Proyecto Chat

IFCT-609: PROGRAMACIÓN DE SISTEMAS INFORMÁTICOS

GRUPO : Míldred Ramírez, Cristian Echauri, Houssam Amrouch, Mateo Crespí, Javier Palacios

Proyecto en GitHub:

https://github.com/helkyar/ChatUpp

Índice

1. Informe Legal

Este software o hardware está desarrollado para uso general en una variedad de aplicaciones de administración de información. No está desarrollado ni diseñado para su uso en ninguna aplicación intrínsecamente peligrosa, incluidas las aplicaciones que pueden crear un riesgo de lesiones personales. Si usa este software o hardware en aplicaciones peligrosas, entonces será responsable de tomar todas las medidas apropiadas a prueba de fallas, respaldo, redundancia y otras para garantizar su uso seguro. Oracle Corporation y sus afiliadas renuncian a cualquier responsabilidad por los daños causados por el uso de este software o hardware en aplicaciones peligrosas.

Oracle y Java son marcas comerciales registradas de Oracle y / o sus afiliadas. Otros nombres pueden ser marcas comerciales de sus respectivos propietarios.

Intel e Intel Inside son marcas comerciales o marcas comerciales registradas de Intel Corporation. Todas las marcas comerciales de SPARC se utilizan bajo licencia y son marcas comerciales o marcas comerciales registradas de SPARC International, Inc. AMD, Epyc y el logotipo de AMD son marcas comerciales o marcas comerciales registradas de Advanced Micro Devices. UNIX es una marca registrada de The Open Group.

Este software o hardware y documentación pueden proporcionar acceso o información sobre contenido, productos y servicios de terceros. Oracle Corporation y sus afiliadas no son responsables y renuncian expresamente a todas las garantías de ningún tipo con respecto al contenido, productos y servicios de terceros, a menos que se establezca lo contrario en un acuerdo aplicable entre usted y Oracle. Oracle Corporation y sus afiliadas no serán responsables de ninguna pérdida, costo o daño incurrido debido a su acceso o uso de contenido, productos o servicios de terceros, excepto según lo establecido en un acuerdo aplicable entre usted y Oracle.

Esta documentación NO se distribuye bajo una licencia GPL. El uso de esta documentación está sujeto a los siguientes términos:

Puede crear una copia impresa de esta documentación únicamente para su uso personal. Se permite la conversión a otros formatos siempre que el contenido real no se altere o edite de ninguna manera. No debe publicar ni distribuir esta documentación de ninguna forma ni en ningún medio, excepto si distribuye la documentación de una manera similar a como la distribuye Oracle (es decir, electrónicamente para descargar en un sitio web con el software) o en un CD. -ROM o medio similar, siempre que la documentación se difunda junto con el software en el mismo medio. Cualquier otro uso, como la difusión de copias impresas o el uso de esta documentación, total o parcialmente, en otra publicación, requiere el consentimiento previo por escrito de un

representante autorizado de Oracle. Oracle y / o sus afiliadas se reservan todos y cada uno de los derechos sobre esta documentación que no se hayan otorgado expresamente anteriormente.

2. Introducción

Proyecto Chat nació como un proyecto humanitario creado para unir el mundo en paz y armonía. Fué desarrollado por el Grupo 3 (Cristian, Houssam, Javi, Mateo y Mildred).

Este documento explica la estructura de su base de datos y las funciones que provee.

La realización del proyecto en grupo consiste en la creación de un programa enfocado en la **intercomunicación textual y visual.** Programado en Java mediante la aplicación de escritorio NetBeans (Oracle Corporation).

El programa realiza comunicación por sockets entre IPs.

La aplicación Proyecto Chat (CCW) es una aplicación con la finalidad de poder comunicar a los usuarios desde cualquier lugar utilizando Internet mediante mensajería de texto.

Esta aplicación contendrá un sistema encriptado de inicio de sesión de registro con contraseña cuyos datos serán concatenados y validados por un servidor de base de datos.

La base de datos mencionada que será esencial para el funcionamiento del programa se encargará del almacenamiento de los datos del usuario insertados en el registro, los grupos, personas que ha añadido que permiten la comunicación, el historial de conversaciones entre ambos usuarios que hayan realizado previamente. Estará sometida para cumplir la regulación de protección de datos generales de la UE (GDPR).

El usuario deberá tener un login para poder iniciar sesión, por lo que deberá registrarse primero, al darle al botón de Registro saldrá otra pestaña donde le pedirá que nombre de login y contraseña deberá utilizar cada vez que quiera hacer login, además de unos datos de usuario básico. Al finalizar el registro tendrá que insertar su nueva cuenta creada para iniciar sesión.

Al intentar iniciar sesión, se confirmarán los datos previos y si coinciden las contraseñas concatenadas por la encriptación mediante un hash. Entonces el programa procederá a mostrar su menú principal donde mostrará las conversaciones grupales creadas y los usuarios conectados al servicio en esos momentos.

Se podrá interaccionar con estos usuarios para abrir las conversaciones con ellos mostrando su historial de conversaciones previas.

Este programa corre por red local por lo que no requerirá de ningúna lista de amigos, se podrán crear grupos entre los usuarios conectados en la red.

2.1 Integrantes del Equipo

Este proyecto está compuesto por Houssam Amrouch, Mateo Crespí, Cristian Echauri, Javier Palacios y Mildred Ramírez.

2.2 Planificación del proyecto

La estructuración de las tareas a realizar en la duración del proyecto es la siguiente:

Elaboración de diseño de la aplicación (Mildred)
Redacción del proyecto y búsqueda de fuentes (Houssam, Mildred, Mateo)
Gestión de base de datos, encriptación y protocolos de seguridad (Houssam, Javier)
Programación de la aplicación (Cristian Echauri, Mateo, Javier Palacios)

3. Registro diario de tareas

05/01/2022
Houssam: creación base de datos phpmyadmin(BASE DE DATOS EN PHPMYADMIN)
Mateo: desarrollo documento(Introducción y Diseño WIP)
Cristian: Investigación comunicación socket transmisión de datos
Mildred: Investigación funcionamiento aplicación de chat general
Javier: investigación sobre programación multihilo

14/01/2022
Houssam: perfección y relación de base de datos.
Mateo: código entrelazamiento de base de datos y programa para el login.
Mildred: código y diseño de la interfaz de login.
Javier: escanea la red para ver quién está conectado (quién ha abierto la aplicación) así se enlazan los usuarios de forma automática
Cristian: desapestación de covid, recupera adecuadamente.

19/01/2022
Houssam: generación de base de datos.
Mateo: programación relacion base de datos y java.
Mildred: diseño y presentación de la aplicación.
Javier: programación chat.
Cristian: en proceso de desapestación, recupera aún más adecuadamente.

26/01/2022
Houssam: relaciones PK y FK de base de datos.
Mateo: programación login base de datos.
Mildred: diseño y presentación de la aplicación.
Javier: desarrollo programación comunicación del chat
Cristian: Trabajo empresa

02/02/2022
Houssam: diseño logo, propiedades de copypaste en contraseña
Mateo: desarrollo registro de usuarios mediante base de datos
Mildred: no presente
Javier: desarollo funcionalidad de chat.
Cristian: soporte en el desarrollo del chat.

04/02/2022
Houssam: programación encriptación contraseñas
Mateo: vacaciones
Mildred: inserción de imágenes en base de datos.
Javier: programación conversaciones grupales
Cristian: investigación de protocolos sobre el método de comunicación entre sockets.

