Chat con Sockets TCP Y UDP

ADRIAN DE CASTRO ROMÁN, 2ºDAM A

INDICE

Adrian De Castro Román
2ºDam A

a) CLASES SERVIDOR:

1. Funcionalidades:

- **ServidorTCP:** Es un servidor que utiliza el protocolo de comunicación TCP para establecer conexiones con clientes. Los clientes se conectan al servidor a través de una dirección IP y un puerto específico, y pueden enviar y recibir mensajes de texto. El servidor mantiene una conexión persistente con cada cliente y se encarga de distribuir los mensajes de un cliente a todos los demás clientes conectados.
- **ServidorUDP:** Es un servidor que utiliza el protocolo de comunicación UDP para establecer conexiones con clientes. Los clientes se conectan al servidor a través de una dirección IP y un puerto específico, y pueden enviar y recibir mensajes de texto. A diferencia del protocolo TCP, UDP no establece conexiones persistentes entre el servidor y los clientes, lo que significa que cada mensaje enviado por un cliente debe incluir información sobre el destino del mensaje. El servidor se encarga de distribuir los mensajes de un cliente a todos los demás clientes conectados.

2. Estructura del código:

- **ServidorTCP:** El servidor TCP tendrá una lista que se autoordena sola con la clase "ordenarLista", la cual ordenará los usuarios por orden alfabético, además de dejar primero los administradores. Dispone de un bucle infinito que acepta conexiones entrantes en todo momento, creando hilos para mantener la comunicación con los clientes los cuales procesan los mensajes recibidos por ellos y los procesan como corresponda.
- **ServidorUDP**: El servidor UDP se divide en una única clase: ServidorUDP. Esta clase se encarga de recibir mensajes entrantes de los clientes y de distribuir los mensajes recibidos de un cliente a todos los demás clientes conectados. Cada cliente se identifica por su dirección IP y su puerto, almacenados en una lista, la cual en base a la cantidad de usuarios, limitará la entrada a otros o no, enviándoles el comando "CHATFUL" de ser necesario.

3. Algoritmos principales:

ServidorTCP:

- **Aceptando conexiones:** El servidor TCP utiliza un bucle infinito para esperar y aceptar conexiones entrantes de los clientes. Cuando se recibe una conexión entrante, se crea un nuevo objeto Cliente que se encarga de gestionar la comunicación con ese cliente en particular. El objeto Cliente se almacena en una lista de clientes conectados.
- **Distribución de mensajes:** Cuando un cliente envía un mensaje al servidor, el servidor itera sobre la lista de clientes conectados y envía el mensaje a todos los clientes, excepto al cliente que envió el mensaje.

ServidorUDP:

- **Recepción de mensajes:** El servidor UDP utiliza un bucle infinito para esperar y recibir mensajes entrantes de los clientes. Cuando se recibe un mensaje, se extrae la dirección IP y el puerto del cliente que envió el mensaje. Si el cliente aún no está registrado en la lista de clientes conectados, se agrega a la lista.
- **Distribución de mensajes:** Cuando un cliente envía un mensaje al servidor, el servidor itera sobre la lista de clientes conectados y envía el mensaje a todos los clientes, excepto al cliente que envió el mensaje. Cada mensaje enviado por el servidor debe incluir la dirección IP y el puerto del cliente que envió el mensaje, para que los otros clientes sepan a quién responder si lo desean.

Adrian De Castro Román
2ºDam A

b) CLASES CLIENTE

1. Funcionalidades:

- **ClienteTCP:** Permite la comunicación entre múltiples clientes y un servidor mediante el protocolo TCP. Se encarga de enviar y recibir mensajes entre los clientes y el servidor, y también de mantener actualizada la lista de usuarios conectados. El cliente TCP también tiene una interfaz gráfica que permite al usuario enviar mensajes y ver los mensajes recibidos de otros usuarios.
- **ClienteUDP:** Al igual que el ClienteTCP, permite la comunicación entre múltiples clientes y un servidor. Sin embargo, utiliza el protocolo UDP en lugar de TCP. El cliente UDP también tiene una interfaz gráfica que permite al usuario enviar mensajes y ver los mensajes recibidos de otros usuarios.

