**Estrategia de Resolución**

**Servidor**

El servidor está desarrollado en C y utiliza la biblioteca de sockets Winsock2 para manejar la comunicación con los clientes. La estructura general del servidor incluye la inicialización de WinSock, la creación y configuración de un socket, la vinculación del socket a un puerto, y la espera de conexiones entrantes. Cada conexión entrante es manejada en una función dedicada que procesa la solicitud del cliente y envía la respuesta correspondiente.

**1. Detalles de Implementación**

**1.1. Inicialización y Configuración del Servidor**

El servidor comienza con la inicialización de WinSock mediante WSAStartup. Esto prepara el entorno para la creación y uso de sockets en Windows. Luego, se crea un socket con socket (AF\_INET, SOCK\_STREAM, 0) configurado para utilizar el protocolo TCP/IP.

El socket es vinculado a una dirección IP y un puerto mediante la función bind (). En este caso, se utiliza INADDR\_ANY para permitir conexiones en cualquier interfaz de red disponible y se asigna el puerto 8080. Después de la vinculación, el servidor se pone en estado de escucha con listen (), permitiendo que hasta 3 conexiones entrantes sean puestas en cola.

**1.2 Manejo de Conexiones Entrantes**

Una vez que el servidor está en estado de escucha, utiliza accept () para aceptar conexiones entrantes de los clientes. Cada conexión es manejada en la función manejarCliente (), que recibe el socket del cliente como parámetro.

**1.3 Procesamiento de Solicitudes del Cliente**

Dentro de la función manejarCliente (), se utiliza recv () para recibir el mensaje del cliente. Este mensaje es luego procesado para determinar si se trata de una solicitud para generar un nombre de usuario o una contraseña.

* **Generación de Nombre de Usuario**: Si el cliente solicita la generación de un nombre de usuario, el servidor valida que la longitud solicitada esté entre 5 y 15 caracteres. Luego, se genera un nombre de usuario alternando entre vocales y consonantes de manera aleatoria. El nombre generado es enviado de vuelta al cliente.
* **Generación de Contraseña**: Si el cliente solicita la generación de una contraseña, el servidor valida que la longitud solicitada esté entre 8 y 50 caracteres. La contraseña generada es alfanumérica e incluye tanto mayúsculas como minúsculas. La contraseña generada es enviada de vuelta al cliente.
* **Validación y Manejo de Errores**: Si la longitud solicitada no cumple con los requisitos o si se recibe un comando no reconocido, el servidor envía un mensaje de error al cliente.

**1.4. Respuesta y Cierre de Conexión**

Después de procesar la solicitud y enviar la respuesta al cliente, el servidor cierra la conexión con closesocket (). El servidor continúa escuchando nuevas conexiones y repite el proceso para cada cliente que se conecta.

**2. Estrategias de Validación y Pruebas**

Se realizaron pruebas exhaustivas para asegurar que el servidor maneje correctamente las solicitudes de generación de nombres de usuario y contraseñas. Las pruebas incluyeron la verificación de:

* La correcta generación de nombres de usuario y contraseñas con longitudes válidas.
* El manejo adecuado de entradas inválidas, como longitudes fuera de rango o comandos no reconocidos.
* La capacidad del servidor para manejar múltiples conexiones de clientes, incluyendo un segundo cliente desarrollado en Python.

**Cliente en C**

**Estructura General del Cliente**

El cliente se encarga de la interacción con el usuario, recopilando la entrada para determinar el tipo de solicitud (nombre de usuario o contraseña) y la longitud deseada. Luego, el cliente establece una conexión con el servidor, envía la solicitud y muestra la respuesta recibida.

**1. Detalles de Implementación**

**1.1. Inicialización y Configuración del Cliente**

El cliente comienza con la inicialización de WinSock utilizando WSAStartup, similar al servidor. A continuación, se crea un socket utilizando socket(AF\_INET, SOCK\_STREAM, 0) para la comunicación TCP/IP.

El cliente se configura para conectarse a la dirección 127.0.0.1 (localhost) y al puerto 8080, que es donde el servidor está escuchando. Si la conexión falla, se maneja con un mensaje de error y el cliente se cierra.