16/02/2022
Houssam: Desencriptación de contraseñas y gestión de datos
Mateo: soporte soluciones errores y documentación.
Mildred: no presente
Javier: programación creación de grupales y envío de mensajes
Cristian: Programación y ayuda a solución de errores

25/02/2022
Houssam: desencriptación de contraseñas y gestión de datos
Mateo: actualización de información decrépita del documento
Mildred: apoyo moral y emocional mediante su gran alegría y sonrisa, también diseño.
Javier: solución de bugs y programación chat..
Cristian: desarrollo cámara web.

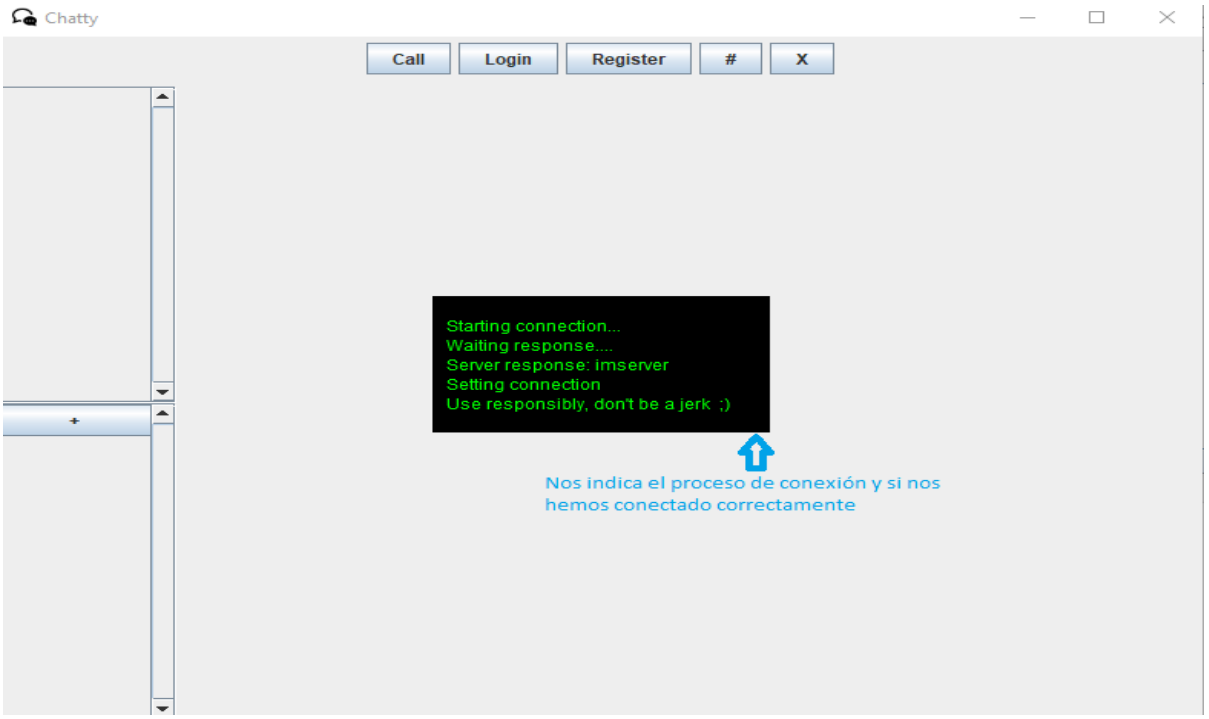
02/03/2022
Houssam: documentación librería de imports.
Mateo: documentación librería de imports.
Mildred: documentación librería de imports.
Javier: programación chat.
Cristian: envío de información entre clientes de la cámara web

04/03/2022
Houssam: programación encriptación de contraseñas.
Mateo: documentación funcionamiento programa.
Mildred: documentación de los métodos del programa.
Javier: ayuda en el funcionamiento y entendimiento del programa.
Cristian: envió de información entre clientes de cámara web

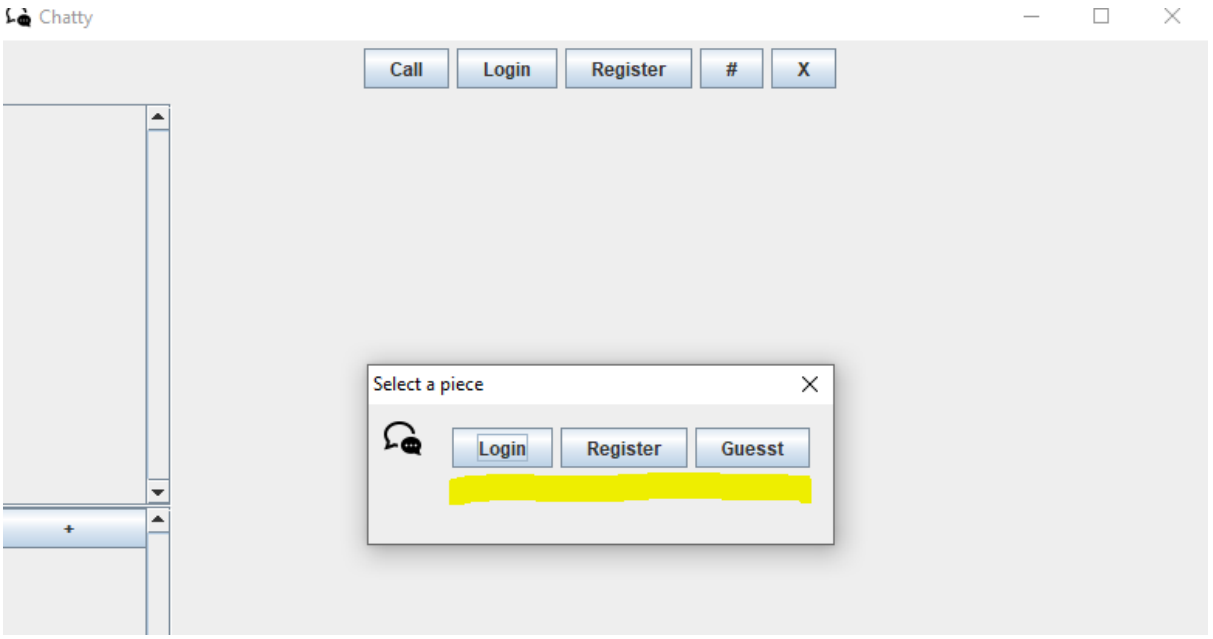
4. Diseño Aplicación

Antes de ejecutar la aplicación del chat, cualquier miembro del chat tiene que ejecutar el servidor.

