Trabajo Práctico de Redes y Comunicaciones

Alumno: Roger Ricciardulli

DNI: 42043541

Materia: Redes y Comunicaciones

2do Cuatrimestre de 2021

Estrategia de resolución:

La aplicación se divide en dos proyectos, Cliente y Server.

**Server:**

El Server contiene todas las estructuras en las que se van guardando los distintos datos, las tres estructuras son Usuario, TipoDeMedicamento, Medicamento, de esas estructuras declaro variables globales (junto a otras que tienen diferentes funciones, pero básicamente lo que hacen es posibilitar la comunicación rápida de datos sin tener que ir pasándolos a otras funciones) para utilizarlas en todo el programa, en dichas variables cargo los datos según sea necesario.

Todas las variables posibilitan el funcionamiento:

\*contadorMedicamento permite llevar un conteo de los medicamentos existentes, este se fija en un documento txt llamado contador1 para saber cuántos medicamentos hay, y lo aumenta en caso de que sea necesario.

\*contadorTipoDeMedicamento de la misma forma ocurre con el anterior este sirve para contar los tipos de medicamento, funcionando de la misma manera solo que llama a un documento txt llamado contador2.

\*usuarioActual este permite identificar en todo momento el usuario que esta interactuando con el programa, por lo que cuando un usuario se loguea este valor se actualiza.

\*datos guarda la información devuelta por las funciones para enviársela al cliente.

\*elección permite identificar cual es la interacción que hizo el usuario y devolverle una determinada respuesta.

\*iniciarConexion hace que una vez que se detecte un cliente a este se le envíe la solicitud de logueo y lo sigue mandando hasta que el usuario se loguee o se supere la cantidad de intentos y se reinicie el proceso.

\*intentos es un contador que permite saber cuantos intentos de logueo lleva el usuario. Se reinicia una vez pasado los 3 intentos.

\*puertoEscucha es el puerto escucha en el cual espera la conexión.

\*tAux guarda en una variable auxiliar los valores del tipo de medicamento en caso de que un valor no se modifique o de paso no se realice ninguna modificación.

\*mAux guarda en una variable auxiliar los valores del medicamento en caso de que un valor no se modifique o de paso no se realice ninguna modificación.

\*tipoAModificar permite saber cuál es el número, dentro del vector de tipo de medicamento, a modificar.

\*medAModificar permite saber cuál es el número, dentro del vector de medicamento, a modificar.

\*u es el vector de usuarios en los que se van a guardar los datos recolectados de un txt llamado archivo (contiene los usuarios designados). Tiene de valor designado 4, es decir pueden ser 5 usuarios.

\*t es el vector de tipo de medicamento en los que se van a guardar los datos recolectados del archivo binario TipoDeMedicamento. Tiene de valor designado 20, es decir pueden ser 21 tipos de medicamento.

\*m es el vector de medicamento en los que se van a guardar los datos recolectados del archivo binario Medicamento. Tiene de valor designado 20, es decir pueden ser 21 medicamentos.

El Server empieza con un bucle while en el cual espera la conexión de algún cliente, ni bien un Cliente se conecta empieza la comunicación, es decir, empieza otro bucle while (yes) en el que con un formato de enviar-recibir se comunica con el cliente hasta que este se desconecte o cierre sesión, si esto sucede el Server rompe el bucle yes y se dispone a escuchar la conexión de otro cliente. Si no detecta ningún cliente o en algún momento se corta la conexión con dicho cliente se queda en espera de otras conexiones.

Solo permite una comunicación por vez por cliente por lo que si otro cliente se conecta este se queda esperando hasta que el cliente, con el cual el servidor se está comunicando, deje de hacerlo.

Server->Enviar: Una vez identifico una conexión con un cliente, como ya mencioné, se le envía la petición de logueo, cuando se supera la cantidad máxima se reinicia el contador, se le envía otro mensaje y se reinicia el proceso de logueo. Después de esto tengo un switch (elección, variable que se carga en el recibir) en el que dependiendo lo que reciba se le enviará un mensaje u otro o se hará determinado paso, como cerrar sesión, mandar archivo, etc.