**1.2. Interacción con el Usuario**

El cliente presenta un menú interactivo al usuario, permitiéndole seleccionar una de las tres opciones:

1. **Generar Nombre de Usuario**: El usuario especifica la longitud deseada, que debe estar entre 5 y 15 caracteres.
2. **Generar Contraseña**: El usuario especifica la longitud deseada, que debe estar entre 8 y 50 caracteres.
3. **Salir**: Esta opción permite al usuario salir del programa.

El cliente valida que la entrada del usuario sea numérica y esté dentro del rango esperado antes de enviar la solicitud al servidor.

**1.3. Comunicación con el Servidor**

Una vez que el usuario selecciona una opción y proporciona la longitud adecuada, el cliente envía un mensaje al servidor en el formato "NOMBRE X" o "CONTRASENA Y", donde X y Y representan la longitud deseada.

El cliente utiliza send () para enviar el mensaje y recv () para recibir la respuesta del servidor. La respuesta es entonces mostrada en la consola del usuario.

**1.4. Manejo de Errores**

Se implementaron varias estrategias para manejar errores:

* **Validación de Entrada**: El cliente verifica que las entradas sean numéricas y estén dentro de los rangos esperados. Si el usuario introduce un valor no numérico o fuera de rango, se muestra un mensaje de error y se le solicita que lo intente de nuevo.
* **Conexión al Servidor**: Si la conexión al servidor falla, el cliente muestra un mensaje de error adecuado.
* **Respuesta del Servidor**: Cualquier mensaje de error recibido del servidor es mostrado directamente al usuario.

**2. Estrategias de Validación y Pruebas**

Se realizaron pruebas para asegurar que el cliente maneje correctamente la interacción con el usuario y la comunicación con el servidor. Las pruebas incluyeron:

* Validación de la entrada del usuario para asegurar que sólo se acepten valores válidos.
* Verificación de que el cliente pueda enviar correctamente las solicitudes al servidor y recibir las respuestas esperadas.
* Pruebas con longitudes de nombres de usuario y contraseñas fuera de los rangos permitidos para asegurar que se muestren los mensajes de error adecuados.

#### Cliente Python Estructura General del Cliente

El cliente en Python sigue una estructura similar al cliente en C, pero aprovechando las características de Python, como la gestión automática de recursos y el manejo de excepciones. El cliente se encarga de recibir la entrada del usuario, establecer una conexión con el servidor, enviar la solicitud y mostrar la respuesta.

**1. Detalles de Implementación**

**1.1. Inicialización y Configuración del Cliente**

El cliente utiliza la función socket.socket() para crear un socket TCP/IP. La conexión se establece con el servidor en la dirección 127.0.0.1 (localhost) y el puerto 8080.

La conexión se gestiona dentro de un bloque with, lo que asegura que el socket se cierre automáticamente al final de la operación, incluso si ocurre un error.

**1.2. Interacción con el Usuario**

El cliente presenta un menú interactivo al usuario, que le permite seleccionar una de las tres opciones:

1. **Generar Nombre de Usuario**: El usuario especifica la longitud deseada, que debe estar entre 5 y 15 caracteres.
2. **Generar Contraseña**: El usuario especifica la longitud deseada, que debe estar entre 8 y 50 caracteres.
3. **Salir**: Esta opción permite al usuario salir del programa.

El cliente valida que la entrada del usuario sea numérica y esté dentro del rango esperado antes de enviar la solicitud al servidor.

**1.3. Comunicación con el Servidor**

Una vez que el usuario selecciona una opción y proporciona la longitud adecuada, el cliente envía un mensaje al servidor en el formato "NOMBRE X" o "CONTRASENA Y", donde X y Y representan la longitud deseada.

El cliente utiliza sock.sendall() para enviar el mensaje al servidor y sock.recv() para recibir la respuesta. La respuesta se decodifica de bytes a una cadena de texto y se muestra en la consola del usuario.

