



Universidad Nacional de Lanús

**DESARROLLO DE SOFTWARE EN SISTEMAS
DISTRIBUIDOS**

Actividad Sockets

Departamento: Desarrollo Productivo y Tecnológico

Carrera: Licenciatura en Sistemas

Año: 2024

Cuatrimestre: 2° Cuatrimestre

Docentes:

Ing. Diego Andrés Azcurra

Lic. Marcos Amaro

Alumno: Franco Ariel Figueroa

DNI: 42.684.291

Índice

<u>1. Código fuente.....</u>	<u>3</u>
<u>2. Estrategia de resolución.....</u>	<u>3</u>
<u>2.1. Servidor.....</u>	<u>3</u>
<u>2.2. Clientes.....</u>	<u>4</u>
<u>3. Pruebas realizadas.....</u>	<u>6</u>
<u>3.1. Pruebas entre el servidor y el cliente C.....</u>	<u>6</u>
<u>3.2. Pruebas entre el servidor y el cliente Python.....</u>	<u>10</u>

1. Código Fuente

Enlace a repositorio público de Github: <https://github.com/Franco2900/Actividad-Sockets.git>

2. Estrategia de resolución

2.1. Servidor

El servidor esta hecho en C y esta compuesto por tres (3) archivos: mainServidor.c, Servidor.h y Servidor.c. El programa fue desarrollado en Windows, por lo cual usa la librería winsock2.h (librería exclusiva de Windows) para poder trabajar con sockets.

El archivo mainServidor.c es el archivo principal del servidor. Se encarga de controlar el orden de ejecución de las funciones del programa, llamando a las funciones declaradas en Servidor.h y definidas en Servidor.c. Una vez que se inicia la ejecución de mainServidor.c, este no va a parar de ejecutarse; la única forma de pararlo es cerrando manualmente el programa. Esto se debe a que el servidor siempre va a estar escuchando por posibles conexiones con clientes. Cuando termina de atender a un cliente, inmediatamente se vuelve a poner en modo escucha esperando por algún cliente nuevo. El servidor solo atiende un cliente a la vez.

El archivo Servidor.h contiene las declaraciones de las funciones que utiliza el archivo mainServidor.c. Las funciones se clasifican en tres (3) opciones distintas según su funcionalidad: funciones del socket (que sirven para crear el socket servidor, que se ponga en modo escucha, acepte clientes y envíe mensajes al cliente), las funciones del punto 1.A para generar el nombre de usuario y las funciones del punto 1.B para generar la contraseña.

El archivo Servidor.c contiene la definición lógica de las funciones declaradas en el archivo Servidor.h. A continuación se explican brevemente la lógica de cada una de sus funciones.

- SOCKET escucharConexiones(): Permite manejar caracteres especiales como palabras con tilde u otros, crea el socket servidor y lo pone en modo escucha en localhost:3000 (o lo que es lo mismo, 127.0.0.1:3000).
- SOCKET aceptarSocket(SOCKET socketServidor): El socket servidor acepta la conexión de un socket cliente.
- void enviarMensaje(char *mensaje, SOCKET socketCliente): Envía un mensaje de tipo string al socket cliente especificado.
- char* generarNombreDeUsuario(int tamaño): Crea un nombre de usuario alternando entre vocales y consonantes, de forma que si un carácter es una vocal el siguiente carácter es una consonante y viceversa. Si el nombre de usuario empieza con consonante o vocal es al azar. La longitud del nombre de usuario depende del tamaño indicado.
- void atenderGeneracionNombreDeUsuario(SOCKET socketCliente): Cuando el socket cliente ingresa un tamaño para el nombre de usuario, el socket servidor verifica que el tamaño sea entre cinco (5) y quince (15). Si la longitud es válida, le manda un mensaje al socket cliente que dice "Longitud valida" y después otro mensaje con el nombre de usuario generado; si la longitud no es válida, le manda un mensaje al socket cliente que dice "Longitud invalida" y espera a que el cliente vuelva a ingresar otra longitud para repetir el proceso hasta que ingrese una válida.
- char* generarContraseña(int tamaño): Crea una contraseña alfanumérica. No alterna entre vocales y consonantes porque no es solicitado por la consigna. Cada carácter es generado al

azar y puede ser una vocal, una consonante o un número. La longitud de la contraseña depende del tamaño indicado.

- `void atenderGeneracionContraseña(SOCKET socketCliente)`: Cuando el socket cliente ingresa un tamaño para la contraseña, el socket servidor verifica que el tamaño sea entre ocho (8) y cincuenta (50). Si la longitud es válida, le manda un mensaje al socket cliente que dice “Longitud valida” y después otro mensaje con la contraseña generada; si la longitud no es válida, le manda un mensaje al socket cliente que dice “Longitud invalida” y espera a que el cliente vuelva a ingresar otra longitud para repetir el proceso hasta que ingrese una válida.

