
Universidad Nacional de Lanús

Departamento de Desarrollo Productivo y Tecnológico

Licenciatura en Sistemas

Desarrollo de Software en Sistemas Distribuidos

# "**Actividad Sockets**"

Docentes: Ing. Diego Andrés Azcurra
Lic. Marcos Amaro

Estudiante: Belén María Renda

DNI: 43321072

Fecha: 22/08/2024

# Introducción

En el marco de la asignatura Desarrollo de Software en Sistemas Distribuidos se nos planteó desarrollar un sistema de "sockets" que estaría compuestos por un servidor y dos clientes, donde uno de los clientes sería en el mismo lenguaje de programación que el servidor y el otro en un lenguaje de programación distinto, ambos generando una conexión con un cliente que iba a poder generar un usuario y/o una contraseña con ciertos criterios y verificaciones. El sistema puede validar tanto la longitud del usuario como de la contraseña.

En esta actividad se presenta la implementación de un servidor en el lenguaje C que utiliza sockets para comunicarse con ambos clientes. El servidor ofrece dos funcionalidades principales:

- Un generador de nombres de usuario
- Un generador de contraseñas

El generador de nombres de usuario crea cadenas que alternan entre vocales y consonantes, con una longitud de entre 5 y 15 caracteres.
El generador de contraseñas crea cadenas alfanuméricas con mayúsculas y minúsculas, con una longitud de entre 8 y 50 caracteres.

Asimismo, se desarrollaron dos clientes que se conectan al servidor. Estos permiten elegir entre generar un nombre de usuario o generar una contraseña enviando la longitud deseada (cumpliendo los parámetros), o salir mediante un menú de opciones. El primer cliente fue implementado en el lenguaje de programación C, mientras que el segundo cliente fue desarrollado en el lenguaje Python.
En este informe se describe la estrategia de resolución de la actividad, incluyendo la estructura del servidor y los clientes, la implementación de las funcionalidades y los resultados obtenidos.

**Link al repositorio en Github:** Actividad Sockets - Github

# Análisis del Problema

1. Desarrollar un servidor en lenguaje C utilizando sockets, implementando estas funcionalidades:

a. **Generador de nombres de usuario:** indicando la longitud del nombre de usuario (debe validar que no sea menor a 5 ni mayor a 15), el servidor generará la cadena correspondiente alternando entre vocales y consonantes. Es decir, si empezó por una vocal, el siguiente carácter será una consonante o viceversa. El servidor elegirá, también al azar, si empieza por vocal o consonante.

**b.** **Generador de contraseñas:** indicando la longitud de la contraseña (debe validar que sea mayor o igual a 8 y menor a 50), el servidor generará una cadena alfanumérica, incluyendo mayúsculas y minúsculas.

2. Desarrollar el cliente en lenguaje C, el cual se conectará al servidor, y por medio de un menú, podrá elegir entre generar un nombre de usuario o generar una contraseña, enviando la longitud deseada. Se debe mostrar la respuesta generada por el servidor (en caso de error de validación también).
3. Desarrollar un segundo cliente, con la misma funcionalidad que el primero, pero en un lenguaje diferente, por ejemplo: Kotlin, Go, Javascript (NodeJS), Python, etc.

El problema planteado requiere la implementación de un sistema de generación de credenciales de usuario que cumpla con ciertos requisitos. A continuación, se analizarán los desafíos y las dificultades que se presentan en este problema.

**Desafíos:**

1. **La generación de nombres de usuario:** Este desafío propone que el nombre de usuario debe cumplir con la restricción de longitud (entre 5 y 15 caracteres) y el deber alternar entre vocales y consonantes.
2. **La generación de contraseñas:** Este desafío propone que la contraseña debe cumplir con la restricción de longitud (entre 8 y 50 caracteres) y el deber incluir valores alfanuméricos aleatorios incluyendo minúsculas y mayúsculas.
3. **La conexión entre servidor y cliente:** Este desafío propone que el servidor debe ser capaz de comunicarse con los diferentes clientes a través de sockets, lo que requiere implementar una lógica de conexión y comunicación segura.
4. **Las validaciones de entradas:** Este desafío propone que el servidor debe validar las entradas de los clientes para garantizar que se cumplen los requisitos de longitud y formato para el nombre de usuario y la contraseña.