2. Estructura del código:

- **ClienteTCP:** El código del ClienteTCP está estructurado en diferentes métodos. El método principal es el constructor de la clase, que se encarga de establecer la conexión con el servidor y enviar y recibir mensajes. También hay métodos para actualizar la lista de usuarios conectados y para mostrar una ventana de espera en caso de que el chat esté lleno o no se pueda establecer la conexión. La clase también tiene una clase interna llamada ChatReader, que se encarga de leer los mensajes recibidos del servidor y actualizar la interfaz gráfica.
- **ClienteUDP:** El código del ClienteUDP también está estructurado en diferentes métodos. El método principal es el constructor de la clase, que se encarga de establecer la conexión con el servidor y enviar y recibir mensajes. También hay métodos para enviar mensajes, actualizar la lista de usuarios conectados y mostrar una ventana de espera en caso de que el chat esté lleno o no se pueda establecer la conexión. La clase tiene un hilo de ejecución llamado hiloReceptor, que se encarga de leer los mensajes recibidos del servidor y actualizar la interfaz gráfica.

3. Algoritmos principales:

- **ClienteTCP:** El algoritmo principal del ClienteTCP es el constructor de la clase. Este método se encarga de establecer la conexión con el servidor y enviar y recibir mensajes. El algoritmo también incluye la clase interna ChatReader, que se encarga de leer los mensajes recibidos del servidor y actualizar la interfaz gráfica. El método de actualización de la lista de usuarios conectados también es importante, ya que se encarga de solicitar la lista al servidor y actualizar la interfaz gráfica con los usuarios conectados.
- **ClienteUDP:** El algoritmo principal del ClienteUDP también es el constructor de la clase, que se encarga de establecer la conexión con el servidor y enviar y recibir mensajes. El método para enviar mensajes también es importante, ya que se utiliza para enviar los mensajes escritos por el usuario al servidor. El hilo de ejecución hiloReceptor es crucial para la lectura de los mensajes recibidos del servidor y la actualización de la interfaz gráfica. También es importante el método de actualización de la lista de usuarios conectados, que se encarga de solicitar la lista al servidor y actualizar la interfaz gráfica con los usuarios conectados.

