato informático que almacena, distribuye y suministra información. Los servidores funcionan basándose en el modelo “cliente-servidor”. El cliente puede ser tanto un ordenador como una aplicación que requiere información del servidor para funcionar. Por tanto, un servidor ofrecerá la información demandada por el cliente siempre y cuando el cliente esté autorizado. Los servidores pueden ser físicos o virtuales. (Tic.Portal, 2022)

Requerimiento del TF1-Parte2:
Realizar un programa en tkinter de Chat TCP/IP. Este debe de permitir establecer comunicación entre uno o varios clientes utilizando los servicios de un servidor de chat. El cliente debe de tener un GUI semejante a la aplicación Chat Serial, pero en lugar de seleccionar un puerto serie de conexión, deberá ingresar la dirección IP del servidor y el número de puertos en dos Entry para establecer la conexión. Debe emular un Chat Service (Client/Server).

Explicación del funcionamiento del programa realizado TF-Parte2:

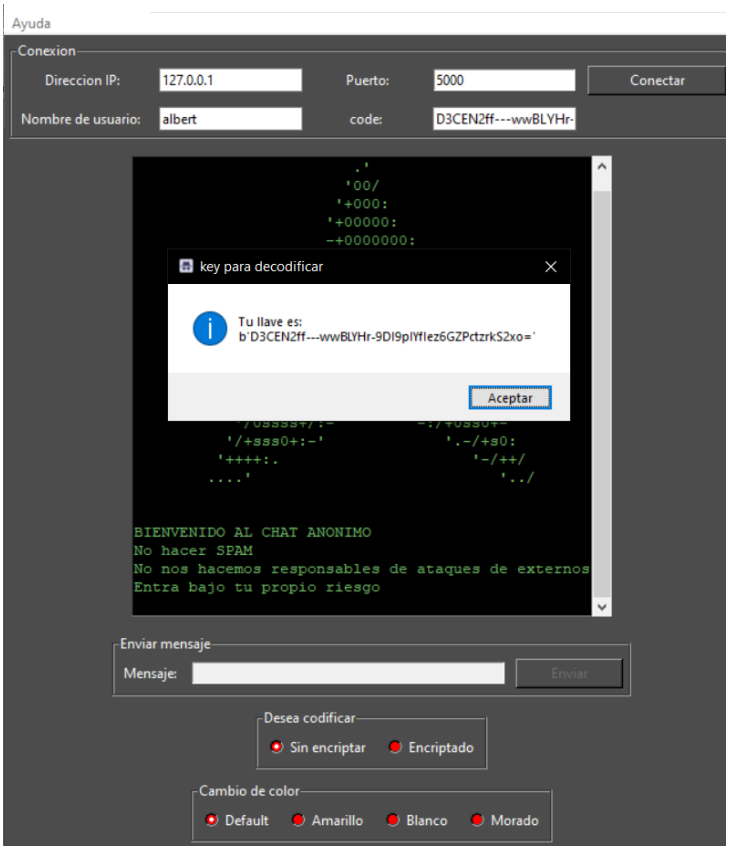
1. Carpeta donde se encuentra el archivo .py y el icono utilizado para un mejor detalle en la interfaz creada.

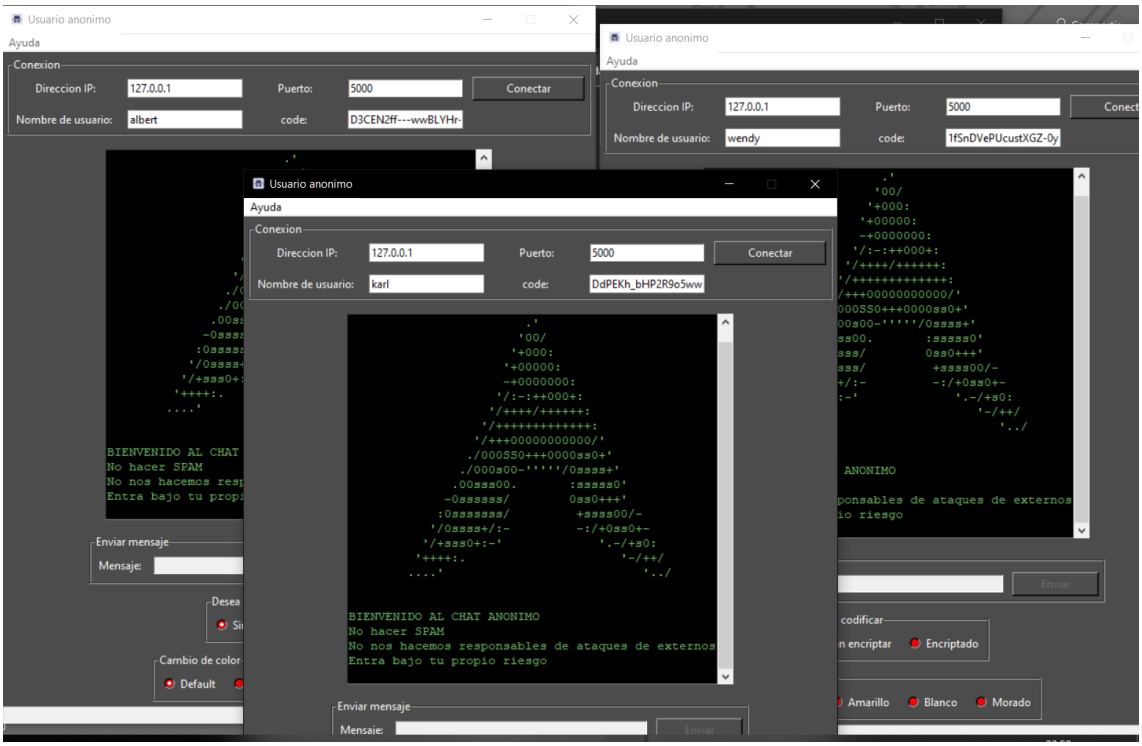
2. Este programa permite establecer la comunicación entre 1 o varios usuarios utilizando el servidor “Server anonimo”. Entonces, ejecutamos en la consola para levantar el servidor.



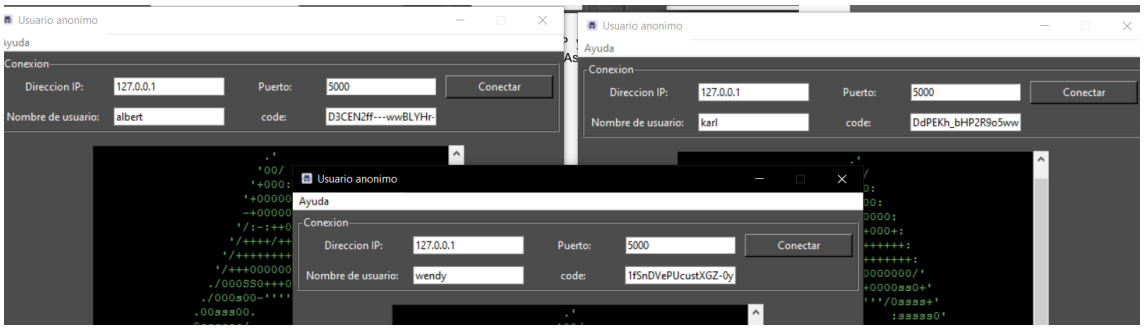
3. Luego iniciamos con la interfaz de cada usuario, el cual es semejante al de la aplicación de Chat Serial. En este caso, ejecutaremos 3 chats. Por lo tanto, se tendrán 3 consolas.