Server->Recibir: Me fijo si el mensaje recibido está vacío, si lo está es que no hay un cliente conectado por lo que pongo el sistema en pausa y rompo el bucle de enviar/recibir. Si se conecta bien, me fijo si esta en proceso de logueo, si lo esta verifico llamando a una función si el usuario y contraseña están bien, digo que el iniciar conexión pasa a ser falso y escribo el ingreso del usuario tanto en el log general del server, como al del usuario particular, llamando a una función y a otra, y fijo la elección del switch en 15 para cuando vuelva al enviar le envíe un mensaje al cliente de “se ha aceptado la conexión”. Por el contrario, si se deniega la conexión se aumenta el contador de intentos.

Paso al switch (elección) que lo que hace es responder de acuerdo a lo que re recibe, llama a diversas funciones y mientras los datos estén correctos sigue el “hilo” hasta terminar dicho funcionamiento.

Por último, me fijo si lo que recibo es un número para empezar el posterior funcionamiento de las funciones.

**Cliente:**

El cliente tiene una clase cliente que contiene Enviar, Recibir y Cerrar Socket.

Tiene una ejecución en la, en un bucle while, primero recibe y luego envía.

Variables globales:

\*datos funciona de la misma manera que la variable del server, permite el traslado de información de manera sencilla.

\*elección igual que el anterior, permite identificar cual es la interacción que hizo el usuario y devolverle una determinada respuesta.

\*puertoEscucha es el puerto escucha en el cual espera la conexión.

Cliente->Enviar: Declaro una opción que permite identificar que es lo que ha escogido el usuario en el menú, también, declaro un clock para por llevar el conteo del tiempo que ha pasado el cliente sin enviar un mensaje al servidor.

Switch (elección) funciona de la misma manera que en el server, dependiendo de que se reciba hará determinada acción.

Como había declarado la opción en 0 esto me permite que acceder al while que mostrara el menú. El menú es bastante sencillo, por ejemplo:

"\n Opcion 1: - Tipo de medicamento"

"\n---------Tipo de medicamento---------:"

“\nOpcion 1: - Crear”

"\nOpcion 2: - Administrar”

"\nOpcion 3: - Volver\n"

Si escojo 1, esto le enviará al servidor el valor 1 y este si todos los datos están bien y no hubo ningún inconveniente, me devolverá el id el tipo de medicamento creado, por el contrario, me dirá que tipo de error sucedió.

Se sale del menú cuando la el buffer, es decir lo que le envió al servidor, es distinto de 3, ya que 3 es la opción que nos permite volver a atrás. Como importante aclaración:

"\n Opcion 5: - Ver tiempo de inactividad\n"

Permite, como ya dije, saber cuánto tiempo ha pasado sin enviar mensaje y si se excede los 2 minutos se desconecta al usuario. Lo hice de esta forma ya que al estar en el menú el cliente esta en modo enviar, por lo que el servidor esta en modo escucha, por ello este no puede enviarle ningún mensaje.

Cliente->Recibir: Guarda en la variable mensaje lo que recibe del buffer, si esta variable esta vacía es que no encontró ningún server, por lo que procederá a mostrarle al cliente el mensaje de: "No se encuentra un servidor" y lo cerrará.

Abre un switch (eleccion) que muestra los datos que recibe y setea la eleccion para seguir con el proceso en el que este parado. Después, lo que hace es hacer muchos ifs fijándose si los mensajes que recibe son alguno de los prefijados, si es así, carga el buffer y setea la eleccion, para después, en el enviar, mostrar el mensaje y guardar los datos que le pide el server, dado que la eleccion cambio y es distinta de 0 se salta el menú y directamente envía los dataos que guardo.

El Cliente esta organizado de tal forma que, al contrario del servidor, primero recibe y después envía.

Ejemplo de comunicación:

Recibo la conexión, le envío la petición de logueo, recibo ciertos datos, como el iniciarConexion está en true, verifico si los datos que me proporciono el usuario son correctos (para agilizar la explicación vamos a suponer que los datos son correctos) como la verificación dio positiva el iniciarConexion lo paso a false y fijo la elección como 15, por lo que no va a entrar en el switch, paso al enviar de nuevo, como la elección es 15 se le envía el mensaje "\nSe ha aceptado la conexion".