**Dificultades:**

1. **La generación de cadenas de nombres de usuario:** Esto propuso una gran dificultad ya que las cadenas no sólo debían cumplir con una longitud determinada, sino que se debía verificar si el carácter era una vocal o consonante para generar la alteración entre dichos.
2. **La seguridad de la conexión:** Esto propuso un cierto grado de dificultad debido a que la comunicación entre el servidor y los clientes debía ser efectiva y segura.
3. **La compatibilidad:** Esto propuso un alto grado de dificultad ya que el segundo cliente requería de otro lenguaje de programación diferente al servidor y teniendo otra estructura gramatical.

## Estructura de la Aplicación

**Servidor:**

**Generador de nombres de usuario:** Módulo encargado de generar nombres de usuarios aleatorios que cumplan con los requisitos de longitud y formato.

**Generador de contraseñas:** Módulo encargado de generar contraseñas aleatorias que cumplan con los requisitos de longitud y formato.
**Cuerpo "main":** Módulo responsable de tener todas las interacciones. Posee dentro el **Socket server** que es el módulo responsable de establecer la conexión con los clientes y recibir solicitudes para generar nombres de usuario y contraseñas. También posee los **Validadores** que son los módulos responsables de validar las entradas de los clientes para garantizar que se cumplan los requisitos de longitud y formato.

**Cliente en C:**

**Socket cliente:** Módulo encargado de establecer la conexión con el servidor y enviar solicitudes para generar nombres de usuario y contraseñas.

**Procesador de respuestas:** Módulo encargado de recibir y procesar las respuestas del servidor, mostrando los resultados al usuario.

**Interfaz del usuario:** Módulo encargado de mostrar el menú al usuario para elegir entre las distintas opciones, ya sea generar un nombre de usuario, generar una contraseña o salir.

**Cliente en Python:**

**Socket cliente:** Módulo encargado de establecer la conexión con el servidor y enviar solicitudes para generar nombres de usuario y contraseñas.

**Procesador de respuestas:** Módulo encargado de recibir y procesar las respuestas del servidor, mostrando los resultados al usuario.

**Interfaz del usuario:** Módulo encargado de mostrar el menú al usuario para elegir entre las distintas opciones, ya sea generar un nombre de usuario, generar una contraseña o salir.