4. Al iniciar se puede observar un icono, en este caso de Incognito y se abrirá una ventana de anuncio con la clave para codificar. Asimismo, se observa una ventana de ayuda que te dará una ventana nueva donde se decodificara los textos que han sido encriptados por fernet. Adicionalmente, se observa que desde un principio el Entry del texto está deshabilitado, al igual que el botón enviar. Además,

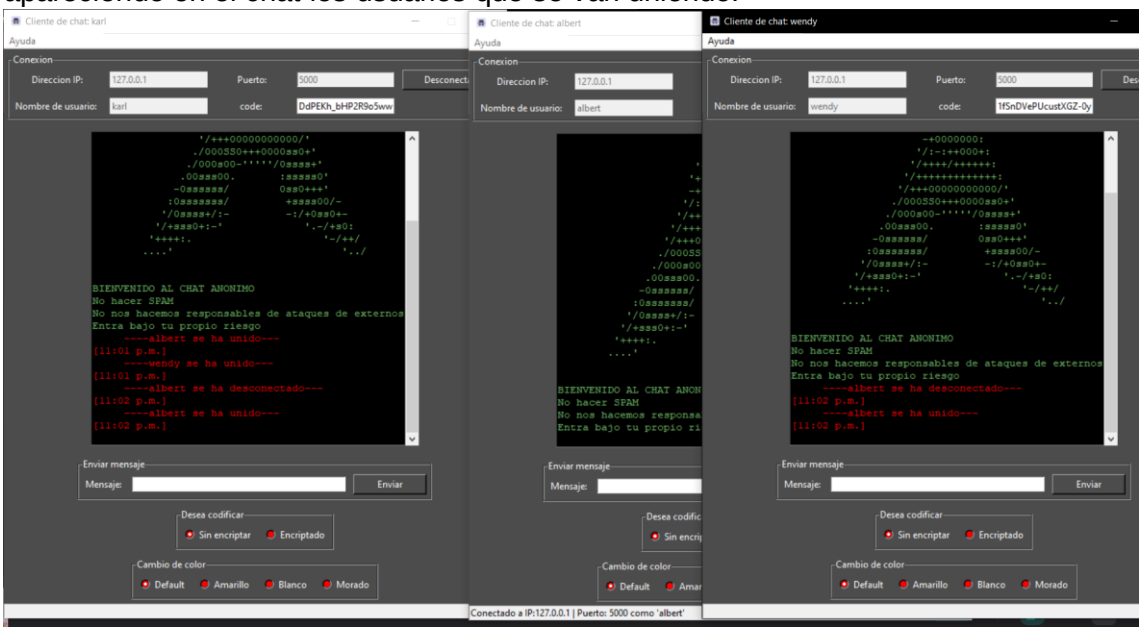




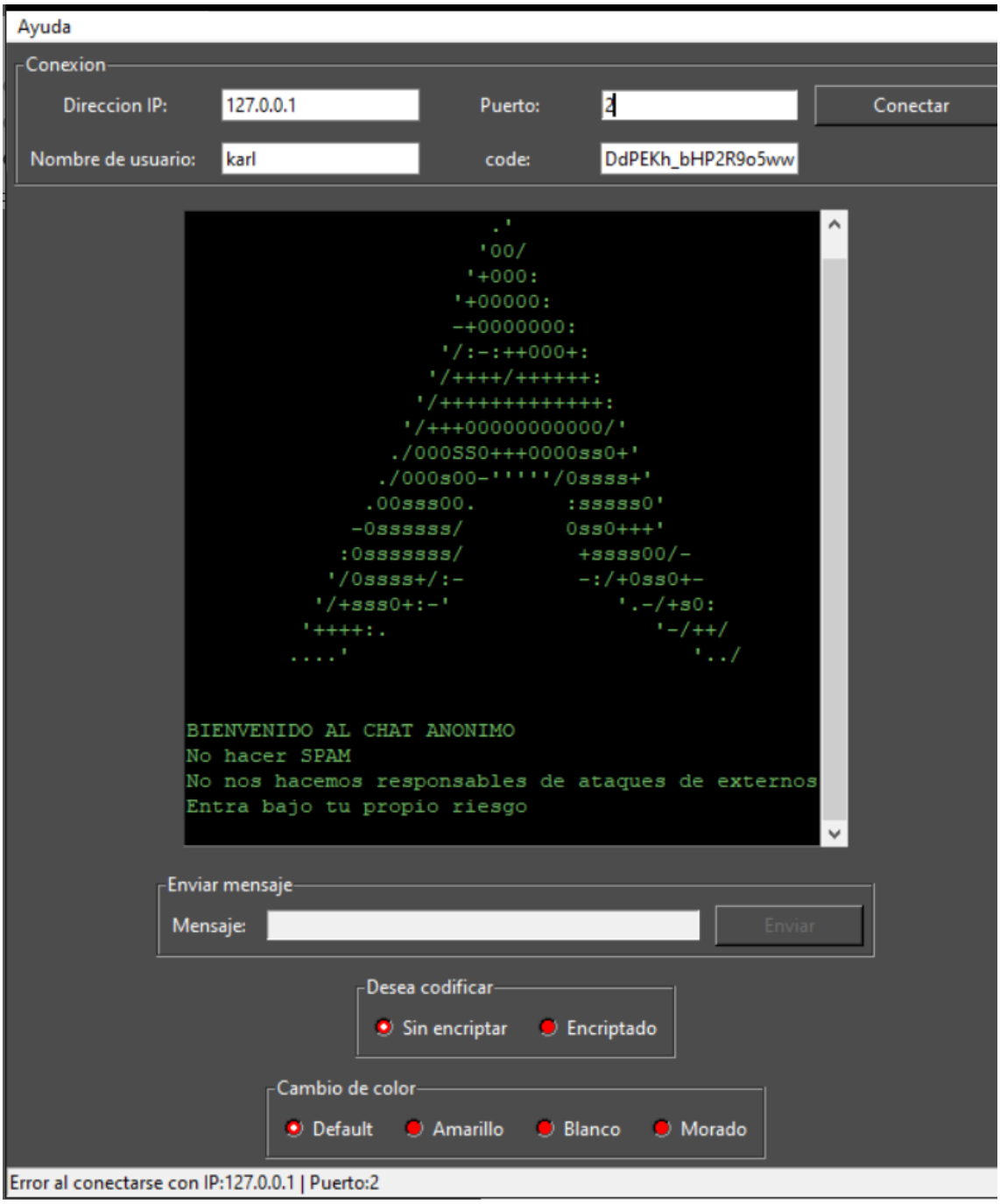
5. En cada ventana de chat se escribe el IP y el Puerto para comunicarse entre todos a través del servidor. Asimismo, se coloca el nombre de usuario deseado.



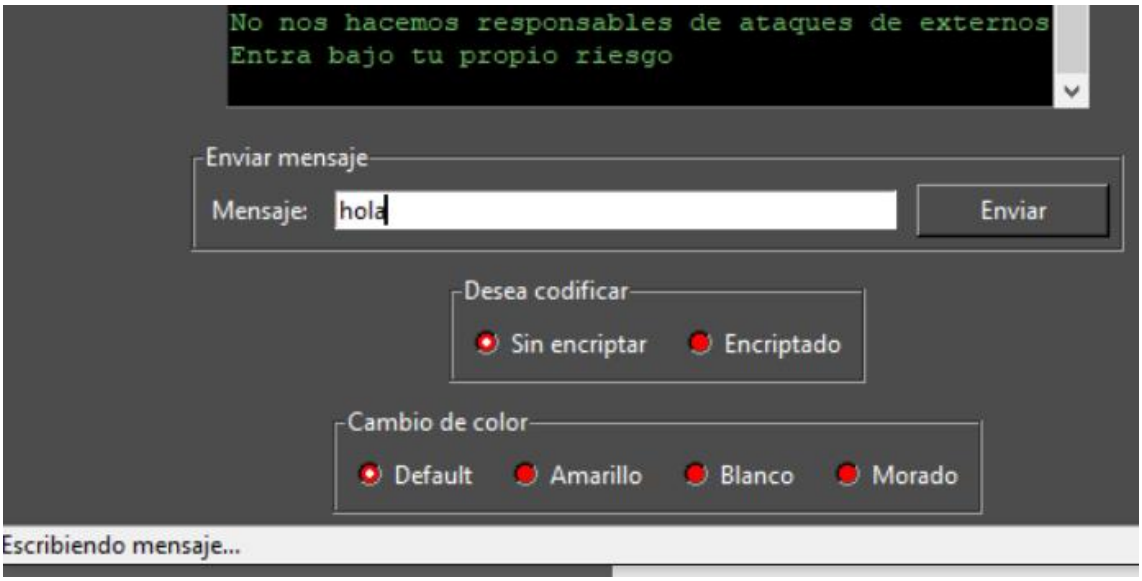
6. Al conectar en los 3 casos, aparece el mensaje en el StatusBar que dice “Conectado a {IP | Puerto} como {nombre de usuario}. Asimismo, el nombre delCliente se actualiza en la parte superior de la Ventana de chat. Al estar conectado se deshabilita parte del frame 1, y el botón conectar pasa a desconectar. Además, va apareciendo en el chat los usuarios que se van uniendo.