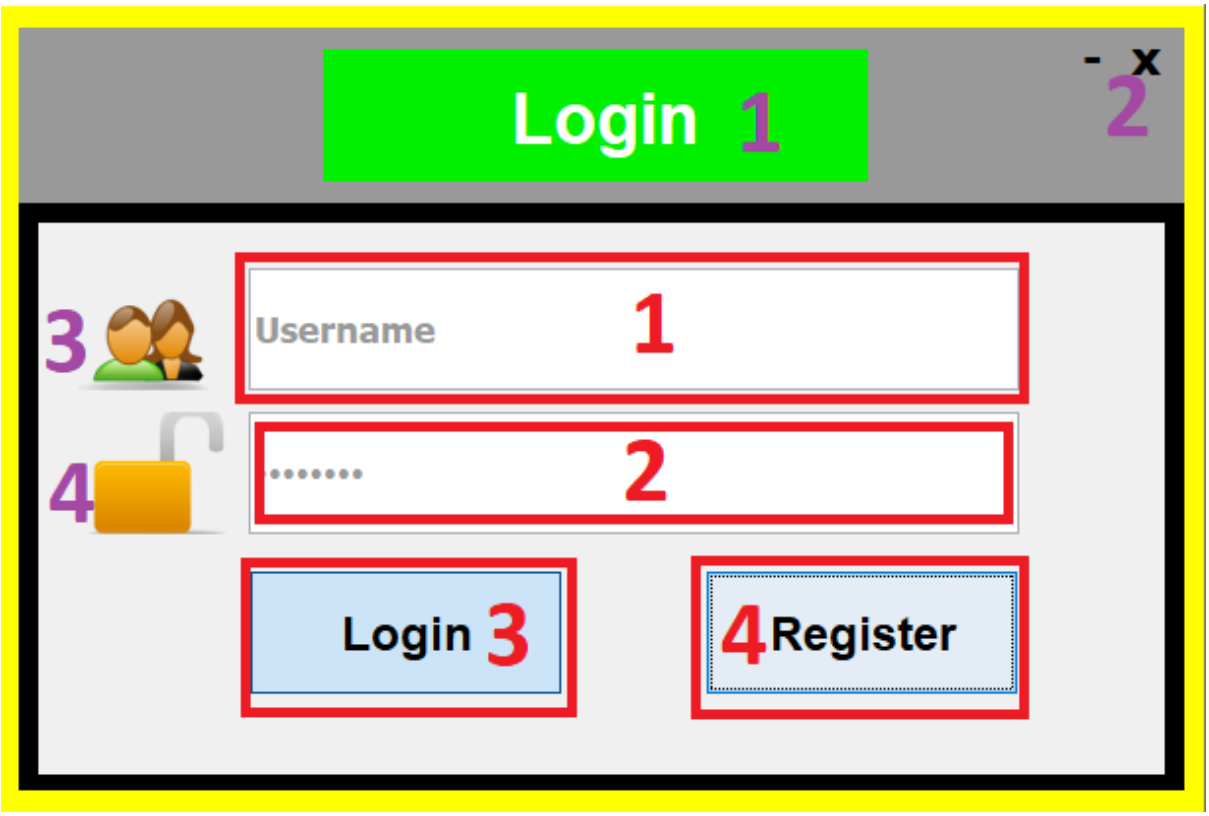
Inicio Conexión Chat



Primero hemos de saber si la conexión ha sido exitosa, si es así, nos indica que tenemos que hacer a través de los botones de login, registro o invitado:



Pestaña de Login



User input

1-Escribir usuario
2-Escribir contraseña
3-Hacer login al chat
4-Abrir registro de chat

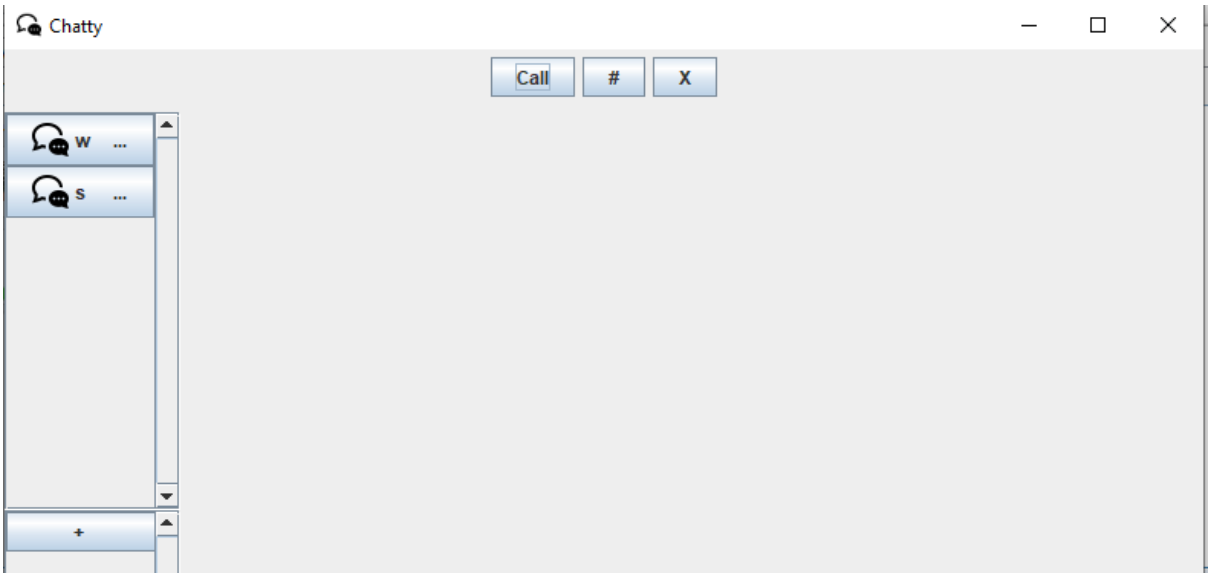
Diseño

1- Visual de login
2- Minimizar y Cerrar
3- Icono usuario
4- Icono contraseña

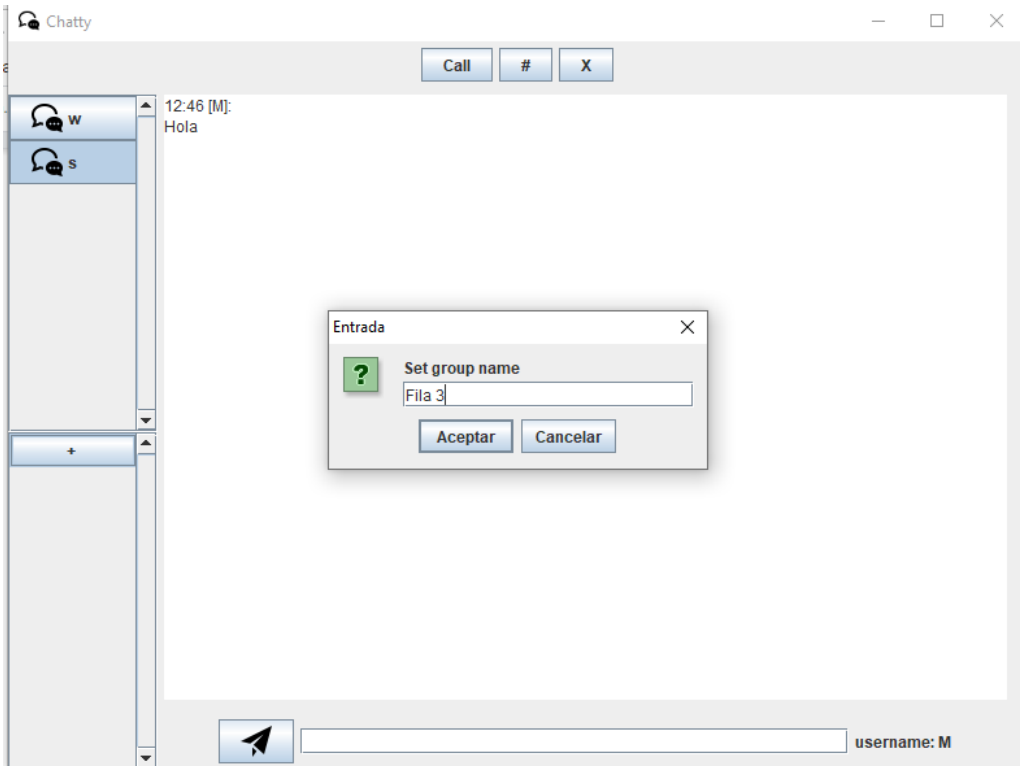
Si ya tienes un usuario y contraseña registrados en la base de datos, puedes logearte dándole al botón de login. Te salen diferentes mensajes de advertencia si haces algo erróneo: si dejas alguno de los campos en blanco o si te equivocas introduciendo la contraseña. Si no estás registrado, puedes registrarte dándole al botón de registro.