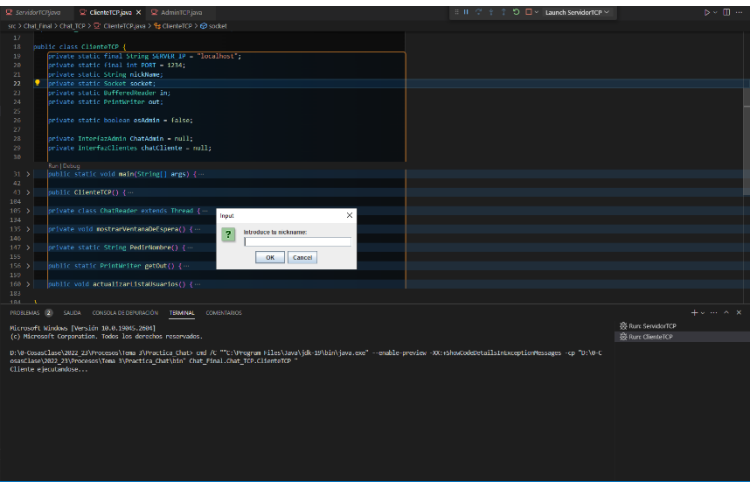
c) CLASES ADMINISTRADOR

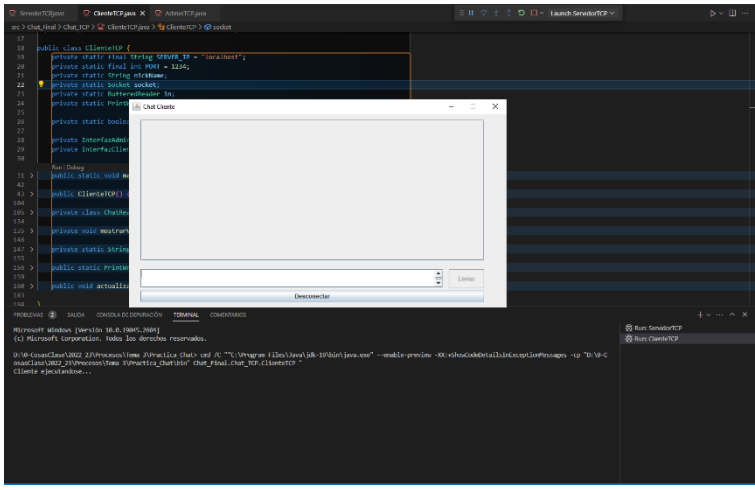
Simplemente inicializan clases Cliente con el argumento “True”, el cual se asigna a la variable “esAdmin”. Dicha variable, servirá para activar las características de administrador, no sólo para que al preguntar el nombre al usuario se le añada el identificador “(ADMIN)”, sino que también tendrá su propia interfaz gráfica, la cual se diferencia con el cliente por 2 elementos, la lista de participantes en el chat y el botón de expulsar.

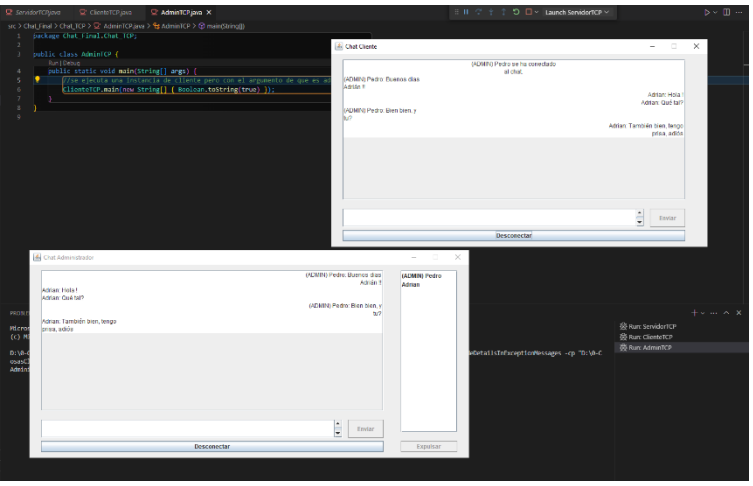
Al iniciarse el Cliente como un Administrador, le permitirá hacer uso de comandos internos de desconexión (que se activará al seleccionar un usuario de la lista y pulsar el botón de expulsar) y de solicitud de la lista de usuarios (que se activará al conectarse el administrador en cuestión al chat y cada vez que un usuario se conecte o desconecte del chat). Al enviar las solicitudes (“/dc nickname” ó “/getUsers”), el servidor comprobará si quien las solicitó fue un administrador, y de ser así procederá acorde a ello:

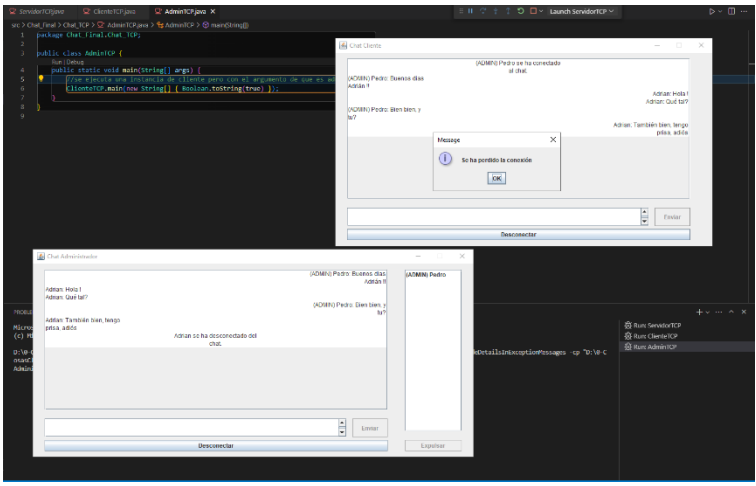
- /dc nickname : el servidor al recibir el comando, buscará quién envió el mensaje y quién es el usuario a desconectar, al identificarlos, procederá a cortar la comunicación con el socket (en el caso de la versión TCP) ó enviar el comando “/dc” al cliente objetivo para que se cierre (en el caso de la versión UDP), en ambos casos, el usuario desconectado será eliminado de la lista de usuarios.
- /getUsers : al recibir este comando el servidor, enviará la lista de los usuarios al administrador que lo solicitó, el cual con esa información, actualizará su lista grafica de los participantes.