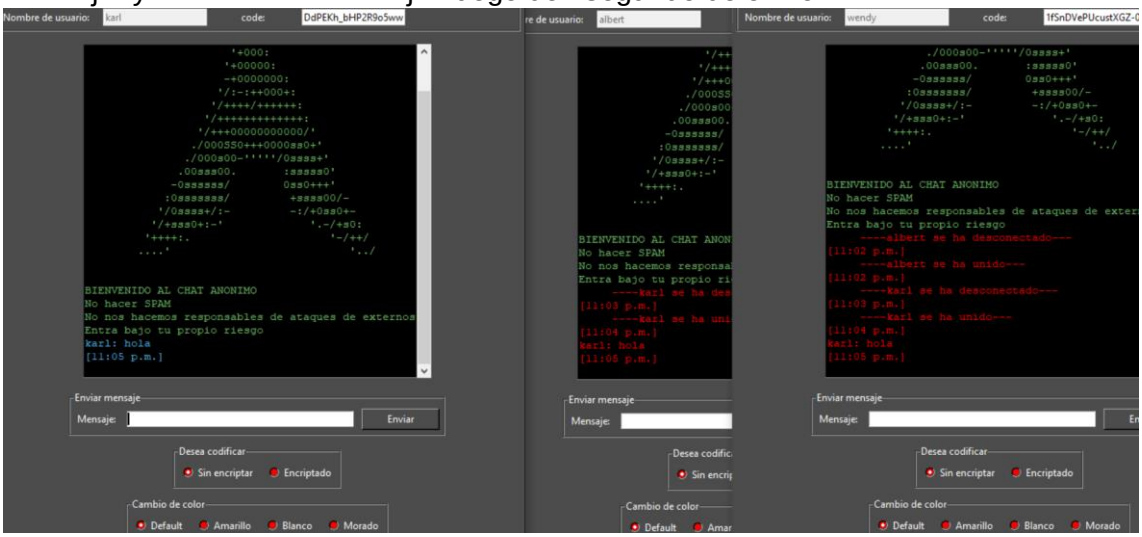
7. Por ejemplo, cuando se coloca una IP incorrecta o Puerto incorrecto y le doy enel botón conectar, en el statusBar aparece un mensaje de “Error: al conectarse con {el valor de la IP ingresada y el Puerto ingresado}”. Y así de la misma formaen todos los chats.



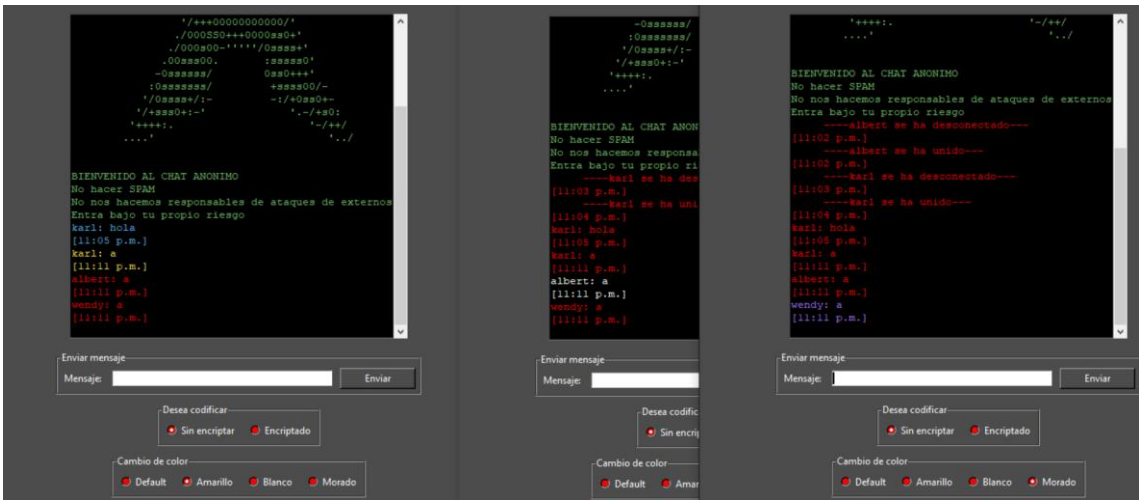
8. Al momento de escribir el mensaje, aparece en el statusbar un mensaje de “Escribiendo mensaje…”. Asimismo, se puede enviar tanto dando clic en el botónenviar como con la tecla ENTER del teclado.



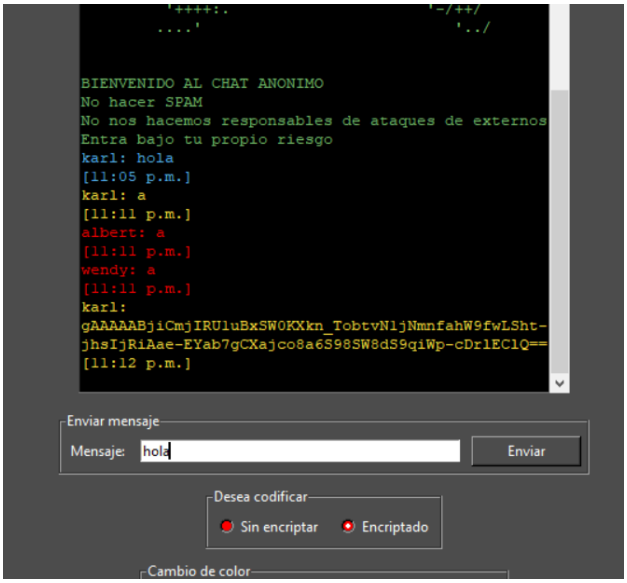
9. Al enviar el mensaje, se observa que el color azul es el color de la persona que envia y el color rojo es para la que recepciona el mensaje. Además, se observa la hora de envío en el chat. Asimismo, se observa los mensajes en el StatusBarde “Enviando mensaje” y “Recibiendo mensaje” luego de 1segundo de envio.

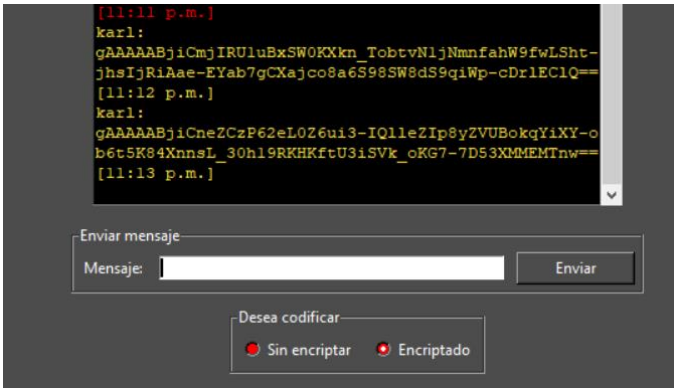


10. Un punto adicional es la ventaja de poder escoger otro color diferente al azul a otra de las opciones disponibles. En este caso se puede escoger entre azul (Default), amarillo, blanco y morado.

