línea horizontal

**Milagros Stephanie Sassenus**

Server

**25 de octubre del 2021**

# VISIÓN GENERAL

El ejecutable del server se encuentra en la carpeta “sockets”. El ejecutable llamado “server\_inicial.exe” debe ejecutarse para poder acceder con la aplicación cliente.

# CARPETAS RELACIONADAS

1. sockets. Es la carpeta raíz de la aplicación. Dentro de ella hay dos subdirectorios más, uno correspondiente a “bin”, contiene los cpp, y otro a “headers”, que contiene los .h. Dentro de bin se encuentra la lógica de los logeos,”login.cpp” , y la del menú en general, “server.cpp”.
2. medicamento. En esta carpeta se encuentra la estructura “medicamento”. Lo tomamos como si fuera una clase, que tiene los crear, set, get, delete, toString() y array\_to\_object(). También incluye funciones como el verificar\_codigo() y reemplazar\_vacio()
3. tipo\_medicamento. En esta carpeta se encuentra la estructura “tipo\_medicamento”. Lo tomamos como si fuera una clase, que tiene los crear, set, get, delete, toString() y array\_to\_object(). También incluye funciones como el reemplazar\_vacio()
4. user. En esta carpeta se encuentra la estructura “user”. Lo tomamos como si fuera una clase, que tiene los crear, set, get, delete, toString() y array\_to\_object().
5. documentacion. En esta carpeta se encuentran todos los archivos referentes a los “logs”. También, se encuentran distintos archivos lógicos, los que son :
   1. get\_files\_binary.cpp . En este apartado se trabaja la lectura y escritura de archivos binarios
   2. get\_files.cpp . En este apartado se trabaja la lectura y escritura de archivos txt
   3. listas.cpp. En este apartado se trabaja el manejo de listas que están compuestas por un string (content).
   4. util.h. Este archivo es incluido por TODOS los cpp anteriormente nombrados. Incluye funciones lógicas que se utilizan, en su mayoría, en medicamento y tipo\_medicamento.

# ESPECIFICACIONES

* Para cambiar el puerto de escucha de un server, se debe acceder a “server.h” y cambiar el valor de DEFAULT\_PORT
* En caso de querer utilizar algún tipo de archivo distinto, cambiar sus rutas en DIRECCION\_BINARIO\_MEDICAMENTO y DIRECCION\_BINARIO\_TIPO

# FUNCIONES

## Enviar y recibir información

En cualquier caso, la información se recibe y se envia como un string. A partir de eso, la información se manipula para obtener el resultado querido.

“int iSendResult = send( socket, respuesta.c\_str(), respuesta.size(), 0 )

int iResult = recv(socket, recvbuf, recvbuflen, 0);”

Donde iSendResult son los bytes que se envían y iResult son los bytes recibidos. Respuesta equivale a un string, y recvbuf equivale a un char\*. recvbuflen es un int.

Para saber más, chequear las funciones obtener\_ingresado\_Array\_por\_cliente() y obtener\_ingresado\_numerico\_por\_cliente () en “server.cpp”.

**Log-in**

Para el log-in, se utiliza el documento “usuarios\_y\_contrasenas.txt” ubicado en la carpeta documentacion

Este paso es el primero que visualiza el cliente, y el indispensable para ingresar al menu. La funcion successfull\_login() es la que maneja esto, y necesita que se le pase una lista con los usuarios disponibles. Se basa en pedir al cliente que ingrese un nombre y una contraseña, estos datos se pasan a string y se buscan en la lista. Si coinciden se pasa al próximo paso, sino se pide de nuevo el dato. Al llegar a los 3 intentos fallidos, la función devuelve -1. Cuando se ingresa todo correctamente, devuelve el id del usuario. En caso de devolver -1, se desconecta al cliente.

## Menú

La función del menu es llamada menu\_socket() y devuelve un int. Hay dos maneras de salir del menú: cerrando sesión o por time-out. En caso de cerrar sesión, devuelve 0. Caso contrario, devuelve -1.

Dentro del menú uno puede acceder a cuatro opciones: tipo medicamento, medicamento, ver registro y cerrar sesión. Las dos primeras, incluyen un sub menú que incluye: crear, modificar, borrar / dar de alta.

## Manejo de medicamentos

Casi todo lo que se hace en el menú se atribuye a medicamento.cpp , en la carpeta medicamento.

La utilización de crear\_Medicamento\_por\_consola:

Con respecto a la selección de la creación de un medicamento, se pide al usuario ingresar los dato acomodados en un switch. El loop es controlado por la variable veces, y cada vez que se corre el loop, se le suma 1.

1. Droga
2. Código. Se verifica con la función verificar\_codigo(). En caso de ser incorrecto, veces -1. Para salir del loop, se puede presionar ENTER o ingresar un código correcto
3. Nombre
4. Tipo medicamento. En este paso, se verifica si el tipo medicamento existe. En caso de no existir, ingresa en un loop hasta que se ingrese un tipo de medicamento existente. En este paso, también se verifica que el conjunto nombre-tipo medicamento no se haya utilizado en otro medicamento. En ese caso, veces -2 y se reingresa el nombre y tipo de medicamento.