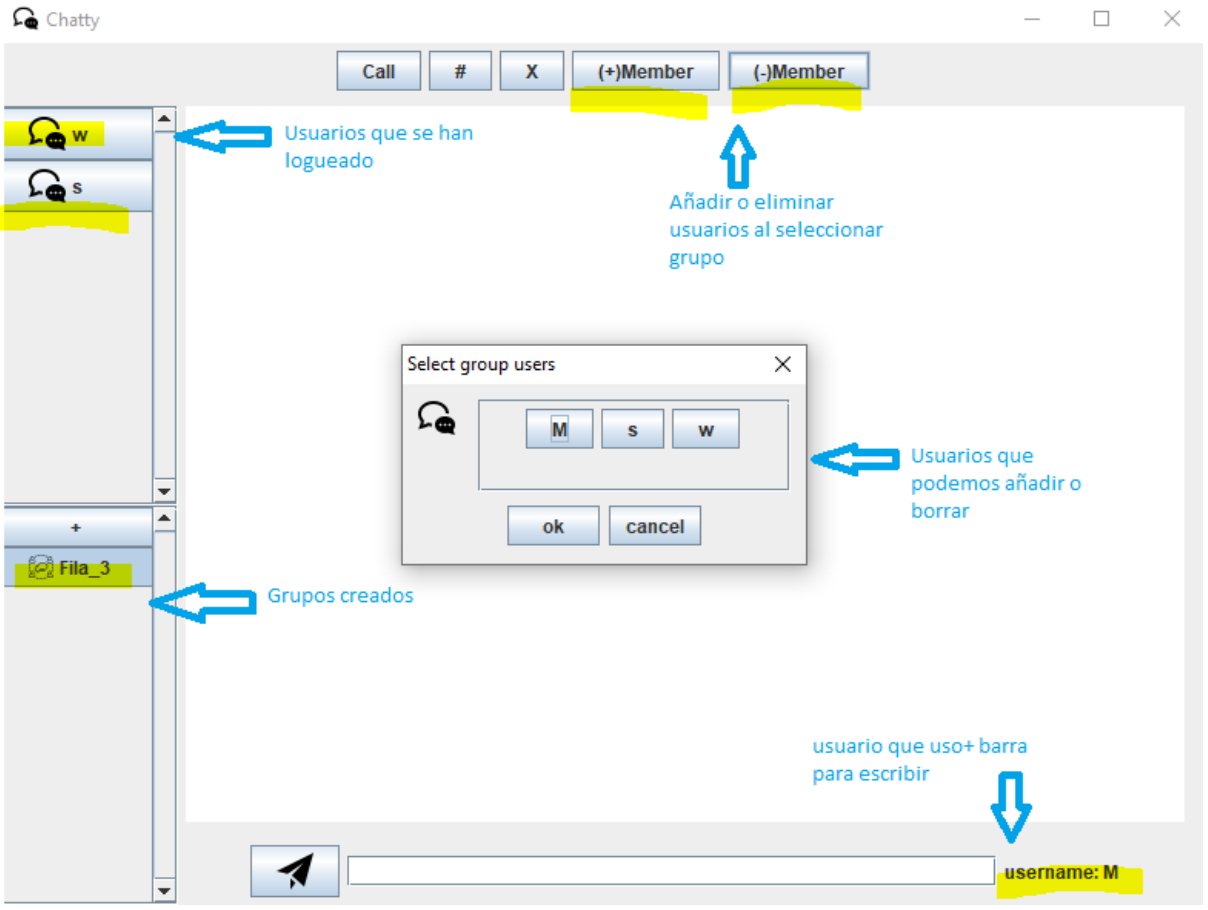
Formulario de registro

Puedes añadirle una foto al usuario al registrarse. Una vez que ya te has registrado, le das al botón de ">>¿Tienes cuenta? Inicia Sesión!" e irás a la pantalla de login.
Una vez te logueas, te salen los usuarios que están conectados (tú entre ellos).



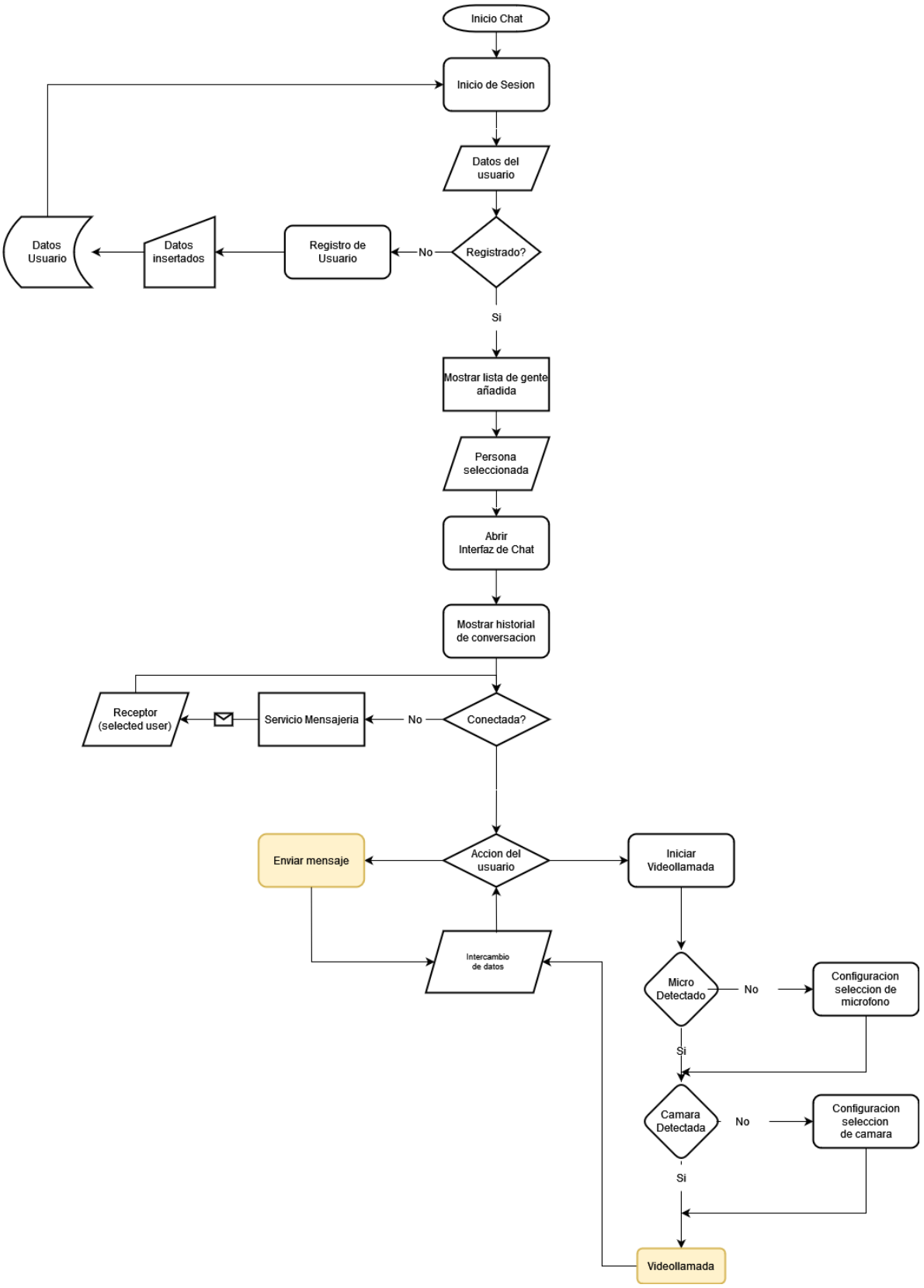
Puedes seleccionar al usuario con quien deseas hablar o crear un grupo y añadir en él los usuarios que te interesen. Una vez creado, al seleccionar el grupo, puedes añadir más miembros o eliminarlos.