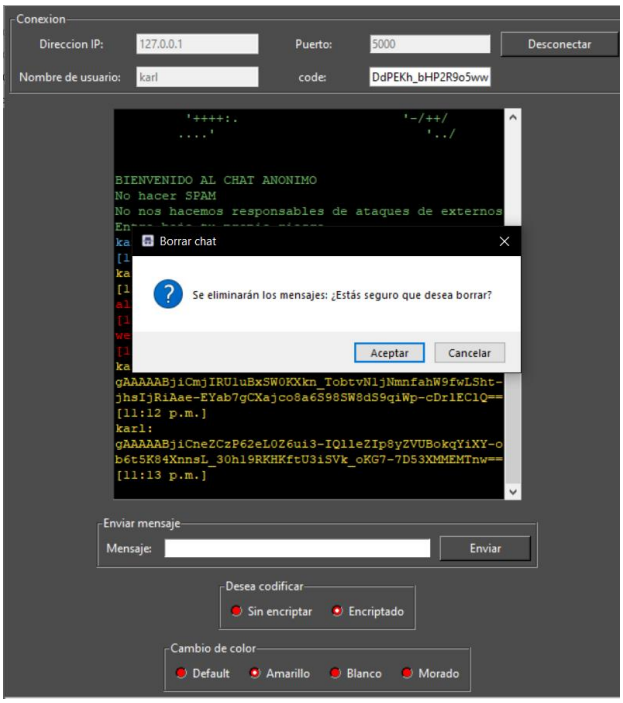


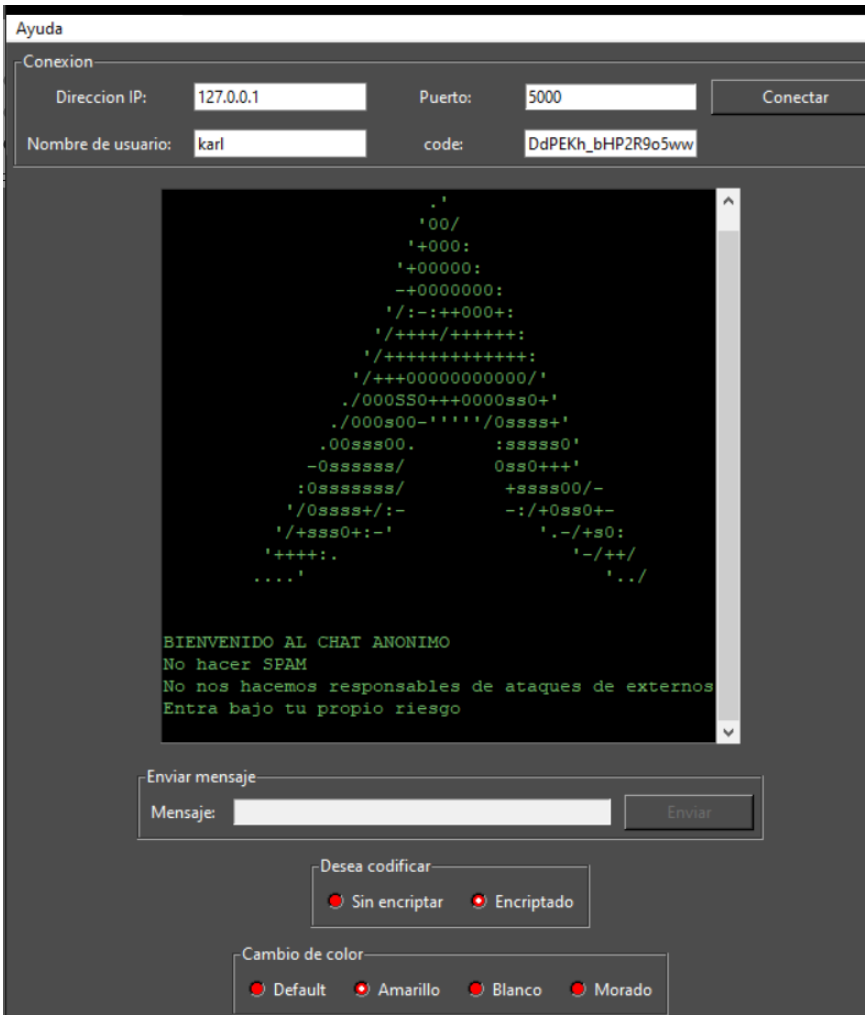
11. Además, se puede enviar mensaje codificado



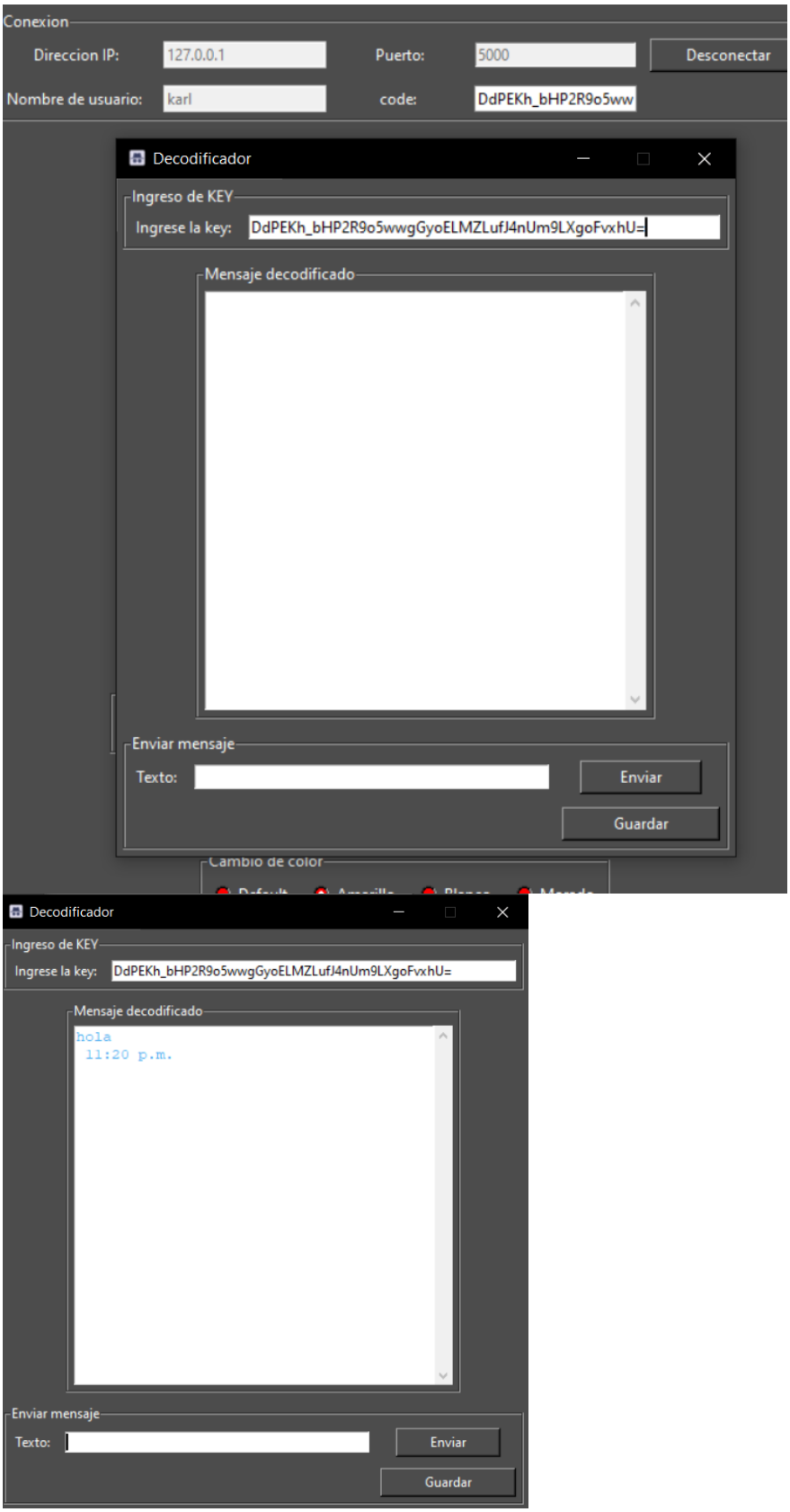


12. Cuando se de en el botón desconectar, así haya texto tanto en el Entry del frame 3 como en el scrolledtext del frame 2, estos se borrarán previamentepreguntando si quiere salir, sabiendo que se borrarán los chats y a su vez se volverá a imprimir el logo del servidor y las reglas del chat

