Programación de

Servicios y Procesos

CLIENTE-SERVIDOR

Diego González Martínez | 12/01/2023

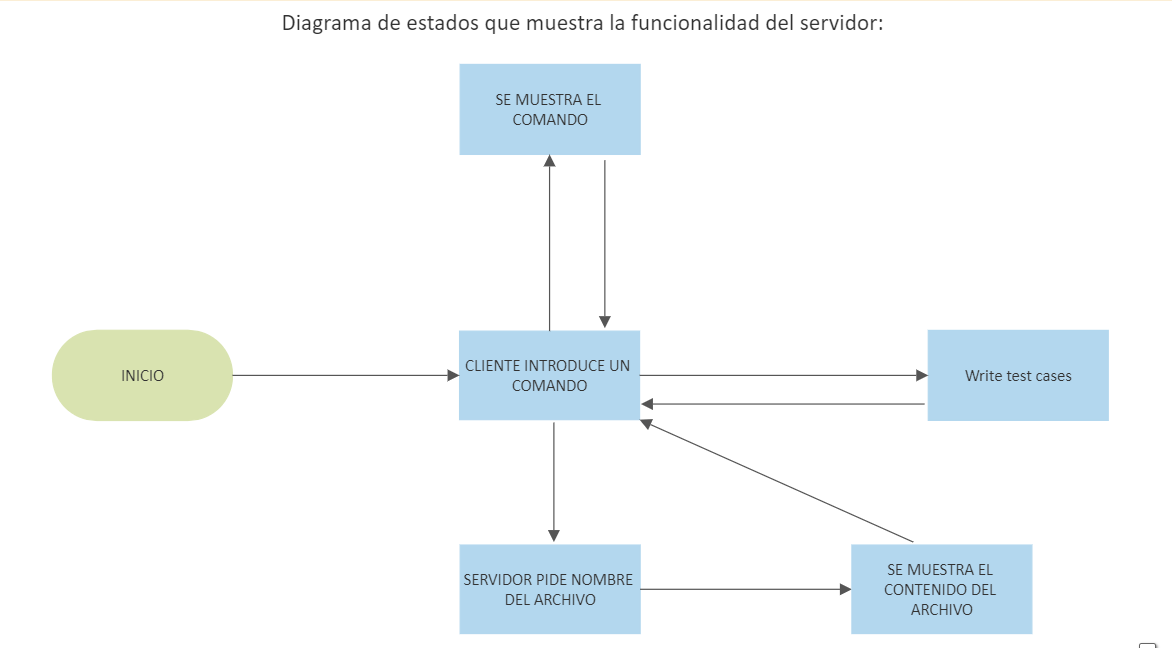
# Ejercicio 4.3

**¿De qué trata el Ejercicio 4?**

**Actividad 4.3.** A partir del ejercicio anterior crea un servidor que una vez iniciada sesión a través de un nombre de usuario y contraseña específico (por ejemplo javier / secreta) el sistema permita Ver el contenido del directorio actual, mostrar el contenido de un determinado archivo y salir.

Este código es un ejemplo de un servidor multithreading en Java. El servidor crea un nuevo hilo para manejar cada conexión entrante de un cliente. El servidor espera que el cliente envíe un nombre de usuario y contraseña válidos para iniciar sesión. Si el inicio de sesión es exitoso, el servidor entra en un bucle while que permite al cliente enviar comandos. Los comandos disponibles son "ls", "get" y "exit". El comando "ls" permite al cliente ver los archivos en un directorio específico, "get" permite al cliente descargar un archivo específico del servidor y "exit" permite al cliente salir del bucle while y cerrar la conexión.

Diagrama de estado:



**¿En qué se diferencian el ejercicio 4.2 y 4.3?**

* El código incluye lógica de inicio de sesión donde el cliente debe proporcionar un nombre de usuario y contraseña.
* Si el nombre de usuario y la contraseña son válidos, el código entra en un bucle while que permite al cliente enviar comandos.
* El código incluye una nueva opción "ls" que permite al cliente ver los archivos en un directorio específico.
* El código también tiene una opción "exit" que permite al cliente salir del bucle while y cerrar la conexión.
* El código cambia la ruta de los archivos que se van a leer, en este caso se encuentran en una carpeta llamada "poemas"
* El codigo no tiene una variable numFiles, esta no tiene uso alguno en el codigo.

En resumen, el código es similar al anterior, pero incluye lógica adicional para manejar la autenticación de usuario, proporcionar acceso a un directorio de archivos y permitir al cliente enviar comandos específicos.

**Clase Servidor-hilo:**

****

Dentro del método "run()", se crean objetos "DataInputStream" y "DataOutputStream" para recibir y enviar datos a través del socket, respectivamente.

El servidor envía un mensaje al cliente solicitando el nombre de usuario y recibe la respuesta.

El servidor envía un mensaje al cliente solicitando la contraseña y recibe la respuesta.

El servidor comprueba si el nombre de usuario y la contraseña son válidos. Si son válidos, el servidor envía una respuesta "true" al cliente y entra en un bucle while que permite al cliente enviar comandos.

Dentro del bucle while, el servidor espera a recibir un comando del cliente. Si el comando es "ls", el servidor envía una lista de archivos en un directorio específico al cliente. Si el comando es "get", el servidor recibe la ruta de un archivo del cliente y envía el contenido del archivo al cliente si existe. Si el comando es "exit", el servidor sale del bucle while y cierra la conexión.

Si el nombre de usuario y la contraseña no son válidos, el servidor envía una respuesta "false" al cliente y no entra en el bucle while. 11. Finalmente, el código cierra los flujos de datos y el socket para liberar los recursos y finalizar la conexión.

El código también maneja las excepciones IOException con un Logger, para poder detectar y registrar errores en caso de suceder alguno.

**Clase Cliente:**

****

****

Antes de solicitar la ruta del archivo al usuario, el cliente recibe un mensaje del servidor pidiendo un nombre de usuario, que el usuario ingresa y envía de regreso al servidor. Luego, el cliente recibe otro mensaje pidiendo una contraseña, que el usuario también ingresa y envía al servidor. El servidor luego envía un valor booleano indicando si el inicio de sesión fue exitoso o no. Si fue exitoso, el cliente entra en un ciclo donde recibe comandos del servidor, como "ls" para listar los archivos en el directorio actual, "get" para solicitar un archivo específico y "exit" para cerrar la conexión. En el caso de "get envia la ruta del archivo, recibiendo si el archivo existe o no y su contenido si existe.

**JAVADOC:**

**C:\4.3Proyecto\target\site\apidocs\**