d) PRUEBAS VERSION TCP:



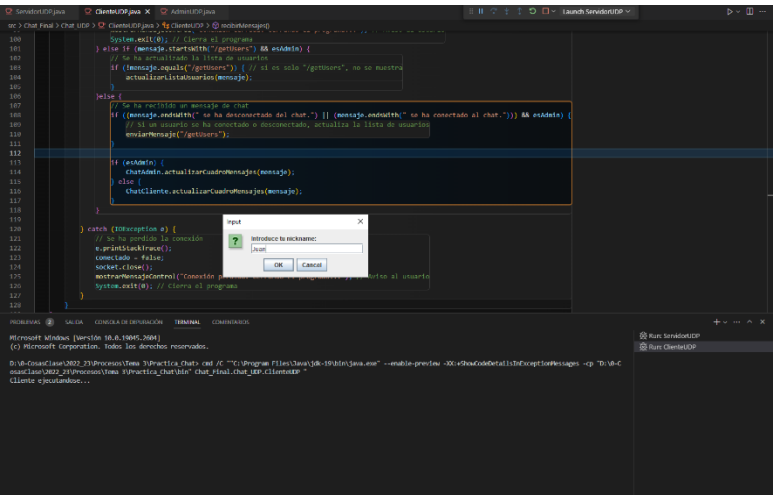


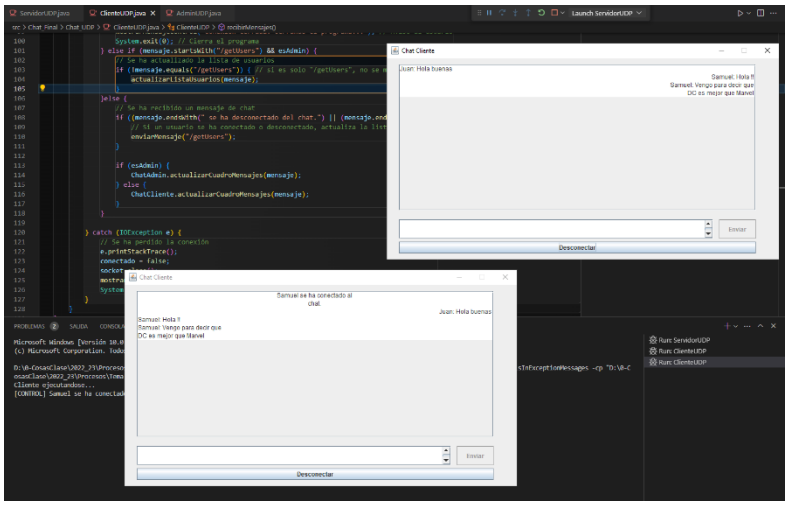


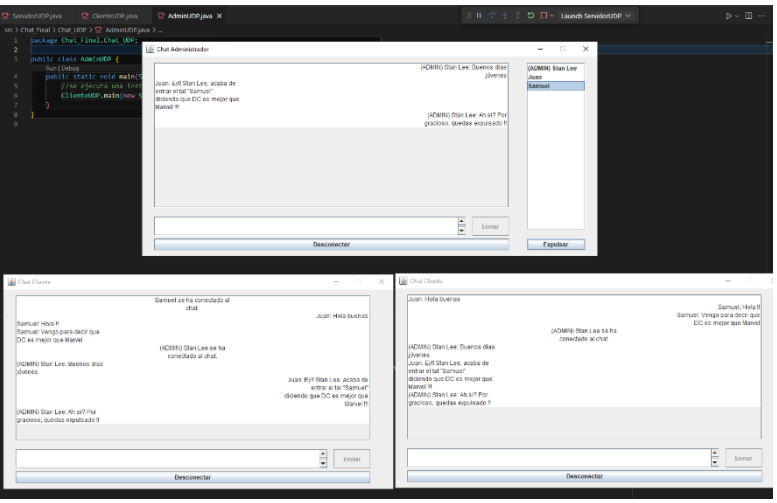