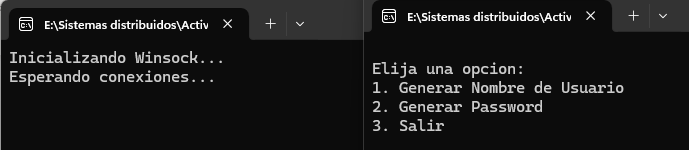
**1.4. Manejo de Errores**

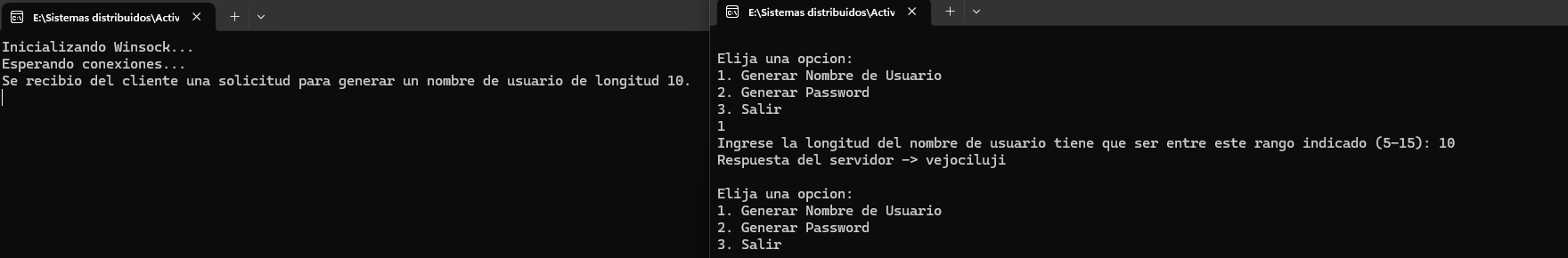
El cliente incluye estrategias para manejar errores comunes:

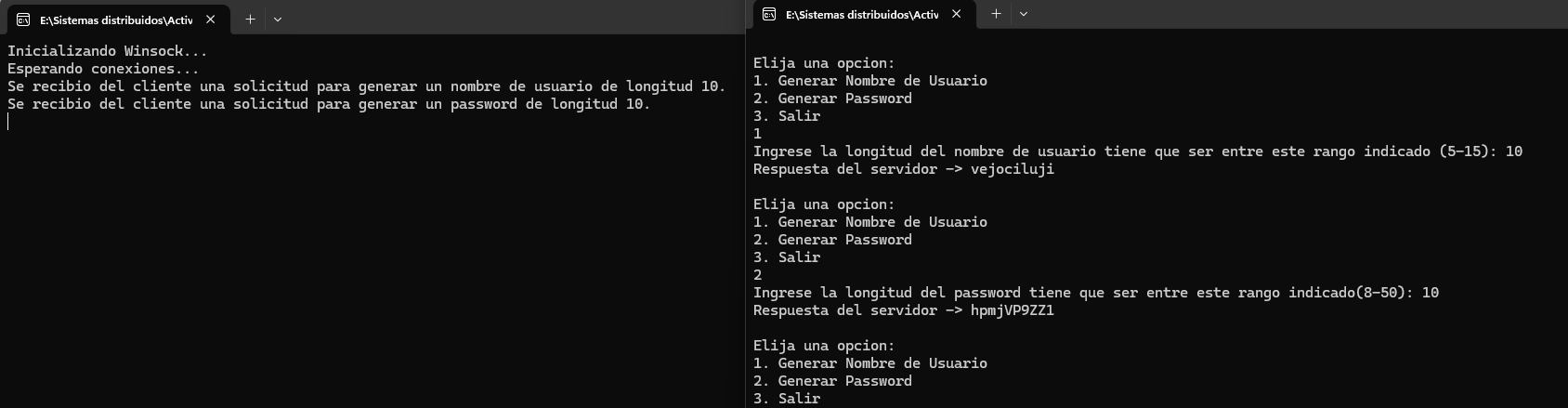
* **Validación de Entrada**: Se valida que la longitud ingresada por el usuario sea un número y que esté dentro del rango permitido. Si no es así, se muestra un mensaje de error.
* **Conexión al Servidor**: La conexión al servidor está envuelta en un bloque with, que garantiza que el socket se cierra correctamente incluso si ocurre un error durante la comunicación