Luego de pedir los datos por consola, se devuelve un crearMedicamento(droga,codigo,nombre,tipo medicamento). El ID que es asignado por la función utiliza getLines() para contar las lineas en el archivo binario. El ID resultaría en las lineas que hay en el archivo binario + 1. El is\_active de medicamento, por default en creación, es verdadero.

Aclaración: para todos los casos, si se ingresa un ENTER (‘\0’) se manda un mensaje de error y se vuelve al submenú.

Otra aclaración : para evitar problemas, la droga, el nombre y el tipo de medicamento se pasan a mayúsculas con la función to\_uppercase()

La administración de medicamentos tiene tres pasos :

1. Buscar: el buscar usa el mostrar\_lista\_med\_filtrada\_y\_obtener\_id() a la cual se le debe pasar una lista con los medicamentos en el binario. Primero ingresa a un loop de 2 veces.
   1. En la primera vez, se muestra “Ingrese criterio de búsqueda según nombre. Si no se desea filtrar, ingrese @”. En el caso de ingreso de “@” veces + 1 . Caso contrario, lista = filtrarNombre() y veces + 1
   2. En la segunda vez, se muestra “Ingrese criterio de busqueda segun Tipo de medicamento. Si no se desea filtrar, ingrese @” . En el caso de ingreso de “@” veces + 1 . Caso contrario, lista = filtrarTipoMedicamento() y veces + 1

A partir de esto, obtenemos la lista filtrada. Si la lista esta vacía, se informa y la mostrar\_lista\_med\_filtrada\_y\_obtener\_id() devuelve -1. Caso contrario, se pide que se ingrese un id. Si el id es mayor a -1 , se devuelve ese id.

1. Modificar y Eliminar/Dar alta: luego de la búsqueda, se pide que se ingrese una opción (modificar\_eliminar\_medicamento())
   1. Caso modificar: se llama a la función de crear\_Medicamento\_por\_consola() y se lo guarda en una variable. Luego, se llama a la función reemplazar\_vacio() que reemplaza los “@” por el medicamento del id ingresado en la búsqueda.
   2. Caso eliminar/dar alta: se cambia el is\_activo por su contrario (!is\_activo)
2. Guardar cambios: dentro de la función modificar\_eliminar\_medicamento() se pregunta si se desean guardar los cambios. En caso verdadero, se retorna el objeto cambiado. Caso contrario, se retorna NULL

A partir de esto, el menú maneja que si recibió un medicamento, modifica en el binario. Caso contrario, no se modifica nada.

## Manejo de tipo medicamentos

Casi todo lo que se hace en el menú se atribuye a tipo\_medicamento.cpp , en la carpeta tipo\_medicamento.

La utilización de crear\_Tipo\_Medicamento\_por\_consola:

Con respecto a la selección de la creación de un tipo de medicamento, se pide al usuario ingresar el nombre del mismo. Luego de pedir el datos por consola, se devuelve un crearTipoMedicamento(nombre). El ID que es asignado por la función utiliza getLines() para contar las lineas en el archivo binario. El ID resultaría en las lineas que hay en el archivo binario + 1.

Aclaración : la denominación se pasan a mayúsculas con la función to\_uppercase()

La administración de tipo de medicamento tiene tres pasos :

1. Buscar: el buscar usa el mostrar\_lista\_tipo\_filtrada\_y\_obtener\_id() a la cual se le debe pasar una lista con los medicamentos en el binario. Primero ingresa a un loop de 2 veces.
   1. En la primera vez, se muestra “Ingrese criterio de búsqueda según denominacion. Si no se desea filtrar, ingrese @”. En el caso de ingreso de “@” veces + 1 . Caso contrario, lista = filtrarDenominacion() y veces + 1
   2. En la segunda vez, se muestra “Ingrese criterio de busqueda segun activo. Si no se desea filtrar, ingrese @” . En el caso de ingreso de “@” veces + 1 . Caso contrario, lista = filtrarActivo() y veces + 1

A partir de esto, obtenemos la lista filtrada. Si la lista esta vacía, se informa y la mostrar\_lista\_tipo\_filtrada\_y\_obtener\_id() devuelve -1. Caso contrario, se pide que se ingrese un id. Si el id es mayor a -1 , se devuelve ese id.

1. Modificar y Eliminar/Dar alta: luego de la búsqueda, se pide que se ingrese una opción (modificar\_eliminar\_medicamento())
   1. Caso modificar: se llama a la función de crear\_Tipo\_medicamento\_por\_consola() y se lo guarda en una variable. Luego, se llama a la función reemplazar\_vacio() que reemplaza los “@” por el tipo medicamento del id ingresado en la búsqueda.
   2. Caso eliminar/dar alta: se cambia el is\_activo por su contrario (!is\_activo)
2. Guardar cambios: dentro de la función modificar\_eliminar\_tipo() se pregunta si se desean guardar los cambios. En caso verdadero, se retorna el objeto cambiado. Caso contrario, se retorna NULL

A partir de esto, el menú maneja que si recibió un tipo medicamento, modifica en el binario. Caso contrario, no se modifica nada.