Una vez hecho esto el cliente entrara en el bucle del menú que le permitirá elegir que desea hacer (supongamos que quiere modificar un tipo de medicamento), se le enviara al servidor el valor 2, el servidor se da cuenta de que es un numero por lo que lo guarda en la variable elección, como la elección es 2 le manda el mensaje de:

"\n\tIngrese la denominacion y si esta activo\n\ten un formato (denominacion, activo): \nDenominacion:\n'c' traera todos los tipos que empiezan con c.\n'jarabe' para traer el tipo jarabe.\n'@' para no aplicar filtro\nActivo:\n's' traera todos los Activos.\n'n' traera todos los Inactivos.\n'@' para no aplicar filtro"

Como la elección sigue siendo 2 al recibir los datos del cliente, se llama a la función administrarTipo y se le pasa por parámetro los datos recibidos, si hay algún problema la elección pasa a ser 10 lo cual hace que al momento de enviar solo le envíe el mensaje de error que dio la función.

Si está todo bien, la elección pasa a ser 8 lo cual hace que se le envíe otro mensaje al cliente y a la hora de recibir pase al case 8 del switch que llama a la función modificarTipo que le pide cuales son los datos a modificar y se fija si el id, que le mando por parámetro el cliente, es correcto. De ser así, se setea la elección como 9 y se le envía el mensaje con la petición al cliente y de este se recibe otros datos.

De nuevo en el switch del recibir como la elección es 9 se accede al case 9, este le pide concretamente cuales son los datos a modificar, llamando a la función pedirDatosTipo, hace algunas validaciones por si hay algún error y si esta todo bien setea la elección como 11. Se envía el correspondiente mensaje y entra al case 11, si lo que recibe es ‘s’ se guarda los datos alterados del tipo de medicamento, llamando a la función guardoTipo y escribo en el archivo del usuario que este ha realizado una acción, llamando a la función escribirArchivoUserAccion.

Finalizando explicare alguna que otra función, para poder observar cómo están estructuradas y como trabajan.

\*pedirDatosMed: Esta función recibe por parámetro la información que le llegue del cliente, es decir el buffer mismo. Para poder interpretar bien la información que le llega transforma dicha información a un string y se fija su tamaño, una vez hecho esto recorre toda la información y calcula cuan larga es cada una de las variables que recibe, utilizando contadores y stops. Una vez identificadas las cantidades, se procede a guardar los datos obtenidos en variables auxiliares y se transforma aquellas que sean necesarias a todo en mayúscula (este proceso se repite en cada vez que se necesite interpretar la información que le llega a la función por parámetro). Después, se asigna a la variable auxiliar de medicamento los valores de la unidad de vector de medicamentos seleccionada.

Posteriormente, se declara una variable de tipo string llamada mensaje que permitirá enviar un mensaje con las distintas respuestas dependiendo lo que devuelva los posteriores pasos. Siguiendo con el método, hay muchos if que se fijan cada una de las posibilidades correspondiente a los datos que ha recibido, llamando funciones que se fijan si no hay ningún error, y si no hay problemas se procede a actualizar el auxiliar de medicamento para posteriormente sobrescribir (en otra función) los datos en el archivo binario correspondientes al medicamento seleccionado.

\*addTipo: Lee los tipos de medicamentos existentes, después abre el archivo binario de tipo de medicamento para así poder actualizarlo de ser necesario. Como recibe por parámetro únicamente el tipo, no necesita interpretar la información, lo que hace es convertir el tipo que recibe en mayúsculas, después abre un if que llama a la función compararTipo(tipo) que se fija si existe el tipo seleccionado, y de ser así manda el mensaje:

"\nEl tipo de medicamento: "+ tipo+"\nya existe, por favor ingrese otra denominacion:"

Si no da error el if, se aumenta el contador de tipo de medicamentos y se setea el tipo nuevo con los datos pertinentes, se guarda en el archivo y por último devuelve el id.