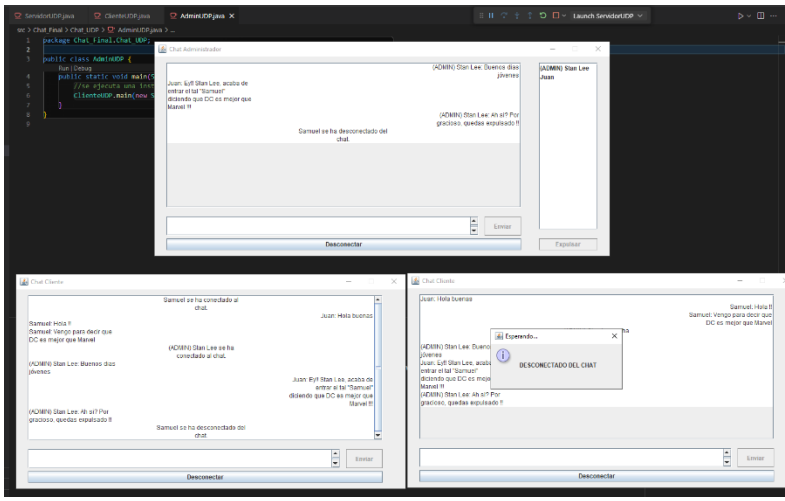
Adrian De Castro Román
2ºDam A

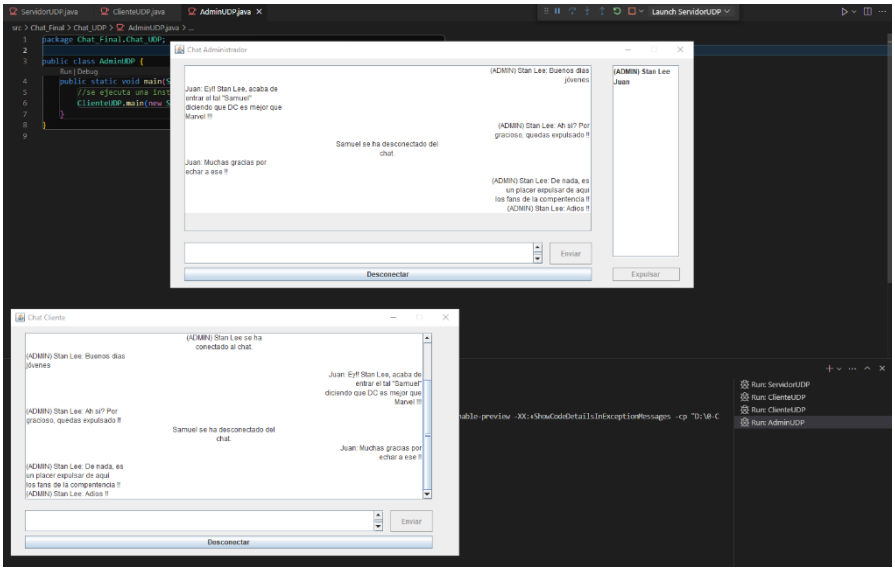
PRUEBAS VERSION UDP:











Adrian De Castro Román
2ºDam A