5. Base de datos

5.1 Diagrama de flujo del programa



5.2 Instalación de la base de datos

La base de datos está protegida por contener información sensible y no es accesible ni descargable por medios legales.

Pero… Si quieres saber como importar una base de datos…

Casi toda importación de SQL consta de dos archivos: un archivo para la estructura de la base de datos y un archivo para los datos como tal.

Nos aseguramos de que la BBDD a importar sea compatible con nuestra versión de MySQL.

Iniciamos el servicio de MySQL e iniciamos sesión:

```
$> mysql -u root
```

Cargamos los archivos en nuestro gestor con la palabra clave SOURCE:

```
mysql> SOURCE C:/ruta-al-archivo/esquema.sql;
mysql> SOURCE C:/ruta-al-archivo/datos.sql;
```

Cambiamos a la base de datos cargada y hacemos un pequeña consulta para comprobar que todo ha funcionado correctamente,

```
mysql> USE nombre_db;
mysql> SHOW FULL TABLES;
```

5.3 Estructura de la base de datos

- Usuarios

La tabla *usuarios* es la lista de la información de todos los usuarios registrados en la base de datos.

Está unida con Chat y Messages mediante Foreign Keys.

Columnas

- **(PK) usuario_id**: identificación interna única del usuario
- **nombre_usuario**: nombre que el usuario utilizará para iniciar sesión
- **nombre_visible**: nombre que los amigos del usuario verán en el chat
- **nombre_legal**: nombre/s legal/es del usuario
- **apellido_legal**: apellido/s legal/es del usuario
- **ultima_ip**: última IP utilizada por el usuario

- Chat_ID

La categoría *chat_id* usa una Primary Key para la relación usuarios - salas de chat, al conocer los participantes muestra el chat correspondiente.

Está unida a Participantes y Mensajes mediante Foreign Keys.

Columnas

- **(PK) chat_id**: identificación interna única de la sala de chat.
- **chat_nombre**: nombre de la sala de chat.

- Mensajes

La tabla *mensajes* guarda todos los mensajes enviados por los usuarios y a que sala de chat va dirigido.

Está relacionada con las tablas usuarios y chat.

Columnas

- **(PK) mensaje_id**: identificación interna única del mensaje
- **chat_id**: identificación única interna de la sala de chat receptora del mensaje
- **id_usuario**: identificación interna única del usuario emisor del mensaje
- **mensaje_texto**: texto del mensaje enviado
- **mensaje_ts**: fecha y hora del mensaje

Participantes

La tabla *participantes* contiene la información necesaria para determinar en qué salas de chat están unidos los usuarios.

Columnas

- **chat_id**: identificación interna única de la sala del chat
- **usuario_id**: identificación interna única del usuario

Usuarios_ip

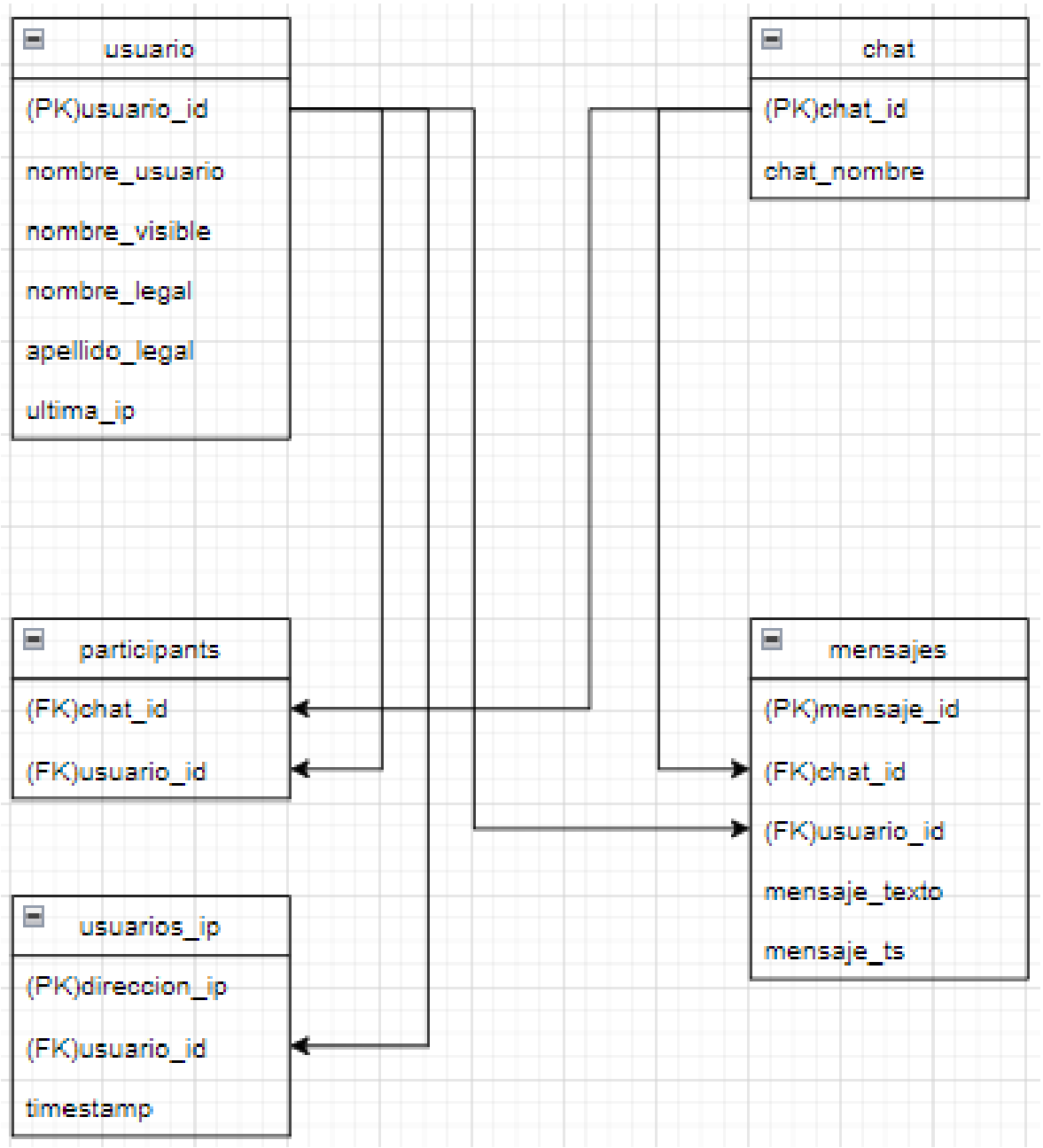
La vista *usuarios_ip* contiene un histórico de las IPs desde las que se han conectado los usuarios y la fecha/hora.