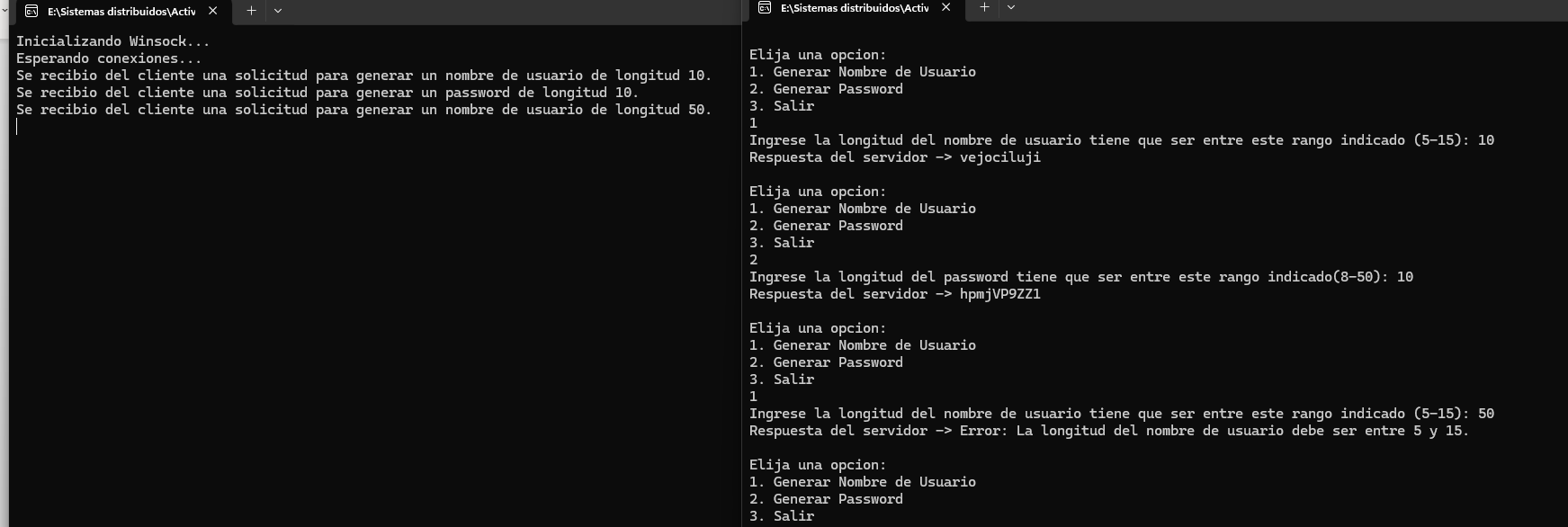
**Captura de Pruebas realizadas**

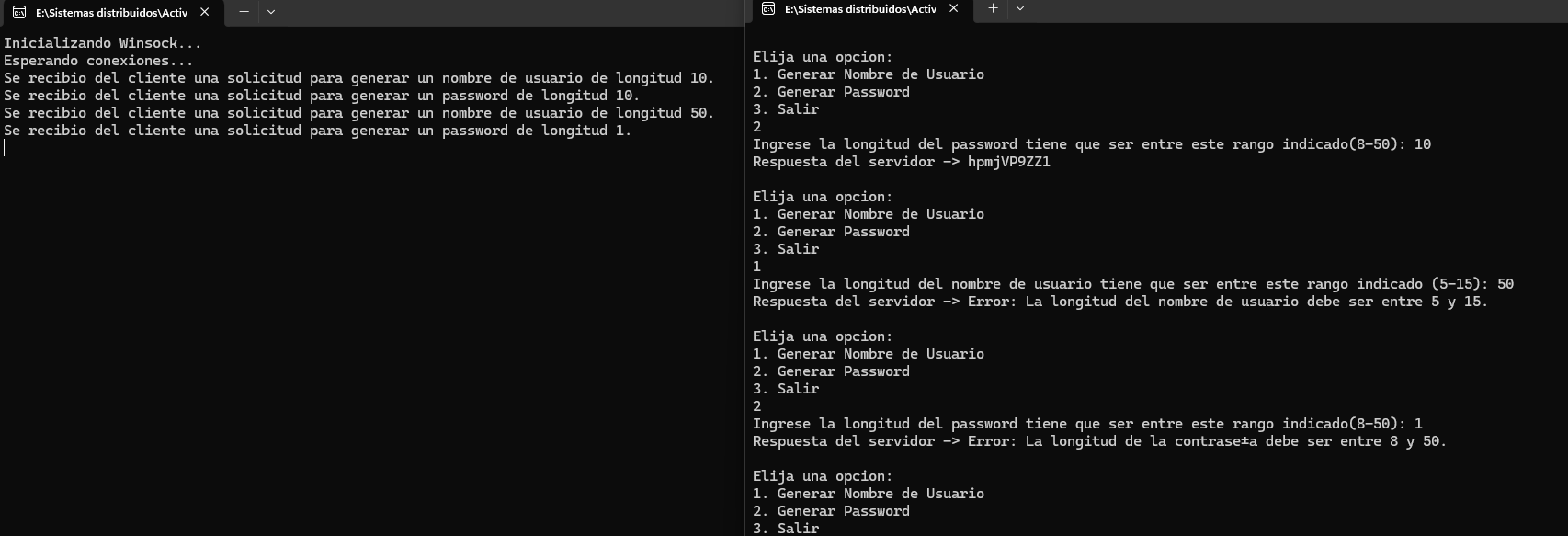
