Nicolás Navarro Garrido
Desarrollo de Software para Syslab

Documentación de Software “Menú Principal de Mensajería Estándar HL7”

El software fue desarrollado con la finalidad de abordar todas las problemáticas encontradas a través del desarrollo del software, estructurado como un menú simplificado, orientado a un usuario diario, se puede encontrar funciones como, verificar la apertura de un puerto, enviar mensajes a un receptor específico, convertir en servidor el equipo y escuchar, almacenar mensajes enviados, entre otras funcionalidades.

Instalación de Python:

Para instalar python 3 en el equipo a utilizar, este debe de ser descargado desde su sitio web oficial (https://www.python.org/downloads/), asegurándose de buscar la versión más reciente, luego de descargar el instalador, este se debe de ejecutar haciendo click sobre él.

Se deben de seguir las instrucciones al pie de la letra, tal como lo mantiene estipulado la pestaña del instalador, antes de la instalación final, se debe seleccionar la opción “Add Python 3.x to PATH”, esta opción sirve para agregarle bibliotecas que no posee por defecto python de una manera mucho más simple y también para poder actualizarlo constantemente.

Para ejecutar el código del servidor, simplemente se debe de hacer click sobre él y aparecerá una ventana tipo consola de comandos, donde se verá la información recibida y relevante.

Funciones Desarrolladas:

- ***manejar_interrupcion(signum, frame):*** Maneja la señal de interrupción (Ctrl+C) y establece la bandera Apagar_Server en True, esta función fue desarrollada con la finalidad de apagar el servidor de manera manual y no solo cerrando el programa.

- ***generar_nombre_archivo():*** Genera un nombre único para los archivos basado en la fecha, hora, minutos, segundos y milisegundos en el cual se solicita la función dentro del programa.

- ***recibir_archivo(conexion, directorio):*** Recibe mensajes HL7 desde un cliente, almacena el mensaje en un archivo y envía un ACK al cliente, esta función recibe como parámetro una conexión socket y un directorio donde se almacenarán los mensajes.

- ***iniciar_servidor(puerto, directorio):*** Inicia el servidor, es decir, abre el puerto correspondiente y se mantiene en un sistema de “escucha” constante hasta que logra

recibir una conexión desde un equipo específico, este al recibir la conexión pasa a ejecutar las funciones previamente mencionadas, tomando en cuenta la prioridad de cada una de estas y el momento adecuado en donde se necesita la información entregada por cada una de ellas.

- ***enviar_archivo(Dip, Puerto, Ruta):*** Inicia una conexión con el receptor, a través de la Ip y el puerto específico, posteriormente a través de la librería "OS", busca la ruta del archivo seleccionado y genera un envío de información a través de una conexión socket, entregando un mensaje de envío exitoso o el error que provoca que no se pueda enviar.

Variables Globales:

- Apagar_Server (boolean): Indica si el servidor debe detenerse, esta se inicializa en un valor "False", cuando se solicita detener el servidor, esta variable toma el valor "True".

Bibliotecas:

· ***Socket:*** Esta biblioteca es utilizada para acceder a las conexiones de los puertos de datos, siguiendo el estándar del modelo OSI.

· ***Signal:*** Biblioteca necesaria para recibir señales entrantes al dispositivo, utilizada específicamente para finalizar conexión del servidor.

· ***Os:*** Utilizada para acceder a funciones naturales del sistema operativo, en este caso para la lectura y validación de la existencia de directorios y el almacenaje de los mensajes.

· ***From datetime import datetime:*** Esta biblioteca es utilizada para extraer información del reloj interno del procesador, por ejemplo, fecha, hora, minutos, segundos, entre otros. Se usa para dar nombre a los archivos a guardar.

·

· ***Configparser:*** Es utilizada para leer archivos con extensión .ini y acceder a los datos relevantes dentro de ellos, en este caso el puerto y el directorio asignado a la mensajería.

