[AA1] Ejercicio 1
Chat TCP – Cliente/Servidor

Objetivo: Recordar la programación cliente/servidor con TCP

Ítem 1: Cómo organizaremos el código

Los organizaremos en dos proyectos dentro de la misma solución: Uno para cliente y otro para servidor.

Recordemos que el servidor no tiene interfaz gráfica y sus acciones deberían ser automáticas, como consecuencia de los mensajes que llegan de los clientes. En el servidor no hay nadie enviando mensajes.

Ítem 2: Cómo será la interfaz gráfica

Prepararemos una interfaz sencilla en la que se puedan escribir y encolar mensajes en una lista.

La ventana es 800x600. Se reserva la parte inferior para escribir y la parte superior para mostrar los mensajes que se van escribiendo.

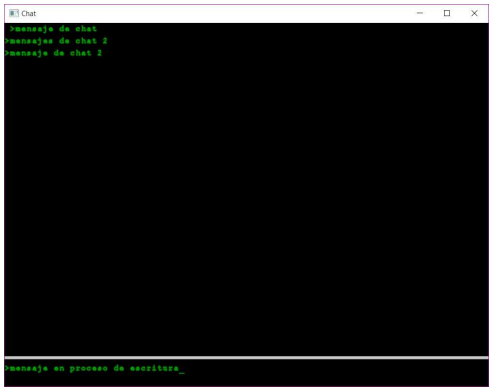
Cada vez que se clique enter, el mensaje de la parte inferior desaparece y pasa a la lista superior.

La lista de mensajes mostrará un máximo de 25 mensajes.

En esta ventana tendrán que visualizarse los mensajes del peer local cuando los envíe y del peer remoto, cuando se reciban.

Tenéis una propuesta de esta parte gráfica en el Classlife.

Esta interfaz sólo se mostrará desde los clientes.



Ítem 3: Cómo se pone en marxa el chat

En primer lugar, se pone en marcha el servidor.

Antes de abrir el listener, nos debe preguntar en qué modo queremos probar el chat. Habrá tres posibilidades:

1. Blocking + Threading
2. NonBlocking
3. Blocking + SocketSelector

Al elegir la opción, se pondrá en funcionamiento el servidor según se haya indicado.

Una vez iniciado el servidor, cuando ya lo tenemos escuchando por el puerto del listener, ponemos en marcha el cliente.

Al poner en marcha el cliente y antes de conectarse con el servidor, se le preguntará en qué modo quiere probar el chat. En este caso, se darán dos opciones:

1. Blocking + Threading
2. NonBlocking

Ítem 4: Funcionalidades que debe cumplir el cliente

- Debe poder conectarse al servidor
- Debe recibir un aviso cuando se conecte un cliente nuevo
- Debe recibir un aviso cuando se desconecte un cliente
- Debe poder enviar sus mensajes
- Debe recibir los mensajes de los demás a través del servidor

Ítem 5: Funcionalidades que debe cumplir el servidor

- Debe aceptar conexiones de nuevos clientes en cualquier momento
- Debe avisar a todos los clientes cuando se conecte un cliente nuevo
- Debe avisar a todos los clientes de cuando se desconecta un cliente

Debe recibir los mensajes de los clientes y reenviarlos a todos (o a todos excepto al que lo envía, depende de la implementación de cada uno).

Ítem 6: Finalización del chat

- Un cliente se desconecta cuando se clica Esc o se cierra la ventana. El servidor debe ser capaz de darse cuenta y avisará a los demás clientes conectados de que alguien se desconectó.
- El servidor finaliza cuando se desconectan todos los clientes que habían conectados. En este caso, será necesario un caso especial al principio, cuando se conecta el servidor desde cero.

Rúbrica

1. N clientes pueden establecer conexión (20%)
2. Se envían los mensajes entre los clientes conectados
    a. Utilizando la versión socket blocking (15%)
    b. Utilizando la versión socket nonblocking (15%)
    c. Utilizando la versión socketselector + blocking (15%)
3. Se controlan los status (15%)
4. Desconexión correcta. Se liberan recursos correctamente. (20%)

Entregable del ejercicio → Código

1. Nos aseguramos de que la solución compila en debug y en release.
2. Nos aseguramos de que el proyecto no utiliza paths absolutos para linkar las librerías.
3. Eliminamos el fichero .vs (oculto, si existe)
4. Hacemos un zip de la solución
5. Lo subimos al Classlife en este formato: Ejercicio1_Nombre1Apellido1Nombre2Apellido2.zip