**Flujo de la aplicación:**

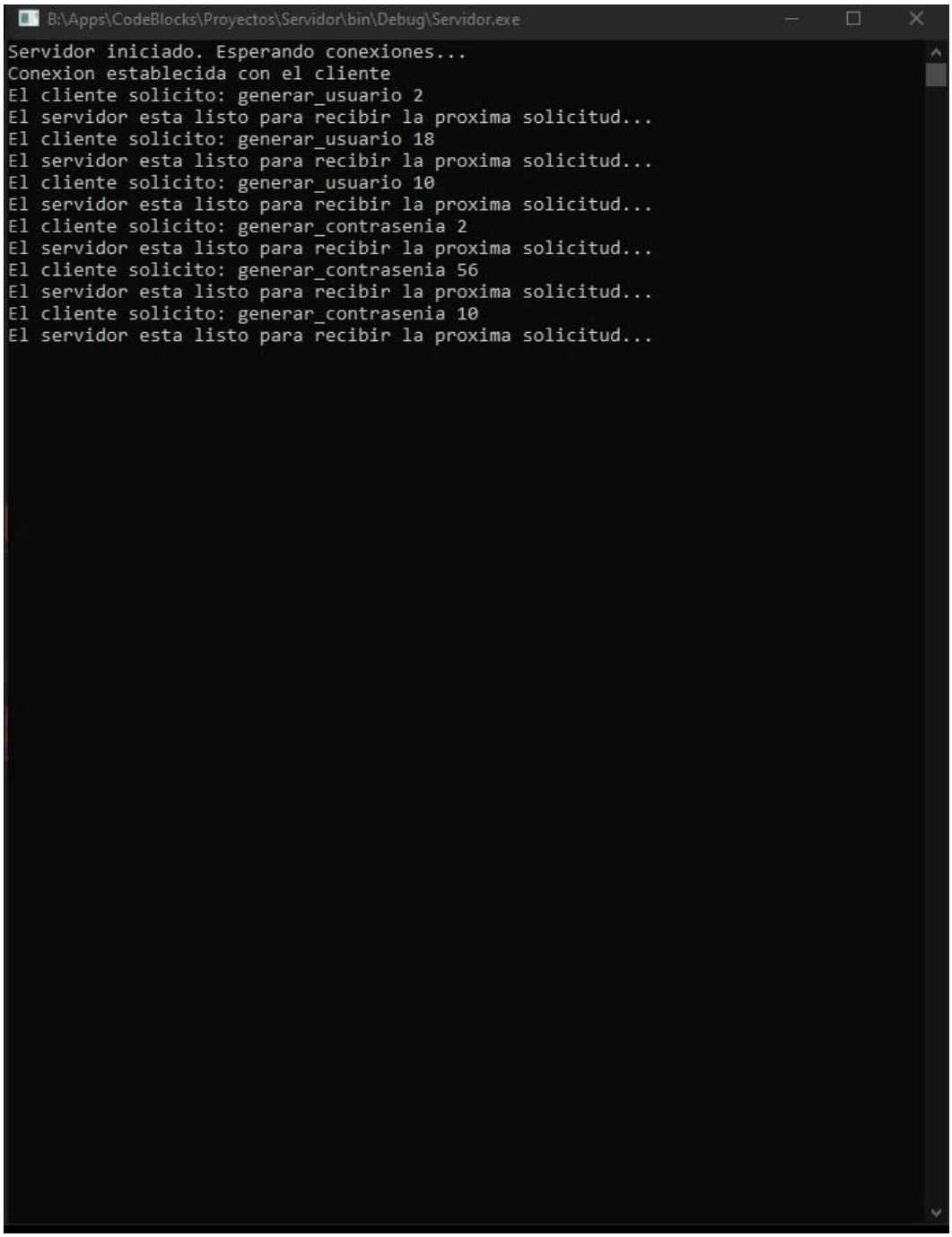
1. El servidor se inicia y aguarda conexión.
2. El usuario mediante el cliente (Ya sea en C o en Python) se conecta con el servidor.
3. El servidor le muestra el menú con opciones en la interfaz siendo este:
   "1. Generar nombre de usuario
   2. Generar contraseña
   3. Salir"
4. El usuario selecciona la opción de generar nombre de usuario o generar contraseña.
5. El servidor le pide la longitud deseada para el nombre de usuario o contraseña.
6. El cliente envía una solicitud al servidor con la longitud deseada para el nombre de usuario o contraseña.
7. El servidor valida la entrada del cliente y genera un nombre de usuario o contraseña aleatorio (que cumple con los requisitos de longitud y formato).
8. El servidor envía la respuesta al cliente.
9. El cliente procesa la respuesta y muestra el resultado por pantalla al usuario.