2.2. Clientes

Fueron realizados dos (2) clientes para el servidor, un cliente en C y un cliente en Python.

El cliente C esta compuesto por tres (3) archivos: `mainCliente.c`, `Cliente.h` y `Cliente.c`. El programa fue desarrollado en Windows, por lo cual usa la librería `winsock2.h` (librería exclusiva de Windows) para poder trabajar con sockets.

El archivo `mainCliente.c` es el archivo principal del cliente. Se encarga de controlar el orden de ejecución de las funciones del programa, llamando a las funciones declaradas en `Cliente.h` y definidas en `Cliente.c`. Cuando se inicia el programa, el usuario tiene un menú en el cuál puede elegir entre tres (3) opciones: generar un nombre de usuario, generar una contraseña o salir (lo cual para la ejecución del cliente).

El archivo `Cliente.h` contiene las declaraciones de las funciones que utiliza el archivo `mainCliente.c`. Las funciones se clasifican en cuatro (4) opciones distintas según su funcionalidad: funciones del socket (que sirven para crear el socket cliente, que se conecte al servidor y envíe mensajes al servidor), la función del punto 1.A para generar el nombre de usuario, la función del punto 1.B para generar la contraseña y las funciones útiles (que sirven para controlar lo que ingresa el usuario).

El archivo `Cliente.c` contiene la definición lógica de las funciones declaradas en el archivo `Cliente.h`. A continuación se explican brevemente la lógica de cada una de sus funciones.

- `SOCKET conectarseAlServidor()`: Permite manejar caracteres especiales como palabras con tilde u otros, crea el socket cliente y lo conecta con el socket servidor en `localhost:3000` (o lo que es lo mismo, `127.0.0.1:3000`).
- `void enviarMensaje(char *mensaje, SOCKET socketCliente)`: Envía un mensaje de tipo string al socket cliente especificado.
- `int menu()`: Se encarga de la navegación del usuario a través del menú del cliente y se asegura que solo ingrese opciones válidas.
- `int ingresarTamano()`: Se asegura de que el cliente solo ingrese un número cuando debe ingresar el tamaño del nombre de usuario o la contraseña.
- `void generarNombreDeUsuario(SOCKET socketCliente)`: El usuario ingresa el tamaño del nombre de usuario y le manda un mensaje al socket servidor indicándole el mismo. Si la longitud es válida, el socket cliente recibe un mensaje que dice “Longitud valida” y después otro mensaje con el nombre de usuario generado; si la longitud no es válida, recibe un mensaje que dice “Longitud invalida” y el cliente tiene que volver a ingresar otra longitud para repetir el proceso hasta que se ingrese una válida.

- void generarContraseña(SOCKET socketCliente): El usuario ingresa el tamaño de la contraseña y le manda un mensaje al socket servidor indicándole el mismo. Si la longitud es válida, el socket cliente recibe un mensaje que dice “Longitud valida” y después otro mensaje con la contraseña generada; si la longitud no es válida, recibe un mensaje que dice “Longitud invalida” y el cliente tiene que volver a ingresar otra longitud para repetir el proceso hasta que se ingrese una válida.

Por otro lado el cliente Python solo está compuesto por un único archivo (Cliente.py), ya que el desarrollo de sockets con el lenguaje Python es mucho más simple que con el lenguaje C. Incluso con menos código, tiene las mismas funcionalidades que el cliente C. El cliente Python también puede crear un socket, conectarse al servidor y mandarle mensajes solicitando la generación de un nombre de usuario o de una contraseña.

También fue desarrollado en Windows y utiliza líneas de código exclusivas de Windows, como por ejemplo: `os.system('cls')` que limpia los mensajes de la consola.

3. Pruebas realizadas

3.1. Pruebas entre el servidor y el cliente C

1) Iniciar servidor

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Socket cliente-servidor\Servidor\bin\Debug\Servidor.exe"  
Escuchando por conexiones con nuevos clientes
```

2) Iniciar cliente C y que este se conecte al servidor

Cliente

```
Conectado con el servidor  
Indique que desea generar  
1) Nombre de usuario  
2) Contraseña  
Ingrese 0 para salir  
_
```

Servidor

```
Escuchando por conexiones con nuevos clientes  
Cliente aceptado  
*****  
Esperando a que el cliente ingrese una opción valida
```

3) Ingresar opciones no válidas en el menú

```
Conectado con el servidor  
Indique que desea generar  
1) Nombre de usuario  
2) Contraseña  
Ingrese 0 para salir  
asfv  
Ingrese solo un número  
673  
Elija una opción valida  
Indique que desea generar  
1) Nombre de usuario  
2) Contraseña  
Ingrese 0 para salir
```

4) Ingresar la opción 1

Cliente

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Socket cliente-servidor\Cliente C\bin\Debug\Cliente.exe"
```

```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
```