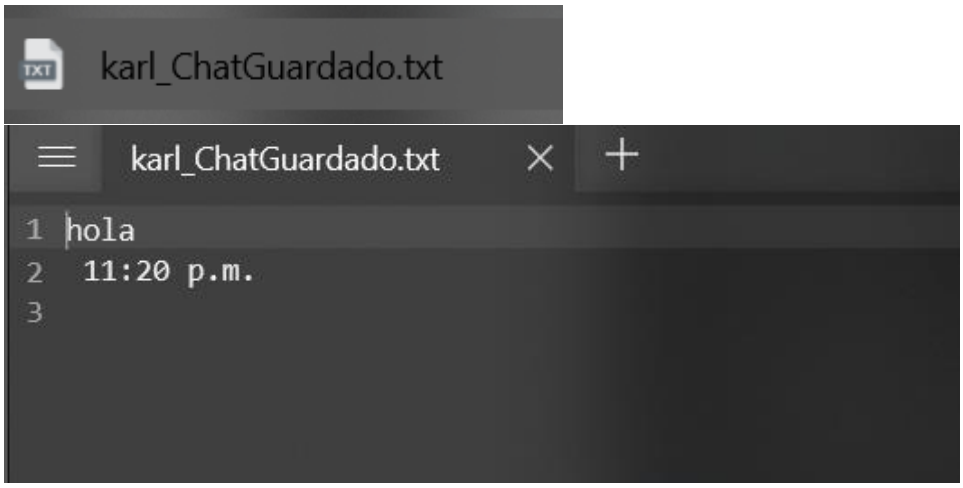




13. En adicional se le puso el key de código Fernet en el Entry donde esta ubicado Code para facilitar la copia de la key. Una vez que se halla copiado el KEY se puede usar la KEY para decodificar como se muestra en la siguiente imagen.



14. Queda recalcar que se agregó un botón para guardar las impresiones que se hicieron en el scrolltext de decodificador y se guardara en un archivo txt con el nombre de usuario