# Pruebas

Se hicieron pruebas para cada validación y caso posible, para mostrar los casos en que falla y que funciona exitosamente tanto con el cliente en C como en Python. A continuación dejo capturas mostrando ambos casos:

**Pruebas con el cliente en C:**

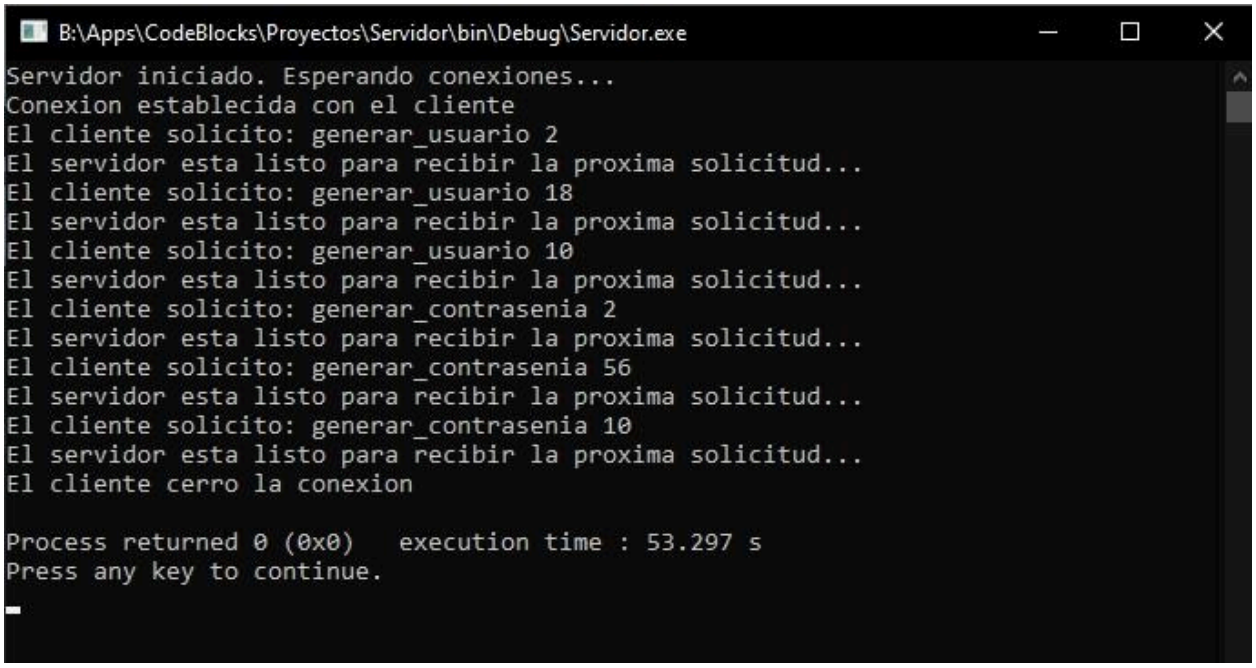
Servidor:



Cliente:

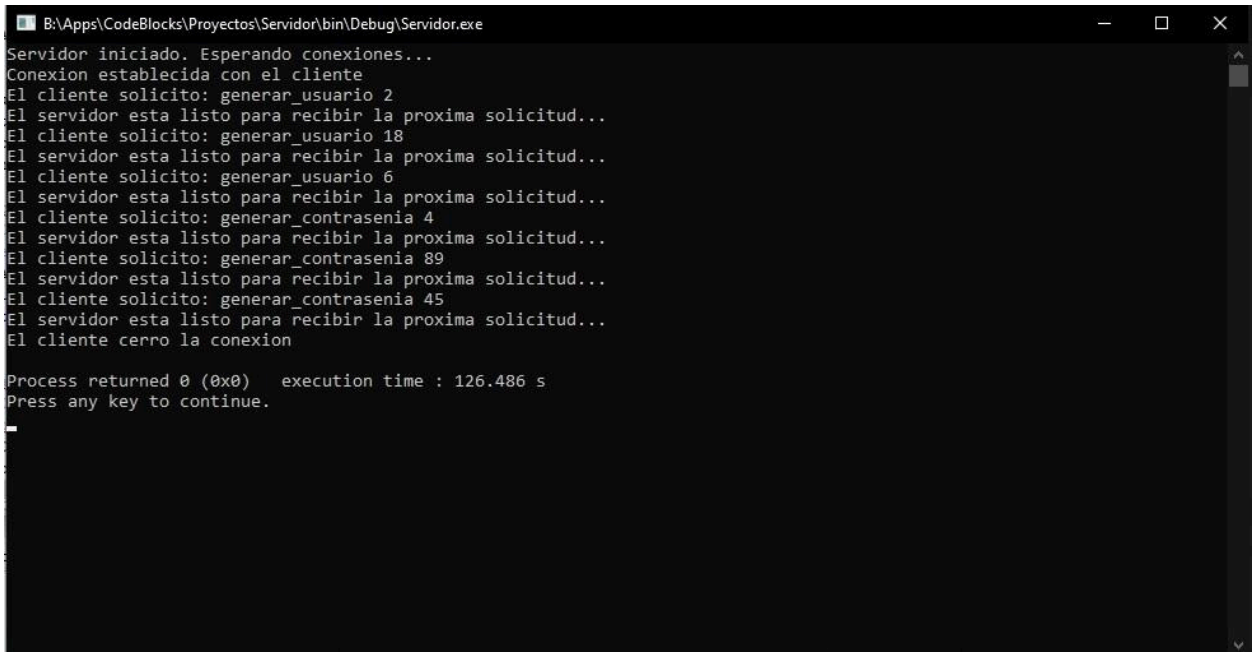

B:\Apps\CodeBlocks\Proyectos\Cliente\bin\Debug\Cliente.exe
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
1
Ingrese la longitud del nombre de usuario: 2
El servidor respondio: Error: La longitud del usuario debe ser entre 5 y 15 caracte
res
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
1
Ingrese la longitud del nombre de usuario: 18
El servidor respondio: Error: La longitud del usuario debe ser entre 5 y 15 caracte
res
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
1
Ingrese la longitud del nombre de usuario: 10
El servidor respondio: oxoVoNowes
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
2
Ingrese la longitud de la contrasenia: 2
El servidor respondio: Error: La longitud de la contrasenia debe ser entre 8 y 50 c
aracteres
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
2
Ingrese la longitud de la contrasenia: 56
El servidor respondio: Error: La longitud de la contrasenia debe ser entre 8 y 50 c
aracteres
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
2
Ingrese la longitud de la contrasenia: 10
El servidor respondio: t_dReOEWXo
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
3

Conexión finalizada del servidor (si el usuario eligió “3. Salir”):