## Escritura de logs

Cuando se inicia el servidor, se conecta un cliente y desconecta, todo eso se escribe en server.log. Para su escritura se utiliza la funcion escribir\_server\_log() que utiliza write\_txt((getFechaHora() + "=======================================================") , DIRECCION\_SERVER\_LOGS);

write\_txt((getFechaHora() + "-" + content) , DIRECCION\_SERVER\_LOGS);\

getFechaHora () devuelve la fecha del servidor en ese momento.

write\_txt() es un printf() a un archivo.

Luego de terminar una operación y guardar los cambios correctamente (incluye inicio y cierre de sesión), se llama a escribir\_log(), funciona similarmente a la función anterior, que guarda los cambios en un archivo único para el cliente.

## Manejo de time-out

Con respecto al time-out, cuando por primera vez se ingresa el socket se le setea un time out de 2 minutos. Para setear ese time out, utilizamos setsockop() en la función de recv(). Si no recibimos nada en dos minutos, la función recv() devuelve -1. A partir de ahí, el menú devuelve -1 y se crea un nuevo thread, que verificará si hay conexiones entrantes. Si hay conexiones entrantes, se desconecta al cliente que se quedó timed-out . Caso contrario, se setea un nuevo time-out de 4 segundos, ya que recv() es una función bloqueante. Serán dos loops que se correran al mismo tiempo, el thread(al cual le asignaremos un sleep de 4 segundos) y uno que es un recv() simple . Si el thread es el primero que recibe respuesta, desconecta al cliente. Caso contrario, se cierra el thread y se vuelve a ingresar al menú.

## Manejo de otras conexiones

Para manejar otras conexiones, se abre un thread. Este thread acepta a un nuevo cliente solo para enviar un mensaje de que el servidor está ocupado, y luego desconecta al cliente nuevo. En caso de time-out, este thread se deshabilita hasta que se resuelva.

**Milagros Stephanie Sassenus**

ReadBinary

**25 de octubre del 2021**

# VISIÓN GENERAL

El ejecutable del readBinary se encuentra en la carpeta “bin” (la general)

# CARPETAS RELACIONADAS

1. documentacion. En esta carpeta se encuentran todos los archivos referentes a los “logs”. También, se encuentran distintos archivos lógicos, los que son :
   1. get\_files\_binary.cpp . En este apartado se trabaja la lectura y escritura de archivos binarios
   2. get\_files.cpp . En este apartado se trabaja la lectura y escritura de archivos txt
   3. listas.cpp. En este apartado se trabaja el manejo de listas que están compuestas por un string (content).
   4. util.h. Este archivo es incluido por TODOS los cpp anteriormente nombrados. Incluye funciones lógicas que se utilizan, en su mayoría, en medicamento y tipo\_medicamento.

# ESPECIFICACIONES

El programa lee los Medicamentos y Tipo medicamentos juntos.

# FUNCIONES

## Leer el binario

Para la lectura, se devuelve una lista de string. Se utiliza

while (1==fread(&res,200,1,fpa))

insertarFinal(lista,std::string(res));

## Mostrar Lista

## Para mostrar la lista, se utiliza el mostrarLista() que va de nodo en nodo, haciendo un printf() de cada content.

## 

## Milagros Stephanie Sassenus

Cliente

**25 de octubre del 2021**

# VISIÓN GENERAL

El ejecutable del cliente se encuentra en la carpeta “cliente” y se llama “main.exe”. Para poder ejecutar el cliente, se debe saber la ip y el puerto de escucha del server.

# CARPETAS RELACIONADAS

1. cliente . Es la carpeta raíz de la aplicación. Dentro de ella hay dos subdirectorios más, uno correspondiente a “bin”, contiene los cpp, y otro a “headers”, que contiene los .h. Dentro de bin se encuentra la lógica de los logeos,”cliente.cpp”.

# FUNCIONES

## Inicializar cliente

Se utiliza la función initialize\_client(), a la cual se le pasa la direccion y el puerto de server pasados por consola. Devuelve un cliente inicializado.

## Recibir y enviar información

En cualquier caso, la información se recibe y se envia como un string. A partir de eso, la información se manipula para obtener el resultado querido.

“int iSendResult = send( socket, respuesta.c\_str(), respuesta.size(), 0 )

int iResult = recv(socket, recvbuf, recvbuflen, 0);”

Donde iSendResult son los bytes que se envían y iResult son los bytes recibidos. Respuesta equivale a un string, y recvbuf equivale a un char\*. recvbuflen es un int.

Esto va a estar en un while () mientras que el sendResult no resulte en un error de socket (-1)

## Manejo server ocupado

En caso de que el server esté ocupado, se muestra un “desea volver a intentar” . Si se ingresa si, se intenta volver a conectar. Caso contrario, cierra el socket cliente y termina la aplicación

## Shut down

La aplicación se cerrará cuando el servidor deje de dar respuesta o, en caso de nunca haberse conectado al servidor, se cierra cuando se ingresa que no se desea volver a conectar. También se finaliza cuando las credenciales ingresadas (dirección ip y puerto del server) son incorrectas y devuelve “getaddrinfo failed”

## 