Código del funcionamiento del programa realizado TF-Parte2

```
#TRABAJO FINAL - Carlos Carbajal Jordan

#Se importan la librerias a utilizar
import threading #ejecución varios procesos a la vez: un proceso para
recibir y otro para enviar message
from threading import Event
import sys
import socket
import time
import tkinter as tk
import tkinter.ttk as ttk
import serial
import serial.tools.list_ports
from tkinter.scrolledtext import ScrolledText
from tkinter.messagebox import showinfo, askokcancel
from datetime import datetime
from collections import OrderedDict
from cryptography.fernet import Fernet



HEADER = 10

class Servidor:
    HOST = "127.0.0.1"
    PORT = 5000

    def __init__(self):
        print("Creando socket...", end="")
        self.conexiones = []
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.socket.bind((Servidor.HOST, Servidor.PORT))
        self.socket.listen()
```

```
        print("OK")

        print("\n================================")
        hora = int(datetime.strftime(datetime.now(),'%H'))
        minutos = int(datetime.strftime(datetime.now(),'%M'))
        if hora>12:
            print(f"Server anonimo iniciado\nTime:[{hora-
12:02d}:{minutos:02d} p.m.]")
        else:
            print(f"Server anonimo
iniciado\nTime:[{hora:02d}:{minutos:02d} a.m.]")
        print("================================\n")

    def run(self):
        th =
threading.Thread(target=self.run_hilo,args=(self.socket,),daemon=True)
        th.start()
        try:
            while True:
                pass
        except KeyboardInterrupt:
            self.socket.close()

    def run_hilo(self,s):
        print("Esperando conexiones")
        try:
            while True:
                conn, addr = s.accept()                 #Esperar
peticiones de conexion
                th = threading.Thread(target=self.handler,
args=(conn,addr), daemon=True)   #Crear hilo que maneja la nueva conexion
                th.start()                                          #Come
nzar hilo
                print(str(addr[0])+":"+str(addr[1]),"conectado")    #Info
rmar sobre la nueva conexion
                self.conexiones.append(conn)                        #Agre
gar la nueva conexion a la lista
        except Exception as e:
            print(e)
            print("Servidor cerrado")

    def handler(self,conn,addr):
        while True:
            try:
                data_header = conn.recv(HEADER)             #Recivir
cantidad de bytes
                data = conn.recv(int(data_header))          #Recivir
mensaje
```

```
                for conexion in self.conexiones:            #Para todas
las conexiones
                    if conexion != conn:                    #Excepto el
emisor
                        conexion.send(data_header+data)     #Mandar el
mismo mensaje
            except:
                print(str(addr[0])+":"+str(addr[1]),"desconectado")
#Informar sobre la desconexion
                self.conexiones.remove(conn)                #Quitar de la
lista
                conn.close()                                #Cerrar
conexion (liberar socket)
                break                                       #Salir del
bucle y terminar metodo

class ClienteGUI:
    def __init__(self,name):

        #-------PARTE DE INTERFAZ-------

        self.master = tk.Tk()
        self.master.title("Usuario anonimo")
        self.master.geometry("+50+50")
        self.master.resizable(0, 0)
        self.master.iconbitmap('icon_chat_sha.ico') #icono de wsp
        self.lineas_archivo = []
        #mensaje con la key del cifrado (inicia un message box)
        self.inicio=True
        #genero la key
        self.KEY = Fernet.generate_key()

        #Menu pricipal (toda la barra)
        main_menu = tk.Menu(self.master)
        #Agregar menu principal a la ventana
        self.master.config(menu=main_menu,bg="#4c4c4c")
        #Crear una lista de opciones (submenu sin nombre)
        info_comandos = tk.Menu(main_menu, tearoff=False)
        #Agregar opciones al submenu. Cada opcion ejecuta una funcion
        info_comandos.add_command(label="decodificar",command=self.decodi
fica)
        #Agregar submenu al menu principal con un nombre
        main_menu.add_cascade(label="Ayuda", menu=info_comandos)


        self.var_encrip = tk.IntVar() #para los radio button
        self.var_color=tk.IntVar()
        # ---------------------- FRAMES ----------------------------
```

```
        frm1 = tk.LabelFrame(self.master,
text="Conexion",bg="#4c4c4c",fg="#ffffff")
        self.frm2 = tk.Frame(self.master,bg="#4c4c4c")
        frm3 = tk.LabelFrame(self.master, text="Enviar
mensaje",bg="#4c4c4c",fg="#ffffff")
        frm4 = tk.LabelFrame(self.master, text="Desea
codificar",bg="#4c4c4c",fg="#ffffff")
        frm5 = tk.LabelFrame(self.master, text="Cambio de
color",bg="#4c4c4c",fg="#ffffff")

        frm1.pack(padx=5, pady=5, anchor=tk.W)
        self.frm2.pack(padx=5, pady=5, fill='y', expand=True)
        frm3.pack(padx=5, pady=5)
        frm4.pack(padx=5, pady=5)
        frm5.pack(padx=5, pady=5)

        # ------------------------ FRAME 1 ----------------------------

        #Textos de widgets
        self.conexionVar = tk.StringVar(value="Conectar")
        self.ipVar = tk.StringVar(value="127.0.0.1")
        self.nombreVar = tk.StringVar(value="5000")
        self.name_antes = tk.StringVar(value=sys.argv[1])
        self.code=tk.StringVar(value=self.KEY)

        #Label, Entry y Button
        self.lblIP = tk.Label(frm1, text="Direccion
IP:",bg="#4c4c4c",fg="#ffffff")
        self.entryIP = ttk.Entry(frm1,textvariable=self.ipVar)
        self.lblPuerto = tk.Label(frm1,
text="Puerto:",bg="#4c4c4c",fg="#ffffff")
        self.entryPuerto = tk.Entry(frm1,textvariable=self.nombreVar)
        self.btnConnect = tk.Button(frm1,
textvariable=self.conexionVar,command=self.conectar_con_servidor,
width=16,bg="#4c4c4c",fg="#ffffff")

        #Colocar nombre
        self.nombre_antes_conectado = tk.Label(frm1, text="Nombre de
usuario: ",bg="#4c4c4c",fg="#ffffff")
        self.nombre_antes_conectar =
ttk.Entry(frm1,textvariable=self.name_antes)
        #KEY generada
        self.nombre_key = tk.Label(frm1, text="code:
",bg="#4c4c4c",fg="#ffffff")
        self.keygenerado = ttk.Entry(frm1,textvariable=self.code)

        self.lblIP.grid(row=0, column=0, padx=5, pady=5)
        self.entryIP.grid(row=0,column=1,padx=5,pady=5)
        self.lblPuerto.grid(row=0,column=2, padx=30, pady=5)
```

```
        self.entryPuerto.grid(row=0,column=3,padx=5,pady=5)
        self.btnConnect.grid(row=0, column=4, padx=5, pady=5)


        self.nombre_antes_conectado.grid(row=1, column=0, padx=5, pady=5)
        self.nombre_antes_conectar.grid(row=1,column=1,padx=5,pady=5)
        self.nombre_key .grid(row=1, column=2, padx=5, pady=5)
        self.keygenerado.grid(row=1,column=3,padx=5,pady=5)
        # ------------------------ FRAME 2 --------------------------
        self.crearTxtChat()
        self.normas()

        # ------------------------ FRAME 3 --------------------------
        #Texto de widget
        self.msgVar = tk.StringVar()

        #Label, Entry y Button
        self.lblText = tk.Label(frm3,
text="Mensaje:",bg="#4c4c4c",fg="#ffffff")
        self.inText = tk.Entry(frm3, textvariable=self.msgVar,width=45,
state='disable')
        #command=lambda: self.enviar_mensaje(None) -> Para darle el
evento que pide.(en este caso None porque no hay evento)
        self.btnSend = tk.Button(frm3, text="Enviar", width=12,
state='disable',command=lambda:
self.enviar_mensaje(None),bg="#4c4c4c",fg="#ffffff")

        self.inText.bind("<Return>", self.enviar_mensaje) #Tecla ENTER
para enviar mensaje
        self.inText.bind("<Key>", self.actualizandoStatusBar) #detectar
caracteres para "Escribiendo mensaje..."
        self.lblText.grid(row=0, column=0, padx=5, pady=5)
        self.inText.grid(row=0, column=1, padx=5, pady=5)
        self.btnSend.grid(row=0, column=2, padx=5, pady=5)
        #-------------------------------fmr4-----------------------------
-----------
        #----------------------------Selecion  de encriptacion (si o no)
        self.sinconversion = tk.Radiobutton(frm4, text="Sin
encriptar",bg="#4c4c4c", variable=self.var_encrip, value=0,
                                            fg='white',
activebackground='#4c4c4c',selectcolor='red',activeforeground='white')
        self.conconversion = tk.Radiobutton(frm4,
text="Encriptado",bg="#4c4c4c", variable=self.var_encrip, value=1,
                                            fg='white'
,activebackground='#4c4c4c',selectcolor='red',activeforeground='white')

        self.lblText.grid(row=0, column=0, padx=5, pady=5)
        self.inText.grid(row=0, column=1, padx=5, pady=5)
        self.btnSend.grid(row=0, column=2, padx=5, pady=5)
```

```
        self.sinconversion.grid(row=0, column=0, padx=5,
pady=5,sticky=tk.W)
        self.conconversion.grid(row=0, column=1, padx=5,
pady=5,sticky=tk.W)

        #----------------------------Selecion  de  color  (si o no)
        self.colorVar = tk.StringVar()
        self.sinccambiocolor = tk.Radiobutton(frm5,
text="Default",bg="#4c4c4c", variable=self.var_color, value=0,
                                              fg='white',
activebackground='#4c4c4c',selectcolor='red',activeforeground='white')
        self.coloramarillo = tk.Radiobutton(frm5,
text="Amarillo",bg="#4c4c4c", variable=self.var_color, value=1,
                                              fg='white'
,activebackground='#4c4c4c',selectcolor='red',activeforeground='white')
        self.colorblanco = tk.Radiobutton(frm5,
text="Blanco",bg="#4c4c4c", variable=self.var_color, value=2,
                                              fg='white'
,activebackground='#4c4c4c',selectcolor='red',activeforeground='white')
        self.colormorado = tk.Radiobutton(frm5,
text="Morado",bg="#4c4c4c", variable=self.var_color, value=3,
                                              fg='white'
,activebackground='#4c4c4c',selectcolor='red',activeforeground='white')

        self.sinccambiocolor.grid(row=0, column=0, padx=5,
pady=5,sticky=tk.W)
        self.coloramarillo.grid(row=0, column=1, padx=5,
pady=5,sticky=tk.W)
        self.colorblanco.grid(row=0, column=2, padx=5,
pady=5,sticky=tk.W)
        self.colormorado.grid(row=0, column=3, padx=5,
pady=5,sticky=tk.W)

        # --------------------------- StatusBar ----------------------
        self.statusBar = tk.Label(self.master, bd=1, relief=tk.SUNKEN,
anchor=tk.W)
        self.statusBar.pack(side=tk.BOTTOM, fill=tk.X)

        # ------------- Control del boton "X" de la ventana -----------
        self.master.protocol("WM_DELETE_WINDOW", self.cerrar_ventana)
        #----------------------------mensaje de entrada de la  key-------
----------------------
        if (self.inicio == True):
            #self.clave = Fernet(self.KEY)
            showinfo(title="key para decodificar", message=f"Tu llave es:
{self.KEY}")
            self.cambionombre=False
            self.inicio = False
        print(f"clave: {str(self.KEY)}")
```

```
        #------PARTE DE CONEXION--------

        self.conectado = False
        self.nombreUsuario = name
        self.evento = Event()

    def boton_guardar(self):
        self.statusBar.config(text=f"Decodificado de {self.nombreUsuario}
guardado")
        archivo=open(f"{self.nombreUsuario}_ChatGuardado.txt","w")
        archivo.writelines(self.lineas_archivo)
        archivo.close()

    def crearTxtChat(self):
        self.txtChat = ScrolledText(self.frm2, height=25, width=50,
wrap=tk.WORD, state='disable')
        self.txtChat.grid(row=0, column=0, columnspan=3, padx=5, pady=5)
        self.txtChat.config(bg = '#000000')

        self.txtChat.tag_config('azul', foreground='#53b1f1')
        self.txtChat.tag_config('rojo', foreground='red')
        self.txtChat.tag_config('verde', foreground='#75bf6a')
        #adicionales
        self.txtChat.tag_config('amarillo', foreground='#ffdf3d')
        self.txtChat.tag_config('blanco', foreground='#ffffff')
        self.txtChat.tag_config('morado', foreground='#a970ff')



    #Conectar con servidor
    def conectar_con_servidor(self):
        # Uso get para recibir el valor de los entry
        self.ip = self.entryIP.get()
        self.puerto = int(self.entryPuerto.get())

        #PARTE PARA CONECTAR
        if self.conectado==False:# si el puerto está desconectado
