Trabajo Práctico Redes Y Comunicaciones 2020

**Integrantes:**

**Ariel Nicolás Heredia DNI: 33691227**

**Quimey Emmanuel Pérez DNI: 41237009**

**Oscar Ruina DNI: 35639606**

**Nicolás Romero Ortiz DNI: 40136661**

**Luciano Moliterno DNI: 40238958**

**Link del video->>**

**https://drive.google.com/drive/folders/1AGMVyeAjFA41T1Lf8BiTGyS2-FOzsLxT**

**Resolución**

**Servidor:**

Primero creamos la clase Servidor que se va a encargar de inicializar el servidor, y poder enviar y recibir mensajes. En el método constructor inicializamos las librerías necesarias para poder manejar sockets, obtenemos la dirección ip que usara el servidor, en este caso será la dirección local. Creamos el socket y el puerto de escucha, asociamos el socket con el puerto, una vez hecho esto ponemos a escuchar al socket servidor por conexiones entrantes. Cuando entre una nueva conexión la aceptamos e indicamos desde que ip viene esa conexión.

Para el manejo de envió y recibo de mensajes entre sockets, dentro de la clase Servidor tenemos dos métodos que se encargan de eso. El método std::string newRecibir() recibe los mensajes del socket cliente, si recibe un mensaje con la etiqueta “x” el servidor se reinicia sino internamente va contando el tiempo de respuesta que tarda el cliente en enviar un mensaje. El método void enviar(string msg) se encarga de enviar mensajes al socket cliente, si el tiempo de respuesta supero los dos minutos el servidor envía un mensaje diciendo que se superó el tiempo de espera y corta la conexión, sino manda el mensaje y continua con la comunicación. El método void Reiniciar(), reinicia el socket y el tiempo de respuesta.

En el método main que se va a ejecutar es en donde creamos una variable de clase Servidor y llamamos al constructor. Luego de esto en un bucle while es en donde se realizara todo el manejo de envió y recibo de mensajes entre servidor y cliente mediante varias funciones. El servidor siempre recibe del cliente y luego envía respuesta.

Lo primero que hace el servidor es enviarle al cliente mensajes preguntando por el usuario y contraseña, cuando recibe la respuesta la función int leerArchivosUsuario(string usuarioContraseña) se encarga de leer el archivo de usuarios en busca de una coincidencia, si no hay coincidencia retorna un 0 y un 1 cuando encuentra el usuario y la contraseña, de vuelta en el main si no se encontró el usuario en 3 intentos se envía al cliente un mensaje de se superó la cantidad máxima de intentos permitidos, si se encontró antes de superar los 3 intentos el servidor envía el menú con las opciones correspondientes.

Las función void generarViajes(Servidor \*server,string usuario) se encarga de guardar en un archivo binario nuevos viajes, recibe del cliente los datos del viaje a guardar y verifica que el viaje ingresado no exista, si existe le envía al cliente un mensaje y si no existe lo guarda en el archivo binario. Para verificar que exista el viaje, lee el archivo en busca de un viaje igual, retornando 0 si no encontró uno igual y 1 si encontró uno igual. Una vez verificado que no exista se escribe el archivo con el viaje nuevo, con sus respectivos asientos, esto en un archivo diferente, uno por cada viaje.

La función void gestionarServicios(Servidor \*server, std::string usuario) se encarga de todo lo relacionado con la búsqueda de viajes, reservas y liberación de asientos de cada viaje. Genera un menú que se le envía al usuario, este menú contiene las opciones para buscar los viajes por origen, fecha, turno y las combinaciones de estas. Para esto se lee el archivo de viajes y se buscan las coincidencias con los datos enviados por el cliente. Se le informa al cliente cuales son los viajes y espera a que el cliente elija uno. Una vez elegido el viaje se le envía al cliente los asientos y el menú de reserva y liberación de asientos. La función void reservarAsiento(Servidor\* server,string servicio,string usuario) se encarga de reservar el asiento que quiere el usuario, si ya está reservado le envía un mensaje de asiento ya reservado, sino lo reserva y se le vuelve a mostrar los asientos, una x indica asiento ocupado, una o indica asiento libre. La función void liberarAsiento(Servidor\* server,string servicio,string usuario) hace lo opuesto a reservar, si el asiento está libre informa esto al usuario, en caso contrario lo libera.

La función void verRegistroActividades(Servidor \*server,std::string usuario) envía al cliente todo lo que hizo, se lee el archivo de usuario línea a línea y se va agregando a una variable de tipo string que será enviada al cliente.

La función void CerrarSesion(Servidor \*server,std::string usuario) envía al cliente un mensaje con la etiqueta “x” , se reinicia el socket servidor.

Función void log(string archivo, string msg), se encarga de guardar todo lo que pasa en el servidor y lo que hace cada cliente.

Cada envió de mensaje de parte del servidor va con una etiqueta que el cliente verifica cuando la recibe. “x”,”M”,”N”,”R”,”B”.

**Cliente:**

Al igual que en el servidor creamos la clase Cliente, para inicializar el socket, obtenemos la ip que va a utilizar, en este caso la ip local, creamos el socket, el puerto, asociamos el socket con el puerto y conectamos el socket con el servidor. Si el puerto del cliente es distinto al del servidor se cierra el socket cliente. También tenemos dos métodos para enviar y recibir mensajes del servidor.

Una función importante es int verificarRecibir(string msg) que verifica las etiquetas recibidas del servidor, si es x se cierra el socket, B para mostrar los asientos de los viajes, R para los menús, y N para mensajes en los que no se necesita responder(esto evita bugs en el while). Cuando se recibe una etiqueta “B”, se llama a la función void mostrarAsientos(string asientos) que le muestra al cliente los asientos del servicio elegido.

En el main se crea la variable de clase Cliente, se pide el puerto para conectarse al servidor y se entra a un while en donde se envía y recibe mensajes del servidor. El cliente siempre recibe mensajes del servidor y envía solo cuando el verificar lo permite.

**Estructura de los Servicios:**

Creamos una clase para los servicios con dos atributos de tipo string, servicio y asientos, en servicio se guarda el origen, destino, fecha y turno del servicio. Al recibir por parte del cliente el servicio el servidor se encarga de de serializar el servicio en dos partes: el servicio con una longitud de 16 bits y los asientos que van desde los bits 16 al 393, para así instanciar un objeto de tipo Servicio que recibe dos strings. Esto permite que la búsqueda de servicios y el traer asientos de un servicio determinado se pueda realizar de una manera más fácil, como también mostrar todo el contenido del archivo binario con una aplicación aparte.

Ejemplo de lectura en una aplicación aparte en donde se muestra uno a uno los servicios creados en el archivo binario.