Servidor

```
Escuchando por conexiones con nuevos clientes
Cliente aceptado
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 1
Se eligio la opción: Generar nombre de usuario
```

5) Ingresar texto en vez de un número

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Socket cliente-servidor\Cliente C\bin\Debug\Cliente.exe"
```

```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
```

```
fdhgr
```

```
Ingrese solo un número
```

6) Ingresar un número que NO este entre 5 y 15

Cliente

```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
```

```
fdhgr
```

```
Ingrese solo un número
```

```
2000
```

```
Longitud invalida
```

```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
```

Servidor

```
Escuchando por conexiones con nuevos clientes
Cliente aceptado
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 1
Se eligio la opción: Generar nombre de usuario
Longitud recibida: 2000
Longitud invalida. Se espera una longitud entre 5 y 15. Informando al cliente
```

7) Ingresar un número que SI este entre 5 y 15

Cliente


```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
fdhgr
Ingrese solo un número
2000
Longitud invalida
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
5
El nombre de usuario generado es: ASUMO
Presione una tecla para continuar . . .
```

Servidor

```
Escuchando por conexiones con nuevos clientes
Cliente aceptado
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 1
Se eligio la opción: Generar nombre de usuario
Longitud recibida: 2000
Longitud invalida. Se espera una longitud entre 5 y 15. Informando al cliente
Longitud recibida: 5
Longitud valida
Nombre de usuario: ASUMO
*****
Esperando a que el cliente ingrese una opción valida
```

8) Ingresar la opción 2

Cliente

 "C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Sockets cliente-servidor\Cliente C\bin\Debug\Cliente.exe"

```
Indique cuantos caracteres quiere en código (debe ser entre 8 y 50)
```

Servidor (recorto la imagen del servidor para que no ocupe demasiado espacio)

```
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 2
Se eligio la opción: Generar contraseña
```

9) Ingresar texto en vez de un número

```
Indique cuantos caracteres quiere en código (debe ser entre 8 y 50)
behd
Ingrese solo un número
```


10) Ingresar un número que NO este entre 8 y 50

Cliente

```
Indique cuantos caracteres quiere en código (debe ser entre 8 y 50)
behd
Ingrese solo un número
3
Longitud invalida
Indique cuantos caracteres quiere en código (debe ser entre 8 y 50)
```

Servidor

```
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 2
Se eligio la opción: Generar contraseña
Longitud recibida: 3
Longitud invalida. Se espera una longitud de entre 8 y 50. Informando al cliente
```

11) Ingresar un número que SI este entre 8 y 50

Cliente

```
Indique cuantos caracteres quiere en código (debe ser entre 8 y 50)
behd
Ingrese solo un número
3
Longitud invalida
Indique cuantos caracteres quiere en código (debe ser entre 8 y 50)
12
El código generado es: k9VRC9sVasHY
Presione una tecla para continuar . . .
```

Servidor

```
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 2
Se eligio la opción: Generar contraseña
Longitud recibida: 3
Longitud invalida. Se espera una longitud de entre 8 y 50. Informando al cliente
Longitud recibida: 12
Longitud valida
Contraseña: k9VRC9sVasHY
*****
Esperando a que el cliente ingrese una opción valida
```

12) Ingresar la opción 0

Cliente (después de presionar una tecla se cierra el programa)

```
Ha finalizado su sesión
Presione una tecla para continuar . . .
```

Servidor

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Socket cliente-servidor\Servidor\bin\Debug\Servidor.exe"
Escuchando por conexiones con nuevos clientes
```

3.2. Pruebas entre el servidor y el cliente Python

1) Iniciar cliente Python y que este se conecte al servidor (el servidor queda abierto esperando por nuevas conexiones después de que se desconecta el cliente C)

Cliente

```
C:\Windows\System32\cmd.exe - python Cliente.py
Microsoft Windows [Versión 10.0.19045.46]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Socket cliente-servidor>
Conectado al servidor
Indique que desea generar
1) Nombre de usuario
2) Contraseña
Ingrese 0 para salir
```

Servidor

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Socket cliente-servidor\Servidor\bin\Debug\Servidor.exe"
Escuchando por conexiones con nuevos clientes
Cliente aceptado
*****
Esperando a que el cliente ingrese una opción válida
```

2) Ingresar opciones no válidas en el menú

```
Conectado al servidor
Indique que desea generar
1) Nombre de usuario
2) Contraseña
Ingrese 0 para salir
qwrX
Ingrese solo un número
456
Elija una opción válida
Indique que desea generar
1) Nombre de usuario
2) Contraseña
Ingrese 0 para salir
```

3) Ingresar la opción 1

Cliente