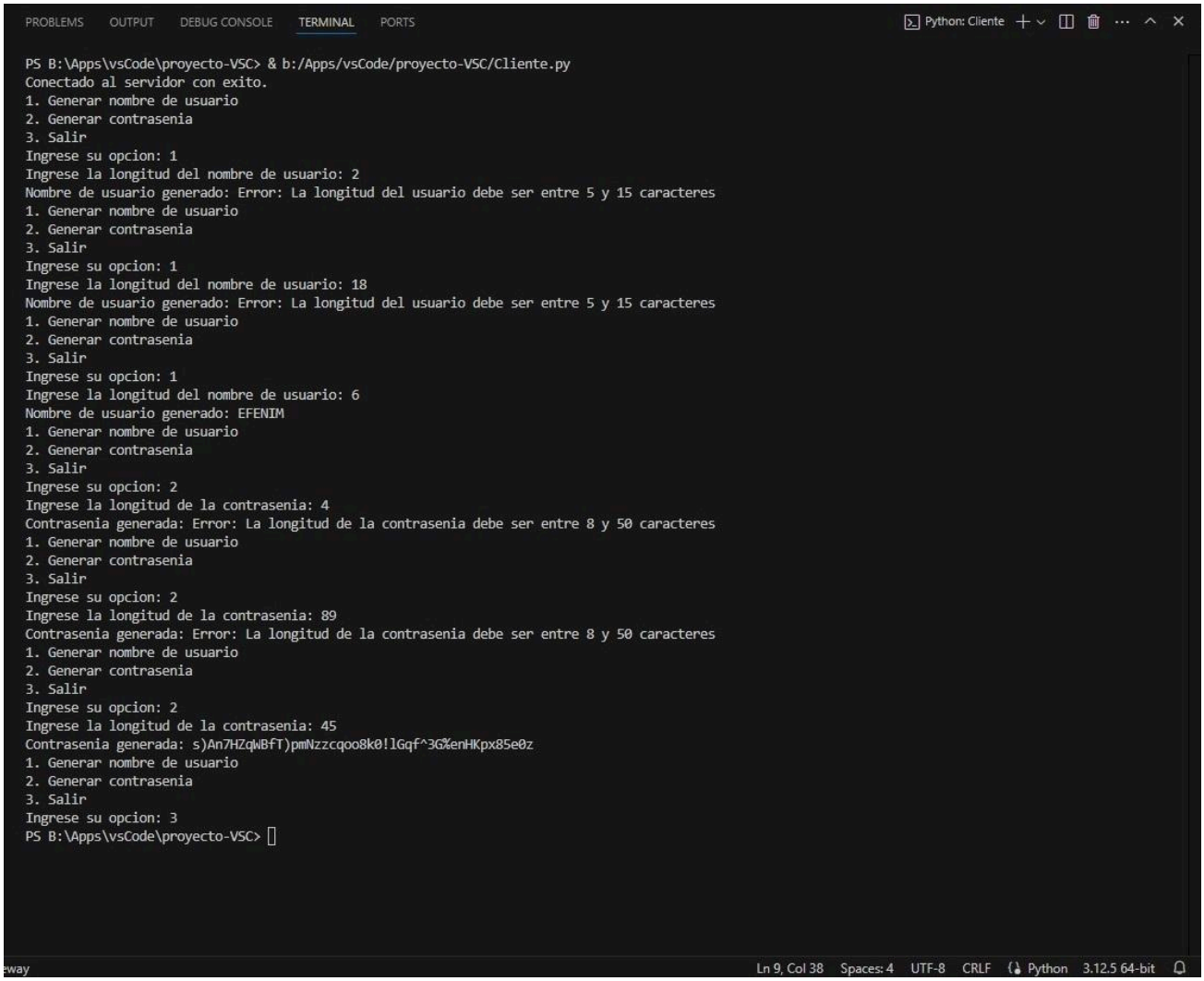



**Pruebas con el cliente en Python:**

Servidor:



Cliente:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python: Cliente
PS B:\Apps\vsCode\proyecto-VSC> & b:/Apps/vsCode/proyecto-VSC/Cliente.py
Conectado al servidor con exito.
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
Ingrese su opcion: 1
Ingrese la longitud del nombre de usuario: 2
Nombre de usuario generado: Error: La longitud del usuario debe ser entre 5 y 15 caracteres
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
Ingrese su opcion: 1
Ingrese la longitud del nombre de usuario: 18
Nombre de usuario generado: Error: La longitud del usuario debe ser entre 5 y 15 caracteres
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
Ingrese su opcion: 1
Ingrese la longitud del nombre de usuario: 6
Nombre de usuario generado: EFENIM
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
Ingrese su opcion: 2
Ingrese la longitud de la contrasenia: 4
Contrasenia generada: Error: La longitud de la contrasenia debe ser entre 8 y 50 caracteres
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
Ingrese su opcion: 2
Ingrese la longitud de la contrasenia: 89
Contrasenia generada: Error: La longitud de la contrasenia debe ser entre 8 y 50 caracteres
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
Ingrese su opcion: 2
Ingrese la longitud de la contrasenia: 45
Contrasenia generada: s)An7HZqWBfT)pmNzzcqoo8k0!lGqf^3G%enHKpx85e0z
1. Generar nombre de usuario
2. Generar contrasenia
3. Salir
Ingrese su opcion: 3
PS B:\Apps\vsCode\proyecto-VSC>
Ln 9, Col 38
Spaces: 4
UTF-8
CRLF
Python
3.12.5 64-bit