****

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería En Ciencias de la Computación

Materia:

Modelo de Redes

Trabajo:

Documentación Chat Java

Maestra/o:

Dr. Iván Olmos Pineda

Alumno:

* Ruiz Lozano Paulo Cesar 201727952
* Díaz Romero Ignacio 201734055

Introducción

El programa chat java, hace una simulación a nivel de la capa de enlace de datos con un programa Cliente y otro Servidor. El Servidor recibe peticiones de “loggin”, busca si el usuario existe en la base de datos, genera un mensaje aleatorio y mezcla el mensaje con la contraseña generando un MD5 que se va a comparar con el MD5 del Cliente, si son iguales entonces se establece una conexión. Todos los mensajes que se envían entre cliente y servidor son encriptados en Base64.

Funcionamiento del programa

El programa necesita ser compilador bajo java 11 o superior, se puede usar un IDE como NetBeans, pero en la carpeta del programa o en el repositorio<https://github.com/Fatake/Chat-Java> en la carpeta /src existen dos carpeta una /cliente y otra /servidor

* Carpeta /Servidor
  + Contiene un script ejecutable por lotes de Windows (ser.bat) que ejecuta estos comandos
    - javac -encoding "UTF-8" -d class Servidor.java
    - java -cp class Servidor
* Carpeta / Cliente
  + Contiene un script ejecutable por lotes de Windows (cli.bat) que ejecuta estos comandos
    - javac -encoding "UTF-8" -d class ClienteGUI.java
    - java -cp class ClienteGUI

El servidor inicia y genera varios hilos para cada petición que existe, el hilo que atiende peticiones siempre está escuchando peticiones, el paso de mensajes ente cliente y servidor se envía como una cadena con la siguiente estructura: Cadena mensaje = “código,valor”. Los códigos que se envían son:

* us,usuario: El cliente recibe un mensaje del usuario a buscar, comprueba si existe o no, si existe procede a enviar un mensaje aleatorio, si no envía in no existe.
* un,null: El cliente recibe un código un en caso de que el usuario no exista.
* ms,mensajeAleatorio: El cliente recibe un mensaje aleatorio del servidor, el cliente procede a mezclarlo con la contraseña proporcionada con el usuario y genera un MD5 que envía al servidor.
* md,MD5Usuario: El servidor recibe un md5 del usuario y procede a compararlo con el md5 que el genera con el mensaje aleatorio y contraseña dentro del servidor. De ser correcto el servidor envía un código de cn de conectado donde ese establece la “conexión” con el cliente. Caso contrario envía un código nn de no conectado.
* cn,Conectado: El cliente recibe un código cn si la contraseña es exitosa.
* nn,NoConectado: El cliente recibe este código si la contraseña es incorrecta.
* fn,null: El servidor recibe este código para fin de conexión.
* ls,null: Cuando existe una conexión exitosa y el cliente necesita saber la lista de amigos, el servidor se la envia.
* Ac,nuevaLista. Mensaje que envía el cliente al servidor para actualizar la lista de amigos de la conexión.
* Ms,env/re,origen,destino,mensaje: Cuando el servidor recibe este mensaje primero divide si se trata de recibir (re) o enviar mensajes (env): separa el origen y destino y el mensaje y estos datos se los pasa al Hilo maestro de comunicación entre hilos (explicado más adelante).

Manejo de mensajes entre Usuarios

El Servidor también inicializa un subproceso un “Hilo Maestro de comunicaciones” que se encarga de iniciar las colas de mensajes entre los hilos cliente. Cada hilo cliente tiene su propia cola de mensajes que se puede acceder con su ID para leer o enviar mensajes.

Desarrollo

El programa consta de 2 packages:

* **Cliente:** Que contiene las clases
  + *ClienteGUI*: Clase principal del cliente, es la interfaz gráfica del usuario. En ella se inicia sesión, se agrega amigos a tu lista de amigos desde la lista de todos los usuarios en la red.(JFrame)
  + *Conversación*: Esta clase hereda de JFrame, en donde mantiene la conversación con otro usuario. Actualizando la lista de mensajes y ordenando los mensajes de forma correcta.
  + *ItemAgregar*: Esta clase hereda de JPanel, en ella se tiene un JLabel y un JButton. Esta se llama dentro de “ClienteGUI” la cantidad de usuarios NO amigos del usuario loggeado para que estos puedan ser agregados.
  + *ItemAmigo*: Esta clase hereda de JPanel, es llamada dentro de “ClienteGUI” la cantidad de usuarios amigos del usuario loggeado. Mediante el botón de este panel se accede a “Conversación”.
  + *ItemMensaje*: Esta clase hereda de JPanel, es llamada dentro de “Conversación”. Se utiliza para mostrar un mensaje de la cola de mensajes del usuario loggeado.
  + *MD5*: Clase que recibe como atributo un String, a esta le genera el MD5 y lo retorna en forma de String.
  + *Mezclador*: Clase que recibe como atributos 2 Strings y genera un String resultante de mezclar las dos.
  + *Usuario*: Clase donde se almacena la información del usuario, nombre y contraseña.
* **Servidor:** Contiene las siguientes clases
  + *ColaHilo*: Clase que guarda la Cola del hilo, esta se comunica con HiloComunicador para recibir y enviar mensajes.
  + *GestorPeticion*: Clase que administra las peticiones de cada usuario.
  + *HiloComunicador*: Clase que hereda a Thread(Hilo). Encargada de administrar las colas de los hilos de petición de los mensajes.
  + *LectorArchivo*: Clase encargada de procesar el archivo “usuaríos.dat” donde se encuentran los datos de los usuarios.
  + *MD5*: Clase que recibe como atributo un String, a esta le genera el MD5 y lo retorna en forma de String.
  + *Mezclador*: Clase que recibe como atributos 2 Strings y genera un String resultante de mezclar las dos.
  + *Servidor*: Clase principal que se encarga de estar escuchando peticiones y por cada petición/usuario nuevo que se necesite genera un hilo(GestorPeticion) para que se encarge de las operaciones del usuario.
  + *Usuario*: Clase donde se almacena la información del usuario. Nombre, contraseña.

Dentro de este package se encuentra el archivo “usuaríos.dat”, donde se almacenan los datos de los usuarios registrados.

Clases Cliente:

ClienteGUI

Atributos

* Listatodos: Atributo ArrayList<String> que almacena la lista de todos los usuarios obtenida del servidor

Constructor

* Se inicializan todos los componentes gráficos

Métodos

* actualizarListaAmigos(): Actualiza la lista de amigos del usuario loggeado
* actualizarListaNoAmigos(): Actualiza la lista de NO amigos del usuario loggeado
* desencriptar():Este método recibe un string en base 64 y retorna el string original.
* encriptar():Este método recibe un string y retorna un string en base 64 encriptado.
* Main(): Método que gestiona la conexión al Servidor, enviendo y recibiendo los parámetros como, el usuario, recibiendo cadena de texto aleatorio, generar un md5 con la mezcla del texto aleatorio y la contraseña del usuario y enviando el md5 para generar o no una conexión de “login”.
* BotonIniciarSesionMouseClicked(): hace visible el frame donde se pide el nombre de usuario y contraseña para poder loggear
* botonLogeoMouseClicked(): Verifica comunicandose con el servidor si los fatos ingresados pertenecen a un usuario. Utilizando MD5 y encriptacion.
* Setters and Getters de Componentes Gráficos.

Conversación

Atributos

* nombreAmigo: Atributo cadena donde se almacena el nombre del usuario amigo con el que se esta teniendo la conversacion
* cliente: Atributo ClienteGUI donde se almacena a que cliente corresponder dicha conversacion
* usuario: Atributo Usuario donde se almacena el nombre y contraseña del usuario loggeado
* o: Atributo ItemMensaje que se utiliza para actualizar la lista de mensajes de la conversación.

Constructores

* Inicializa los componentes gráficos y recibe una cadena que se le asigna al atributo “nombreAmigo”, y un ClienteGUI que se le asigna a “Cliente”.

Métodos

* botonEnviarActionPerformed(): Obtiene el mensaje del Textbox y lo manda comunicándose con el servidor al usuario con el que se tiene la conversacion.
* formMouseMoved(): Actualiza el panel donde tenemos la lista de mensajes.

ItemAgregar

Atributos

* C: Atributo ClienteGUI donde se almacena a que cliente pertenece este item.

Contructores

* Inicializa los componentes gráficos y recibe una cadena, que se la asigna al label donde se almacena el nombre del usuario NO amigo, y un ClienteGUI que se lo asigna a “c”.

Métodos

* AgregarbotonActionPerformed(): Agrega a la lista de amigos del usuario loggeado en el clienteGUI y luego actualiza la lista de amigos y no amigos.

ItemAmigo

Atributos

* C: Atributo ClienteGUI donde se almacena a que cliente pertenece dicho item.
* Conv: Atributo Conversación que se tendrá con el usuario amigo seleccionado.

Constructores

* Inicializa los componentes graficos y recibe una cadena, que se la asigna al label donde se almacena el nombre del usuario amigo.

Métodos

* botonMensajearActionPerformed():Crea una nueva conversacion y se la asigna a “Conv”, luego hace visible “Conv” y hace invisible “C” mientras Conv siga abierta.