```
C:\Windows\System32\cmd.exe - python Cliente.py
```

```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
```

Servidor

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Sockets cli
Escuchando por conexiones con nuevos clientes
Cliente aceptado
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 1
Se eligio la opción: Generar nombre de usuario
```

4) Ingresar texto en vez de un número

```
C:\Windows\System32\cmd.exe - python Cliente.py
```

```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
khfve
Ingrese solo un número
_
```

5) Ingresar un número que NO este entre 5 y 15

Cliente

```
C:\Windows\System32\cmd.exe - python Cliente.py
```

```
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
khfve
Ingrese solo un número
1
Longitud invalida
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
```

Servidor

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Sockets cliente-servidor\Servidor\bin\Debug\Servidor.exe"
```

```
Escuchando por conexiones con nuevos clientes
Cliente aceptado
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 1
Se eligio la opción: Generar nombre de usuario
Longitud recibida: 1
Longitud invalida. Se espera una longitud entre 5 y 15. Informando al cliente
```

6) Ingresar un número que SI este entre 5 y 15

Cliente

```
C:\Windows\System32\cmd.exe - python Cliente.py
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
khfve
Ingrese solo un número
1
Longitud invalida
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 5 y 15)
9
El nombre de usuario generado es: EMOJAmUhI
Presiona Enter para continuar...
```

Servidor

```
"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Sockets cliente-servidor\Servidor\bin\Debug\Servidor.exe"
Escuchando por conexiones con nuevos clientes
Cliente aceptado
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 1
Se eligio la opción: Generar nombre de usuario
Longitud recibida: 1
Longitud invalida. Se espera una longitud entre 5 y 15. Informando al cliente
Longitud recibida: 9
Longitud valida
Nombre de usuario: EMOJAmUhI
*****
Esperando a que el cliente ingrese una opción valida
```

7) Ingresar la opción 2

Cliente

```
C:\Windows\System32\cmd.exe - python Cliente.py
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 8 y 50)
```

Servidor

```
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 2
Se eligio la opción: Generar contraseña
```

8) Ingresar texto en vez de un número

```
C:\Windows\System32\cmd.exe - python Cliente.py
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 8 y 50)
yuop
Ingrese solo un número
```

9) Ingresar un número que NO este entre 8 y 50

Cliente

```
C:\Windows\System32\cmd.exe - python Cliente.py
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 8 y 50)
yuop
Ingrese solo un número
7
Longitud invalida
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 8 y 50)
```

Servidor

```
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 2
Se eligio la opción: Generar contraseña
Longitud recibida: 7
Longitud invalida. Se espera una longitud de entre 8 y 50. Informando al cliente
```

10) Ingresar un número que SI este entre 8 y 50

Cliente

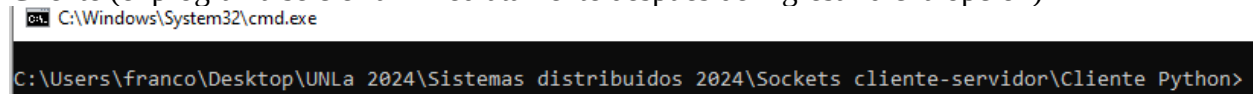
```
C:\Windows\System32\cmd.exe - python Cliente.py
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 8 y 50)
yuop
Ingrese solo un número
7
Longitud invalida
Indique cuantos caracteres quiere en su nombre de usuario (debe ser entre 8 y 50)
40
El código generado es: otFnS62INUCdJNw1M7AMBrAQWLFgDryvLGAWLysS
Presiona Enter para continuar...
```

Servidor

```
*****
Esperando a que el cliente ingrese una opción valida
Opción ingresada: 2
Se eligio la opción: Generar contraseña
Longitud recibida: 7
Longitud invalida. Se espera una longitud de entre 8 y 50. Informando al cliente
Longitud recibida: 40
Longitud valida
Contraseña: otFnS62INUCdJNw1M7AMBrAQWLFgDryvLGAWLysS
*****
Esperando a que el cliente ingrese una opción valida
```

11) Ingresar la opción 0

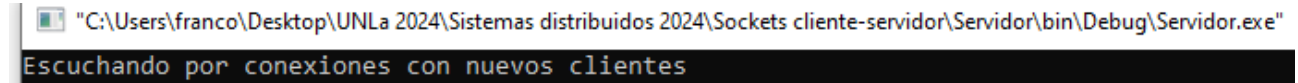
Cliente (el programa se cierra inmediatamente después de ingresar dicha opción)



cmd C:\Windows\System32\cmd.exe

```
C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Sockets cliente-servidor\Cliente Python>
```

Servidor



"C:\Users\franco\Desktop\UNLa 2024\Sistemas distribuidos 2024\Sockets cliente-servidor\Servidor\bin\Debug\Servidor.exe"

```
Escuchando por conexiones con nuevos clientes
```