Está relacionada con la tabla usuarios.

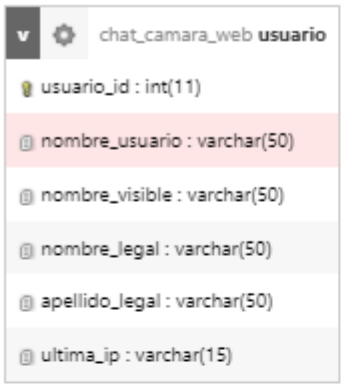
Columnas

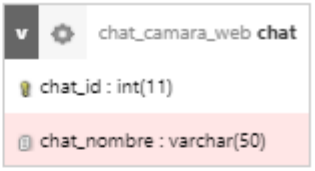
- **(PK) direccion_ip** : dirección IP desde la que se ha conectado el usuario
- **usuario_id** : identificación interna única del usuario
- **timestamp**: fecha y hora de conexión

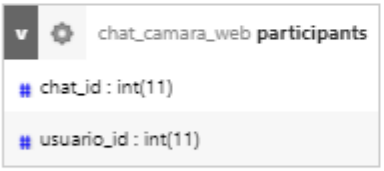
5.4 Diagrama relacional

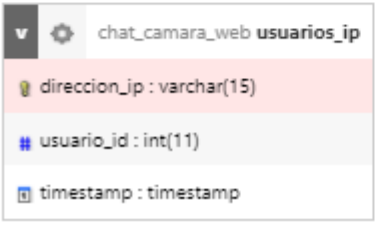


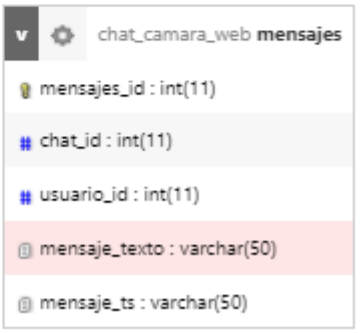
5.5 Diagrama en PHPMYADMIN











6. Librerías e Imports

- **java.awt.Color;**

Implementa qué colores pueden ser utilizados en el programa.

- **java.awt.event.*;**

java.awt.event.ActionListener; Otorga a un objeto del programa la habilidad de reaccionar cuando ocurre un evento.
java.awt.event.ActionEvent; Relacionado con el ActionListener, al detectar que el usuario ha reaccionado al evento indicado, actuar en respuesta a ello

```
gbtn.addActionListener((ActionEvent e) -> {
    //unselectButtons(e);
    addGroupButtons();
    chatxt.setText(chatstorage.get(chatid));
    chatID = chatid;
    chat.setVisible(true);
    adress = "";
 });
```

java.awt.event.KeyAdapter; Otorga a un elemento detectar los eventos creados por por las acciones a elementos swing para acciones del usuario, como cuando es pulsado.
java.awt.event.KeyEvent; Relacionado con el KeyAdapter, al detectar el evento designado, activar y ejecutar la acción.

```
userinput.addKeyListener(new KeyAdapter(){
    public void keyPressed(KeyEvent pressed){sendToChat(pressed);}
});
```

java.awt.event.WindowEvent; indica cuando ha sido interactuada la ventana.

- **java.io**

Son los paquetes que se encargan de la entrada y salida de flujos de información entre el programa y el sistema así como el acceso a archivos externos.

java.io.BufferedReader; Lee un flujo de caracteres y utiliza un búfer para que el programa pueda leerlo al completo y eficientemente.

```
try {
        URL tempURL = new URL("http://checkip.amazonaws.com/");
        HttpURLConnection tempConn = (HttpURLConnection)tempURL.openConnection();
        InputStream tempInStream = tempConn.getInputStream();
        InputStreamReader tempIsr = new InputStreamReader(tempInStream);
        BufferedReader tempBr = new BufferedReader(tempIsr);

        publicIP = tempBr.readLine();
```

java.io.EOFException; Excepción para el try-catch de cuando ha llegado al final del archivo o flujo de datos inesperadamente.

```
catch(EOFException ex){System.out.println("Wrong chat connection protocol");}
```

java.io.File; Permite acceder a archivos externos al programa mediante el sistema operativo para obtener sus directorios.

```
JFileChooser chooser = new JFileChooser();
chooser.setCurrentDirectory(new File(System.getProperty("user.home")));

File selectedImage = chooser.getSelectedFile();
path = selectedImage.getAbsolutePath();
jLabel_imgpath.setText(path);
```

java.io.IOException; Utilizado para generar un error generalizado de cuando ocurre un error dentro del paquete .io.

```
private void sayHelloToChat(Socket request, Package p) throws IOException{
```

java.io.InputStream; Clase padre de los InputStream que se ocupa de la entrada de flujo de bytes al programa.

java.io.InputStreamReader; Lee los bytes del InputStream y los convierte en caracteres.

```
String getip = locateip.getHostAddress();
txt.append("New connection: "+getip);
p.setStatus("imserver");
Socket sendmsg = new Socket(getip, 9090);
ObjectOutputStream msgpackage = new ObjectOutputStream(sendmsg.getOutputStream());
msgpackage.writeObject(p);

msgpackage.close(); sendmsg.close();
```

java.io.ObjectOutputStream; Deconstruye un objeto para poder mandar la información que luego se puede leer usando el ObjectInputStream.

```
String getip = locateip.getHostAddress();
txt.append("New connection: "+getip);
p.setStatus("imserver");
Socket sendmsg = new Socket(getip, 9090);
ObjectOutputStream msgpackage = new ObjectOutputStream(sendmsg.getOutputStream());
msgpackage.writeObject(p);

msgpackage.close(); sendmsg.close();
```

java.io.ObjectInputStream; Recoge la información del ObjectOutputStream para reconstruir el objeto.

```
while(true){
    try (Socket request = port.accept()) {

        ObjectInputStream entrada = new ObjectInputStream(request.getInputStream());
        p = (Package) entrada.readObject();
```

java.io.Serializable; Interfaz que se implementa a las clases para poder ser serializadas (por ObjectOutputStream y ObjectInputStream).

```
public class Package implements Serializable{

    private String nick, ip, info, msg, status;
    private Map<String, String[]> ips;

    public void setNick(String nick){this.nick = nick;}
    public void setIp(String ip){this.ip = ip;}
    public void setInfo(String info){this.info = info;}
    public void setMsg(String msg){this.msg = msg;}
    public void setStatus(String st){this.status = st;}
    public void setObj(Map<String, String[]> ips){this.ips = ips;}
    public String getNick(){return nick;}
    public String getIp(){return ip;}
    public String getInfo(){return info;}
    public String getMsg(){return msg;}
    public String getStatus(){return status;}
    public Map<String,String[]> getObj(){return ips;}
}
```