1. **Pruebas con cliente en C**

****

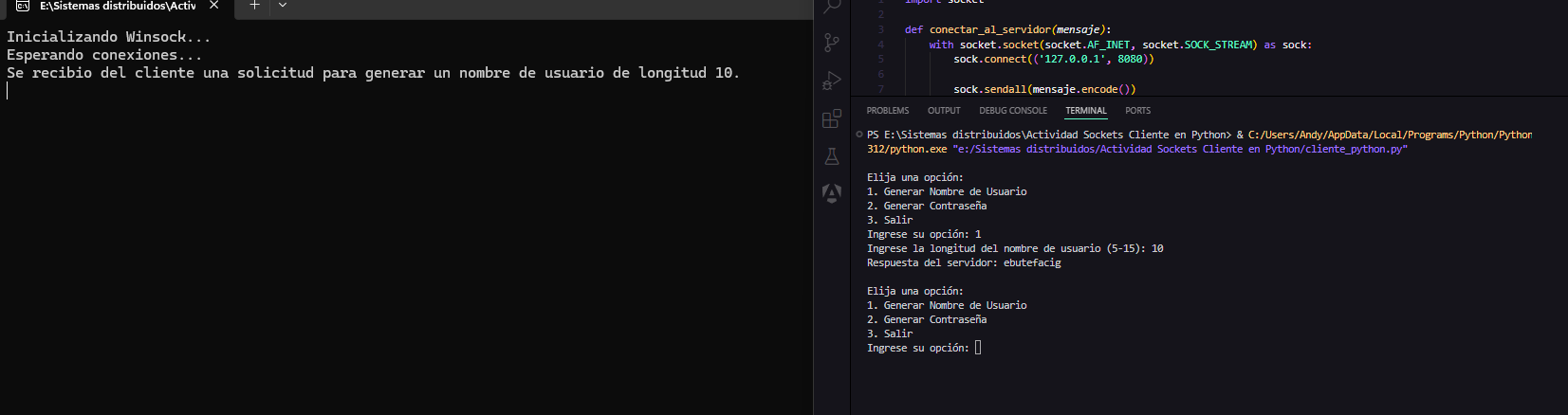
****

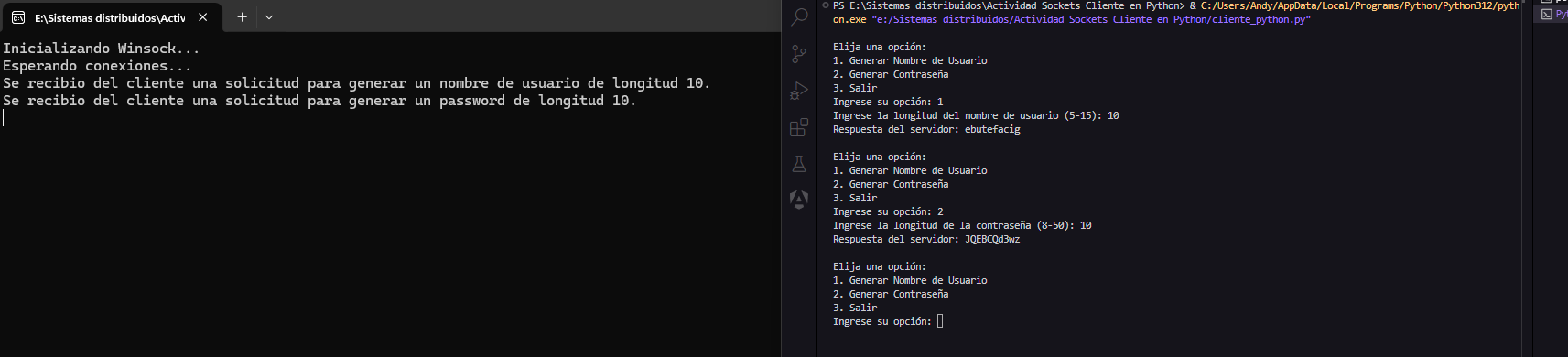