Nicolás Navarro Garrido
Desarrollo de Software para Syslab

Código Fuente del Servidor:

```
import socket #Biblioteca para Trabajar con los Puertos de Conexión

import signal #Biblioteca para Realizar acciones con las señales entrantes al dispositivo

import os #Biblioteca para Acceder a Funciones Naturales del Sistema Operativo

from datetime import datetime #Biblioteca para Extraer datos Temporales del Reloj

import configparser #Biblioteca para Leer Archivos de Extensión .Ini


# Menú principal

aux = -1

while aux != 0:

#Menú donde se muestran las diferentes funcionalidades del sistema.

    print("\nOpción 1: Validar Puerto Abierto/Cerrado")

    print("Opción 2: Verificar Puertos Abiertos y Disponibles")

    print("Opción 3: Enviar Mensaje por Puerto Específico")

    print("Opción 4: Convertir Máquina en Servidor")

    print("Opción 5: Leer Mensajes HL7 ORU")

    print("Opción 0: Finalizar el Programa")

#Selección de la opción que se necesita utilizar.

    aux = int(input("\n Ingrese su Opcion--> "))


    if aux == 1:

        # Backend de Opción 1

        # Validador de Puerto Abierto/Cerrado

        print("--------------------Validar Puerto--------------------")

        DIp = input("Ingrese Dirección IPv4 del equipo--> ")

        Puerto = int(input("Ingrese Puerto de Interés--> "))
```

```
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)

        try:
            sock.connect((DIp, Puerto))
            print("El Puerto:", Puerto, " en la Dirección IPv4:", DIp, " está Abierto")
        except socket.error:
            print("El Puerto:", Puerto, " en la Dirección IPv4:", DIp, " está Cerrado")
        finally:
            sock.close()

    elif aux == 2:
        # Función para Verificar los Puertos Abiertos y Disponibles
        RangoI = int(input("Ingrese Rango Inferior para Búsqueda de Puertos Abiertos--> "))
        RangoS = int(input("Ingrese Rango Superior para Búsqueda de Puertos Abiertos--> "))
        Ip = input("Ingrese Dirección IPv4 del equipo--> ")
        ogR = range(RangoI, RangoS)
        Puertos_Disp = []

        for puerto in ogR:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.settimeout(1)
            resultado = sock.connect_ex((Ip, puerto))

            if resultado == 0:
                Puertos_Disp.append(puerto)
```

```
            sock.close()


        if Puertos_Disp:
            print("Puertos abiertos en", Ip, "-->", Puertos_Disp)
        else:
            print("No se encontraron puertos abiertos en ", Ip)


    elif aux == 3:
        # Conexion por Puerto Especifico y Envio de Informacion
        DIp = input("Ingrese Dirección IPv4 del Servidor--> ")
        Puerto = int(input("Ingrese Puerto para Conexión--> "))
        Variante = input("¿Desea Enviar un Archivo? (SI/NO)--> ").upper()


        if Variante == "SI":
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as cliente:
                try:
                    cliente.connect((DIp, Puerto))
                    MensajeA = "ARCHIVO"
                    cliente.sendall(MensajeA.encode('utf-8'))
                except ConnectionRefusedError:
                    print("Error: No se pudo establecer conexión con el servidor.")


            print("Datos para enviar el archivo")
            Ruta = input("Ingrese Ruta del Archivo a Enviar--> ")


            def enviar_archivo(DIp, Puerto, Ruta):
```

```
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as cliente:
                try:
                    cliente.connect((DIp, Puerto))
                    print("Conexión establecida con el servidor de Ip", DIp, "en el Puerto", Puerto)

                    # Enviar nombre del archivo al servidor
                    Nombre_A = os.path.basename(Ruta)
                    cliente.sendall(Nombre_A.encode('utf-8'))

                    # Enviar el contenido del archivo al servidor
                    with open(Ruta, 'rb') as archivo:
                        for datos in iter(lambda: archivo.read(1024), b''):
                            cliente.sendall(datos)

                    print("Archivo", Nombre_A, " enviado exitosamente.")

                except ConnectionRefusedError:
                    print("Error: No se pudo establecer conexión con el servidor.")

            enviar_archivo(DIp, Puerto, Ruta)

        elif Variante == "NO":
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as cliente:
                try:
                    cliente.connect((DIp, Puerto))
                    Menaux = "MENSAJE"
```

```
                cliente.sendall(Menaux.encode('utf-8'))