- **java.Net**

Comprende los paquetes que se ocupan de las conexiones de red

java.net.ConnectException; Notifica cuando hay un error conectándose a un socket remoto.

```
try {
        URL tempURL = new URL("http://checkip.amazonaws.com/");
        HttpURLConnection tempConn = (HttpURLConnection)tempURL.openConnection();
        InputStream tempInStream = tempConn.getInputStream();
        InputStreamReader tempIsr = new InputStreamReader(tempInStream);
        BufferedReader tempBr = new BufferedReader(tempIsr);

        publicIP = tempBr.readLine();
```

java.net.HttpURLConnection; Se utiliza para establecer conexión con una URL.

```
try {
        URL tempURL = new URL("http://checkip.amazonaws.com/");
        HttpURLConnection tempConn = (HttpURLConnection)tempURL.openConnection();
        InputStream tempInStream = tempConn.getInputStream();
        InputStreamReader tempIsr = new InputStreamReader(tempInStream);
        BufferedReader tempBr = new BufferedReader(tempIsr);

        publicIP = tempBr.readLine();
```

java.net.InetAddress; Representa la dirección IP (protocolo de internet).

```
  private void setServerIP(Socket mysocket, Package p) {
      userInfo.append("   Setting connection \n");

      InetAddress locateip = mysocket.getInetAddress();
      serverIP = locateip.getHostAddress();
```

java.net.NetworkInterface; Interfaz creada por el usuario que contiene lista de IPs

```
public static ArrayList getLocalIp(){

    ArrayList<String> localIP = new ArrayList<>();
    try {
        for (NetworkInterface iface : Collections.list(NetworkInterface.getNetworkInterfaces())) {
            // Due to the amount of the interfaces, we will only print the ones online
            if (iface.isUp()) {localIP.add(iface.getInetAddresses().nextElement().getHostAddress());}
        }
    } catch (SocketException ex) {
        Logger.getLogger(GetIP.class.getName()).log(Level.SEVERE, null, ex);
    }
    return localIP;
}
```

java.net.ServerSocket; Permite crear sockets de servidores a la espera de peticiones, la ejecuta en algunos casos devuelve una respuesta.

```
        public void run() {
            try {
                ServerSocket port = new ServerSocket(9090);
                String nick, ip, move, msg;
                Package p;

                while(true){
                    try (Socket mysocket = port.accept()) {
                        ObjectInputStream entrada = new ObjectInputStream(mysocket.getInputStream());
                        p = (Package) entrada.readObject();

                        userInfo.append("   Server response: "+ p.getStatus()+"\n");
```

java.net.Socket; Permite crear sockets de cliente que es el punto de comunicación entre 2 máquinas.

```
  while(true){
      try (Socket mysocket = port.accept()) {
          ObjectInputStream entrada = new ObjectInputStream(mysocket.getInputStream());
          p = (Package) entrada.readObject();

          userInfo.append("   Server response: "+ p.getStatus()+"\n");
```

```
try {
    try (Socket socket = new Socket(server,9999)) {
        packager.Package p = new packager.Package();
        p.setNick(nick);
        p.setIp(adress);
        p.setStatus(status);
        p.setMsg(msg);
        p.setInfo(chatid);

        ObjectOutputStream objp = new ObjectOutputStream(socket.getOutputStream());
        objp.writeObject(p);
        socket.close();
    }

} catch (IOException ex) {ex.printStackTrace();}
```

java.net.SocketException; Notifica cuando hay un error entra la comunicación de sockets.

```
public static ArrayList getLocalIp(){

    ArrayList<String> localIP = new ArrayList<>();
    try {
        for (NetworkInterface iface : Collections.list(NetworkInterface.getNetworkInterfaces())) {
            // Due to the amount of the interfaces, we will only print the ones online
            if (iface.isUp()) {localIP.add(iface.getInetAddresses().nextElement().getHostAddress());}
        }
    } catch (SocketException ex) {
        Logger.getLogger(GetIP.class.getName()).log(Level.SEVERE, null, ex);
    }
    return localIP;
}
```

java.net.URL; Permite especificar URLs (Uniform Resource Locator) que conectan a las páginas WEB.

```
URL tempURL = new URL("http://checkip.amazonaws.com/");
HttpURLConnection tempConn = (HttpURLConnection)tempURL.openConnection();
```

```
try {
        URL tempURL = new URL("http://checkip.amazonaws.com/");
        HttpURLConnection tempConn = (HttpURLConnection)tempURL.openConnection();
        InputStream tempInStream = tempConn.getInputStream();
        InputStreamReader tempIsr = new InputStreamReader(tempInStream);
        BufferedReader tempBr = new BufferedReader(tempIsr);

        publicIP = tempBr.readLine();

        tempBr.close();
        tempInStream.close();
```

HOUSSAM

- **java.sql.Connection;**

Una conexión (sesión) con una base de datos específica. Las sentencias SQL se ejecutan y los resultados se devuelven dentro del contexto de una conexión.

La base de datos de un Connection Objeto puede proporcionar información que describe sus tablas, su gramática SQL admitida, sus procedimientos almacenados, las capacidades de esta conexión, etc. Esta información se obtiene con el método getMetaData.

```
public class Login extends javax.swing.JFrame {
    DefaultTableModel model = null;
    String driver = "com.mysql.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/ccw?autoReconnect=true&useSSL=false";
    String user = "root";
    String pass = "";
    Connection conn = null;
    PreparedStatement ps = null;
    Statement st = null;
    ResultSet rs = null;
    int islogin;
    String loginuser;
    String password;
    String registerlogin;
```

java.sql.DriverManager;

El servicio básico para administrar un conjunto de controladores JDBC.

es una clase estática de Java™ 2 Plaform, Standard Edition (J2SE) y Java SE Development Kit (JDK). DriverManager gestiona el conjunto de controladores Java Database Connectivity (JDBC) que están disponibles para que los utilice una aplicación

```
public void verifyLogin(){
        loginuser = txtUser.getText().toLowerCase();
        password = String.valueOf(txtPassword.getPassword());
        try {
            Class.forName(driver);
            try {
                conn = DriverManager.getConnection(url, user, pass);
                st = conn.createStatement();
                rs = st.executeQuery("SELECT * FROM usuario WHERE nombre_usuario='" + loginuser + "'"
                        + "and contrasenya='" + password + "'");
                if (rs.next()) {
                    JOptionPane.showMessageDialog(null, "Acceso autorizado");
                } else {
                    JOptionPane.showMessageDialog(null, "Por favor comprueba las credenciales");
                }
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(null, "No vaAAAAAAAAA2");
            }
        } catch (ClassNotFoundException e) {
            JOptionPane.showMessageDialog(null, "No va");
        }
```

java.sql.PreparedStatement;

Un objeto que representa una instrucción SQL precompilada.

Una instrucción SQL se compila previamente y se almacena en un objeto PreparedStatement. Este objeto se puede usar para ejecutar de manera eficiente esta declaración varias veces.

en la clase register podremos encontrarlo

```
public class Register extends javax.swing.JFrame {

    DefaultTableModel model = null;
    String driver = "com.mysql.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/ccw?autoReconnect=true&useSSL=false";
    String user = "root";
    String pass = "";
    Connection conn = null;
    PreparedStatement ps = null;
    Statement st = null;
    ResultSet rs = null;
    int islogin;
```

```
            txtGenero = "Masculino";
        }
        ps = conn.prepareStatement("INSERT INTO `usuario`( `nombre_usuario`, `nombre_v
        ps.setString(1, txtUser);
        ps.setString(2, txtNickname);
        ps.setString(3, txtName);
        ps.setString(4, txtSurname);
        ps.setString(5, txtPassword);
        ps.setString(6, txtGenero);
        ps.executeUpdate();
verManager.getConnection(url, user, pass);
```

java.sql.ResultSet;