ItemMensaje:

Atributos

* No tiene

Constructores

* Inicializa los componentes graficos y recibe una cadena que se la asigna al label donde almacena el mensaje.

Métodos

* No tiene mas que los que un JPanel tiene por default.

Usuario

Atributos

* Nombre: atributo cadena privado que almacena el nombre del usuario.
* Password: atributo cadena privado que almacena la contraseña del usuario

Constructores

* No tiene.

Métodos

* getName(): Retorna una cadena el nombre del usuario.
* getPass(): Retorna una cadena la contraseña del usuario.
* toString(): Retorna en cadena la información del usuario.

Mezclador

Atributos

* No tiene

Constructores

* No tiene

Métodos

* Mezcla(): recibe un texto A que es el mensaje aleatorio, y un texto B, ingresa el texto b dentro del texto a separado por cada tamaño del texto B sobre 2.

MD5

Atributos

* INT\_A: variable con el valor hexadecimal 0x67452301.
* INT\_B: variable con el valor hexadecimal 0xEFCDAB89L.
* INT\_C: variable con el valor hexadecimal 0x98BADCFEL.
* INT\_D: variable con el valor hexadecimal 0x10325476.
* SHIFT\_AMTS: arreglo con los valores 7, 12, 17, 22, 5, 9, 14, 20, 4,11, 16, 23, 6, 10, 15, 21.
* TABLE\_T: Arreglo que tiene valores de una función sin().

Constructores

* No tiene

Métodos

* CalculaMD5(): recibe un arreglo de bytes que es el mensaje a general el MD5.
* toHexString(): recibe un arreglo de bytes a cambiar a String.
* getMD5(): Retorna el MD5 generado en forma de cadena.

Clases Servidor:

ColaHilo

Atributos

* colasMensajes: Atributo ConcurrentHashMap<> donde se almacena la cola de mensajes del usuario loggeado.
* CAPACIDAD: Atributo int que indica el tamaño de la cola de mensajes.
* ID\_COMUNICADOR: Atributo int que indica el ID de conexión con el Comunicador de Hilos

Constructores

* No tiene

Métodos

* EnviarMensaje(id,mensaje): Se ingresa a la cola el mensaje que se recibió del Comunicador
* RecibirMensaje(id): Se toma de la cola el mensaje que se recibio

GestorPeticion

Esta clase es hija de la clase Thread o hilo

Atributos

* Usuarios: Lista ligada de usuarios.
* Entrada: Buffer lector.
* Salida: permite escribir al socket.
* Socket: Socket cliente.
* Comunicador: ColaHilo de este hilo donde se almacenan los mensajes

Constructores

* Recibe un Socket cliente, una lista de usuarios y una ColaHilo.

Métodos

* Run(): Se encarga de leer y escribir en el buffer, es la transmisión de datos entre el cliente y servidor, compara si existe el usuario en el servidor, genera un texto aleatorio, mezcla el texto, envía el texto aleatorio, compara el MD5 del cliente con el del servidor y permite o no una conexión si son iguales. Escucha si se genera una nueva conversación y manda los mensajes al comunicador de hilos.
* buscaUsuario(): Función que busca un usuario y retorna el índice de este en el arreglo.
* generaTexto(): Función que genera un texto aleatorio. De 586 caracteres
* desencriptar():Este método recibe un string en base 64 y retorna el string original.
* encriptar():Este método recibe un string y retorna un string en base 64 encriptado.

ComunicadorHilo

Atributos

* Usuarios: Arraylist donde se guarda todos los usuarios
* Comunicador: ColaHilo donde se guardaran todos los mensajes que se quieran mandar en todas las conversaciones.

Contructores

* Recibe una lista, que se le asigna a usuarios, y una ColaHilo que se la asigna a Comunicador.

Metodos

* Run(): Se encarga de recibir los mensajes y mandar a las colas de cada usuario los mensajes que les correspondern.
* buscaUsuario():Busca dentro de la lista un usuario especifico.

LectorArchivo

Atributos

* ContendioArchivo: lista ligada que tiene todo el archivo

Constructores

* No tiene

Métodos

* leerArchivo(): función privada que lee un archivo pasado por parámetros y almacena el contendio en la variable contendioArchivo.
* procesaArchivo(): Genera una lista ligada de todos los usuarios de la “base de datos”.

Conclusión

Mediante el desarrollo de este proyecto comprendí como funcionan hoy en día los chats de comunicación, los canales y medios que se utilizan para que la comunicación entre dos o mas equipos de computo sea segura y confiable. Por encriptación y huella digital es una manera muy segura de proteger la información que se esta intercambiando para que de esta manera no haya fuga de información. *Ignacio Díaz Romero*