            except ConnectionRefusedError:
                print("Error: No se pudo establecer conexión con el servidor.")
        time.sleep(2)

        Mensaje = input("Ingrese su Mensaje a Enviar--> ")
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as cliente:
            try:
                cliente.connect((DIp, Puerto))
                print("Conexión establecida con la Ip,", DIp, " y el Puerto", Puerto)

                # Enviar datos al servidor
                cliente.sendall(Mensaje.encode('utf-8'))
                print("---------Mensaje Enviado Exitosamente---------")
            except ConnectionRefusedError:
                print("Error: No se pudo establecer conexión con el servidor.")
    else:
        print("\nOpción no Válida")

elif aux == 4:
    # Backend para Iniciar Servidor y Escuchar un Puerto Especifico
    # Variable global para indicar si el servidor debe detenerse
    Apagar_Server = False

    def manejar_interrupcion(signum, frame):
        global Apagar_Server
```

```
        print("\n Recibida Señal de Detención. Deteniendo el Servidor...")
        Apagar_Server = True

    def generar_nombre_archivo():
        # Generar un nombre único basado en la fecha y hora actual
        fecha_hora_actual = datetime.now().strftime("%Y%m%d%H%M%S")
        nombre_archivo = f"HL7TESTEO_{fecha_hora_actual}"
        return nombre_archivo

    def recibir_archivo(conexion, directorio):
        datos = conexion.recv(4096).decode('latin-1')
        if not datos:
            print("\n Conexion cerrada")
            return

        # Recibir el nombre del archivo
        nombre_archivo = generar_nombre_archivo() + ".res"
        print("\n Archivo en Transmision: ", nombre_archivo)
        if not nombre_archivo:
            return

        # Crear un directorio para cada archivo
        ruta_directorio = os.path.join(directorio).replace('\\', '/')

        print("\n Archivo por Guardar en: ", ruta_directorio)
```

```
        os.makedirs(ruta_directorio, exist_ok=True)

        # Crear y escribir el archivo en el directorio
        ruta_archivo = os.path.join(ruta_directorio, nombre_archivo)
        print("\n Direccion del Archivo a Guardar: ", ruta_archivo)
        print("\n Datos Almacenados en el Archivo", datos)

        # Abrir el archivo y escribir datos
        text_file = open(ruta_archivo, "w")
        n = text_file.write(datos)
        text_file.close()

        Mensaje = "Archivo Recibido Correctamente"
        conexion.sendall(Mensaje.encode('latin-1'))
        print(Mensaje, ":", nombre_archivo)

    def iniciar_servidor(puerto, directorio):
        global Apagar_Server

        # Configurar el manejador de señales para Ctrl+C
        signal.signal(signal.SIGINT, manejar_interrupcion)

        # Crear un socket IPv4 y TCP
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as servidor:
            # Enlazar el socket a la dirección y puerto especificados
            servidor.bind(('0.0.0.0', puerto))
```

```
            print("Servidor Escuchando el Puerto", puerto)


            while not Apagar_Server:
                # Establecer el servidor en modo de escucha
                servidor.listen()
                try:
                    # Esperar a que llegue una conexión
                    conexion, direccion_cliente = servidor.accept()
                    print("Conexión Establecida desde", direccion_cliente)


                    # Iniciar la recepción de archivos en el directorio especificado
                    recibir_archivo(conexion, directorio)
                except socket.error as e:
                    print("Error en la Conexion: ", e)


    if __name__ == "__main__":
        Parametros = configparser.ConfigParser()
        Parametros.read('Config.ini')


        Puerto = int(Parametros.get('Servidor', 'Puerto'))
        Directorio = Parametros.get('Servidor', 'Directorio')
        iniciar_servidor(Puerto, Directorio)


elif aux == 5:
    # Función para Leer Mensajes ORU Almacenados en Archivos
    ruta = input("Ingrese Ruta del Archivo Almacenado--> ")
```

```
        mensaje_hl7 = leer_archivo_hl7(ruta)



        if mensaje_hl7:

            # Procesa el mensaje HL7 como cadena de texto

            Lectura_Hl7(mensaje_hl7)


    elif aux == 0:

        # Backend para Finalizar el software

        print("\nFin del Programa")

        print("-----------------------------------------------")


    else:

        # Llega aquí cuando la opción seleccionada no es válida

        print("\nOpción no Válida")

        print("-----------------------------------------------")
```