****

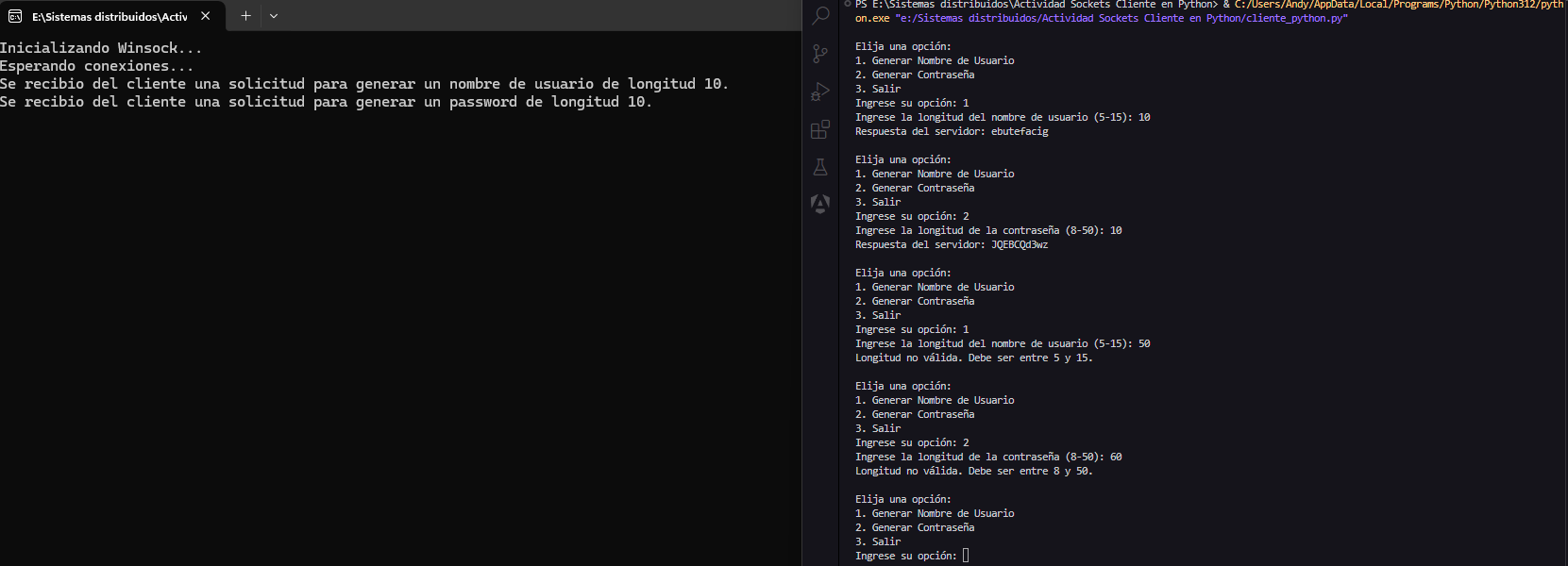
****

****

1. **Prueba con Cliente en Python**

****

****

****