            try:
                self.nombreUsuario = self.name_antes.get()

                self.statusBar.config(text=f"Conectando a IP: {self.ip} |
Puerto: {self.puerto} como '{self.nombreUsuario}'")
                #Estableciendo comunicación con servidor
                self.socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
                self.socket.connect((self.ip, self.puerto))
```

```
                #Configurar widgets
                self.entryPuerto.config(state='disabled')
                self.entryIP.config(state='disabled')
                self.conexionVar.set("Desconectar")
                self.inText.config(state='normal') #La entrada de texto
se activa
                self.btnSend.config(state='normal') #El boton de enviar
se activa
                #self.codecolor.config(state='normal')#activo ingreso de
color
                #self.escogecolores=self.codecolor.get()
                self.conectado=True #entonces se vuelve conectado
                self.statusBar.config(text=f"Conectado a IP:{self.ip} |
Puerto: {self.puerto} como '{self.nombreUsuario}'")
                self.hilo_recibir =
threading.Thread(target=self.recibir_mensaje,daemon=True)
                self.hilo_recibir.start()
                self.txtChat.delete(0, 'end')
                msg = f"     ----{self.nombreUsuario} se ha unido---"
                msg_encode = msg.encode("utf-8")
                msg_encode =
(f"{len(msg_encode):<{HEADER}}{msg}").encode("utf-8")
                self.socket.send(msg_encode)

                self.master.title(f"Cliente de chat:
{self.nombreUsuario}")

                self.nombre_antes_conectar.config(state = 'disabled')
            #Si no se pudo conectar...
            except Exception as e:
                print(e)
                self.statusBar.config(text=f"Error al conectarse con
IP:{self.ip} | Puerto:{self.puerto}")
        #PARTE PARA DESCONECTAR
        else:# si el puerto está conectado. Este elsef entra cuando se
está desconectando.

            if askokcancel(title="Borrar chat", message="Se eliminarán
los mensajes: ¿Estás seguro que desea borrar?"):
                msg = f"     ----{self.nombreUsuario} se ha desconectado-
--"
                msg_encode = msg.encode("utf-8")
                msg_encode =
(f"{len(msg_encode):<{HEADER}}{msg}").encode("utf-8")
                self.socket.send(msg_encode)
                time.sleep(0.5)

                self.socket.close()
```

```
                self.conexionVar.set("Conectar") #el botón de pone en
conectar
                self.inText.delete(0, 'end') #Para borrar los caracteres
en el entry del frm3 cuando se desconecta.
                #Creando uno nuevo scrolledText. Simula la limpieza de
chat.
                self.crearTxtChat()
                #Como es para desconectar la entrada de texto y el boton
se deshabilitan.
                self.normas()

                self.inText.config(state='disabled')
                self.btnSend.config(state='disabled')
                #self.codecolor.config(state='disabled')
                self.entryPuerto.config(state='normal')
                self.entryIP.config(state='normal')
                self.evento.set()
                self.conectado=False #puerto desconectado
                self.master.title("Cliente de Chat")
                self.nombreUsuario = sys.argv[1]
                self.nombre_antes_conectar.config(state = 'normal')
    #Cierra ventana y conexiones
    def cerrar_ventana(self):
        try:
           self.socket.close()
        except:
            pass
        self.master.destroy()

    #Abre informacion sobre el servidor
    def infoServidor(self):
        showinfo(title="Información Servidor",
message=f"IP:{Servidor.HOST}\n Puerto:{Servidor.PORT}")


    def enviar_mensaje(self,event): #el evento es NONE. Se envia con
click en Enviar o con el ENTER del teclado
        # Texto del mensaje enviado
        texto_entry = self.inText.get()


        if (self.var_encrip.get() == 1):
            fernet = Fernet(self.KEY)
            texto_cod=str(fernet.encrypt(texto_entry.encode()))
            texto_cod=texto_cod[2:]
            texto_cod=texto_cod[:-1]

        elif (self.var_encrip.get() == 0):
            texto_cod=str(texto_entry)
```

```
        texto = self.nombreUsuario + ": " + texto_cod
        texto_serial = texto.encode("utf-8")


        if (texto_entry != ''):
            texto_encode = f"{len(texto_serial):<{HEADER}}".encode("utf-
8")+texto_serial # se crea el mensaje
            self.socket.send(texto_encode) #se envia el mensaje hacia el
servidor.
            self.txtChat.config(state='normal') #ScrolledText estado
normal (habilitado)

            #Al enviar mensaje, se tendrá texto de color azul
            if(self.var_color.get()==0):
                self.insertar_texto(texto,'azul')
            if(self.var_color.get()==1):
                self.insertar_texto(texto,'amarillo')
            if(self.var_color.get()==2):
                self.insertar_texto(texto,'blanco')
            if(self.var_color.get()==3):
                self.insertar_texto(texto,'morado')

            # Threading
            #El contador para borrar el statusBar.Después de un 1seg
borra el mensaje de escribiendo y recibiendo mensaje.
            #Aquí se crea el hilo, que ejecuta el método tiempo.
            th1sec = threading.Thread(target=self.tiempo, args=(1,0,),
daemon=True) #1,0 enviando mensaje. (tiempoespera,envio)
            th1sec.start() #Aquí comienza el hilo y comienza a contar
1seg, por el método tiempo.

            #Borrar entrada de texto
            self.inText.delete(0, 'end') #Luego de enviarse el mensaje
debe de borrarse en el entry.
            self.txtChat.config(state='disabled') #Deshabilita el chat
para evitar escribir.



    def recibir_mensaje(self):
        while True:
            try:
                data_header = self.socket.recv(HEADER)
                if data_header:
                    data = self.socket.recv(int(data_header))
                    mensaje = data.decode('utf-8') #se decodifica
                    self.txtChat.config(state='normal')
```

```
                    # Threading
                    #Se crea un hilo para contar 1seg. Luego de ello se
va a cambiar el texto del statusBar.
                    th2sec = threading.Thread(target=self.tiempo,
args=(1,1,), daemon=True) #1,1 recibiendo mensaje
(tiempoespera,recibiendo)
                    th2sec.start()
                    #Al recibir mensaje, se tendrá texto de color rojo
                    self.insertar_texto(mensaje,'rojo')
                    self.txtChat.config(state='disabled')
            except:
                break
        print("Conexion cerrada")
        self.statusBar.config(text="Desconectado") #Para limpiar el
StatusBar cuando se desconecta

    #El mensaje que aparece en el status cuando se detecta caracteres en
el entry del frame3
    def actualizandoStatusBar(self,event):
        self.statusBar.config(text='Escribiendo mensaje...')

    #Método para insertar un texto en el ScrolledText. Tanto los mensajes
de los usuarios junto con la hora de envío.
    def insertar_texto(self,mensaje,colormsj): #Argumentos (mensaje,
color del mensaje)
        hora = int(datetime.strftime(datetime.now(),'%H'))
        minutos = int(datetime.strftime(datetime.now(),'%M'))
        linea = ""

        if hora>12:
            linea =str(mensaje)+f"\n[{hora-12:02d}:{minutos:02d} p.m.]\n"

        else:
            linea =str(mensaje)+f"\n[{hora:02d}:{minutos:02d} a.m.]\n"
        self.txtChat.insert(tk.INSERT,linea,colormsj)


        self.txtChat.yview(tk.END)


    def tiempo(self,delay,i):
        # El tiempo lo cambias en threading
        if i==0:
            self.statusBar.config(text="Enviando mensaje... ")
            time.sleep(delay)               #Tiempo de 1s
            self.statusBar.config(text="")  #Luego de un 1 segundo lo
borra
        elif i==1:
            self.statusBar.config(text="Recibiendo mensaje... ")
```

```
            time.sleep(delay)               #Tiempo de 1s
            self.statusBar.config(text="")  #Luego de un 1segundo lo
borra

    def decodifica(self):
        #Abre ventana nueva
        self.codigo = tk.Toplevel(self.master)

        self.codigo.title("Decodificador")
        self.codigo.geometry("+100+100")
        self.codigo.resizable(0, 0)
        self.codigo.iconbitmap('icon_chat_sha.ico')
        self.codigo.config(bg="#4c4c4c")
        self.lineas_archivo = []

        #----------subventana frames---------------------------------
        self.frm11=tk.LabelFrame(self.codigo,text= "Ingreso de
KEY",bg="#4c4c4c",fg="#ffffff")
        self.frm22 = tk.LabelFrame(self.codigo,text="Mensaje
decodificado",bg="#4c4c4c",fg="#ffffff")
        self.frm33 = tk.LabelFrame(self.codigo, text="Enviar
mensaje",bg="#4c4c4c",fg="#ffffff")
        self.frm11.pack(padx=5, pady=5,anchor=tk.W)
        self.frm22.pack(padx=5, pady=5, fill='y', expand=True)
        self.frm33.pack(padx=5, pady=5,anchor=tk.W)


        #-------------------frm subventana1-----------------------------
------
        #-------------------frame 1--------------------------------------
---------
        self.lblText1 = tk.Label(self.frm11, text="Ingrese la
key:",bg="#4c4c4c",fg="#ffffff")
        self.inText1 = tk.Entry(self.frm11, width=60)
        self.lblText1.grid(row=0, column=0, padx=5, pady=5)
        self.inText1.grid(row=0, column=1, padx=5, pady=5)

        # ----------------------- FRAME 2 ---------------------------
        self.txtChat1 = ScrolledText(self.frm22, height=20, width=40,
wrap=tk.WORD)
        self.txtChat1.config(state='disabled')
        self.txtChat1.grid(row=0, column=0, columnspan=3, padx=5, pady=5)

        #-------------------------FRAME 3-------------------------------
------------
        self.lblText2 = tk.Label(self.frm33,
text="Texto:",bg="#4c4c4c",fg="#ffffff")
        self.inText2 = tk.Entry(self.frm33, width=45)
```

```
        self.btn_guardar = tk.Button(self.frm33, text="Guardar",
width=16, command=self.boton_guardar,bg="#4c4c4c",fg="#ffffff")

        self.inText2.bind("<Return>", self.enviar_mensaje_key) #Tecla
ENTER para enviar mensaje
        self.inText2.bind("<Key>", self.actualizandoStatusBar) #detectar
caracteres para "Escribiendo mensaje..."

        self.btnSend1 = tk.Button(self.frm33, text="Enviar",
width=12,command=lambda:
self.enviar_mensaje_key(None),bg="#4c4c4c",fg="#ffffff")


        self.lblText2.grid(row=0, column=0, padx=5, pady=5)
        self.inText2.grid(row=0, column=1, padx=5, pady=5)
        self.btnSend1.grid(row=0, column=2, padx=5, pady=5)
        self.btn_guardar.grid(row=1, column=2, padx=5, pady=5)

    def enviar_mensaje_key(self,event):
        #agarra la llave que se puso en el entry para decodificar

        self.KEY_DEC=self.inText1.get()
        fernet_dec = Fernet(self.KEY_DEC)
        #agarra  el dato a desencriptar
        mensaje_cifrar=self.inText2.get()
        self.txtChat1.config(state='normal', foreground='#53b1f1')
        #proceso de desencriptar
        mensaje_dec = fernet_dec.decrypt(mensaje_cifrar).decode()
        #fecha
        fecha=f"{datetime.strftime(datetime.now(),'%H'+':'+'%M')}"
        hora = int(fecha[:2])
        minutos = int(fecha[-2:])
        if hora>12:
            codificado=str(mensaje_dec)+f"\n {hora-12:02d}:{minutos:02d}
p.m.\n"

        else:
            codificado=str(mensaje_dec)+f"\n {hora:02d}:{minutos:02d}
a.m.\n"

        self.txtChat1.insert(tk.INSERT,codificado)

        self.txtChat1.yview(tk.END)
        self.lineas_archivo.append(codificado)
        self.txtChat1.config(state='disabled')
        self.inText2.delete(0, 'end')

    def normas(self):
        fileObject = open("logo.txt", "r")
```

```
        data = fileObject.read()
        print(data)
        linea=data
        self.txtChat.config(state='normal')
        self.txtChat.insert(tk.INSERT,linea,'verde')
        fileObject = open("mensaje.txt", "r")
        data = fileObject.read()
        linea=data
        self.txtChat.insert(tk.INSERT,linea,'verde')
        self.txtChat.yview(tk.END)
        self.txtChat.config(state='disabled')


def main():
    #Crear servidor
    if len(sys.argv)==1:
        Servidor().run()
    #Crear cliente
    if len(sys.argv)==2:
        cliente = ClienteGUI(sys.argv[1])
        cliente.master.mainloop()

if __name__=="__main__":
    main()
```

BIBLIOGRAFIA

Ferrara, Darla (2020)*¿Cuáles son los tipos de comunicación de datos?* Recuperado 30 de Noviembre de 2022.https://techlandia.com/cuales-son-tipos-comunicacion-datos-lista_447998/

Helmut (2020). *Protocolos de comunicación*. Recuperado 30 de noviembre de 2022.https://www.lifeder.com/protocolos-de-comunicacion/

Flores, Frankier (2020) Qué es Visual Studio Code y qué ventajas ofrece.Recuperado 30 de Noviembre de 2022.https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

Olga Weis (2022) *Comunicación Serie.* Recuperado 30 de Noviembre de 2022https://www.serial-port-monitor.org/es/articles/serial-communication/

Peña,Eric(2020) UART: A Hardware Communication
Protocol Understanding Universal Asynchronous Receiver/Transmitter. Recuperado 30 de noviembre de 2022 https://www.analog.com/media/en/analog-dialogue/volume-54/number-4/uart-a-hardware-communication-protocol.pdf

Rohde & Schwarz. (2022). *Entendiendo el UART*. Recuperado 30 de Noviembre de 2022, dehttps://www.rohde-schwarz.com/lat/productos/prueba-y- medicion/osciloscopios/educational-content/entendiendo-el-uart_254524.html#:~:text=UART%20significa%20transmisor%2Dreceptor%20as%C3%ADncrono,y%20recibir%20en%20ambas%20direcciones

Prudente, Jorge. (2019, 26 de noviembre). *Envío y recepción de los datos*. Recuperado 30 de Noviembre de 2022, de http://jorgearturoprudenteramirez.over-blog.com/2019/11/6.6- envio-y-recepcion-de-los-datos.html

Python. (2022). *threading — Paralelismo basado en hilos*. Recuperado 30 de Noviembre de2022, de 2022 https://docs.python.org/es/3.10/library/threading.html

Python. (2022). *Widget de texto desplazado*. Recuperado 30 de Noviembre de 2022, dehttps://docs.python.org/es/3/library/tkinter.scrolledtext.html

Keepcoding. (2022, 14 de marzo). *¿Qué es un Socket?.* Recuperado 30 de Noviembre de2022, de https://keepcoding.io/blog/que-es-un-socket/

IONOS. (2018, 15 de agosto). *¿Qué es Ethernet (IEEE 802.3)?.* Recuperado 30 de Noviembre de 2022, de https://www.ionos.es/digitalguide/servidores/know-how/ethernet-ieee-8023/#:~:text=Ethernet%20designa%20a%20una%20tecnolog%C3%ADa,se%20crea%20mediante%20conexiones%20Ethernet

Cisco. (2022). *What Is Wi-Fi?.* Recuperado 30 de Noviembre de 2022, de https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html

Tic.Portal. (2022, 14 de marzo). *¿Qué es un servidor y para qué sirve?.* Recuperado 30 de Noviembre de 2022, de https://www.ticportal.es/glosario-tic/servidores