Proporciona varios métodos para obtener los datos de columna correspondientes a un fila

```
public class Register extends javax.swing.JFrame {

    DefaultTableModel model = null;
    String driver = "com.mysql.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/ccw?autoReconnect=true&useSSL=false";
    String user = "root";
    String pass = "";
    Connection conn = null;
    PreparedStatement ps = null;
    Statement st = null;
    ResultSet rs = null;
    int islogin;
```

```
593     private void jButton_RegistroActionPerformed(java.awt.event.ActionEvent evt) {
594         // TODO add your handling code here:
595         // TODO add your handling code here:
596         txtUser = userUser.getText().toLowerCase();
597         try {
598             Class.forName(driver);
599             try {
600                 conn = DriverManager.getConnection(url, user, pass);
601                 st = conn.createStatement();
602                 rs = st.executeQuery("SELECT * FROM usuario WHERE nombre_usuario='" + txtUser + "'");
603                 if (!rs.next()) {
604                     txtPassword = String.valueOf(userPassword.getPassword());
605                     System.out.println(txtPassword);
606                     txtPassword2 = String.valueOf(userPassword2.getPassword());
607                     System.out.println(txtPassword2);
608                     if (txtPassword.equals(txtPassword2)) {
```

java.sql.SQLException;

Es la excepción que se lanza cuando hay algún problema entre la base de datos y el programa Java JDBC. Por ejemplo, cuando realizamos una consulta debemos de poner nuestro código que maneje el ResultSet dentro de de un bucle try-catch.

```
649                 } catch (SQLException ex) {
650                     JOptionPane.showMessageDialog(null, "No se pudo conectar con el servidor");
651                 }
652             } catch (ClassNotFoundException e) {
653                 JOptionPane.showMessageDialog(null, "");
654             }
```

java.sql.Statement;

Proporciona métodos para ejecutar consultas con la base de datos . La interfaz de declaración es una fábrica de ResultSet, es decir, proporciona un método de fábrica para obtener el objeto de ResultSet.

```
public class Register extends javax.swing.JFrame {

    DefaultTableModel model = null;
    String driver = "com.mysql.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/ccw?autoReconnect=true&useSSL=false";
    String user = "root";
    String pass = "";
    Connection conn = null;
    PreparedStatement ps = null;
    Statement st = null;
    ResultSet rs = null;
    int islogin;
```

```
593     private void jButton_RegistroActionPerformed(java.awt.event.ActionEvent evt) {
594         // TODO add your handling code here:
595         // TODO add your handling code here:
596         txtUser = userUser.getText().toLowerCase();
597         try {
598             Class.forName(driver);
599             try {
600                 conn = DriverManager.getConnection(url, user, pass);
601                 st = conn.createStatement();
602                 rs = st.executeQuery("SELECT * FROM usuario WHERE nombre_usuario='" + txtUser + "'")
603                 if (!rs.next()) {
604                     txtPassword = String.valueOf(userPassword.getPassword());
605                     System.out.println(txtPassword);
606                     txtPassword2 = String.valueOf(userPassword2.getPassword());
607                     System.out.println(txtPassword2);
608                     if (txtPassword.equals(txtPassword2)) {
```

- **java.text.DateFormat;**

Es una clase abstracta de la subclase de formato de fecha / hora. Podemos usar esta clase para ayudarnos a completar la conversión entre fecha y texto, es decir, convertir de un lado a otro entre objetos Date y objetos String

- **java.text.SimpleDateFormat;**

Nos ayuda a mostrar las fechas en el formato que queramos o a reconstruirlas a partir de una cadena de texto

MILDRED

- **java.util.**

Contiene el marco de colecciones, las clases de colección heredadas, el modelo de eventos, las instalaciones de fecha y hora, la internacionalización y las clases de utilidad diversas (un tokenizador de cadenas, un generador de números aleatorios y una matriz de bits).

java.util.ArrayList;

La clase ArrayList en Java, es una clase que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el número de elementos que almacena, lo hace de forma dinámica, es decir, que no es necesario declarar su tamaño como pasa con los Arrays. Los ArrayList nos permiten añadir, eliminar y modificar elementos.

```
public static ArrayList getLocalIp(){

    ArrayList<String> localIP = new ArrayList<>();
    try {
        for (NetworkInterface iface : Collections.list(NetworkInterface.getNetworkInterfaces())) {
            // Due to the amount of the interfaces, we will only print the ones online
            if (iface.isUp()) {localIP.add(iface.getInetAddresses().nextElement().getHostAddress());}
        }
    } catch (SocketException ex) {
        Logger.getLogger(GetIP.class.getName()).log(Level.SEVERE, null, ex);
    }
    return localIP;
```

java.util.Collections;

Una colección representa un grupo de objetos. Estos objetos son conocidos como elementos. Cuando queremos trabajar con un conjunto de elementos, necesitamos un almacén donde poder guardarlos. En Java, se emplea la interfaz genérica **Collection** para este propósito. Gracias a esta interfaz, podemos almacenar cualquier tipo de objeto y podemos usar una serie de métodos comunes, como pueden ser: añadir, eliminar, obtener el tamaño de la colección… Partiendo de la interfaz genérica **Collection** extienden otra serie de interfaces genéricas. Estas subinterfaces aportan distintas funcionalidades sobre la interfaz anterior.

La **colección en Java** es un marco que proporciona una arquitectura para almacenar y manipular el grupo de objetos. Java Collections puede lograr todas las operaciones que realiza en un dato, como la búsqueda, la ordenación, la inserción, la manipulación y la eliminación. Java Collection significa una sola unidad de objetos. Java Collection framework proporciona muchas interfaces (Set, List, Queue, Deque) y clases (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

```java
public static ArrayList getLocalIp(){

    ArrayList<String> localIP = new ArrayList<>();
    try {
        for (NetworkInterface iface : Collections.list(NetworkInterface.getNetworkInterfaces())) {
            // Due to the amount of the interfaces, we will only print the ones online
            if (iface.isUp()) {localIP.add(iface.getInetAddresses().nextElement().getHostAddress());}
        }
    } catch (SocketException ex) {
        Logger.getLogger(GetIP.class.getName()).log(Level.SEVERE, null, ex);
    }
    return localIP;
```

java.util.Date;

La clase java.util.Date representa la fecha y la hora en java. Proporciona constructores y métodos para tratar la fecha y la hora en java. La clase java.util.Date implementa la interfaz Serializable, Cloneable y Comparable<Date>. Es heredado por las interfaces java.sql.Date, java.sql.Time y java.sql.Timestamp.

```java
    public void actionPerformed(ActionEvent event) {
        if(event.getSource() == sendbtn){
            try {
                String msg = timeFormat.format(new Date())+"["+nick+"]:\n"+userinput.getText() + "\n";
                Send.message( adress, msg, nick, "messaging", chatID);
```

```java
public void sendToChat(KeyEvent pressed) {
    String msg = timeFormat.format(new Date())+"["+nick+"]:\n"+userinput.getText() + "\n";
```

java.util.HashMap;

java.util.Map;

java.util.logging.Level: la clase Level define un conjunto de niveles de registro estándar que se pueden usar para controlar la salida del registro. Los objetos de nivel de registro están ordenados y especificados por enteros ordenados. Habilitar el registro en un nivel determinado también habilita el registro en todos los niveles superiores.

java.util.logging.Logger: proporciona una variedad de métodos con los que se pueden registrar los datos.

- javax.swing.

Swing es una biblioteca gráfica para Java. Incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, listas desplegables y tablas.

Una GUI de Swing consta de dos elementos clave: **componentes y contenedores**. Sin embargo, esta distinción es principalmente conceptual porque todos los contenedores

también son componentes. La diferencia entre los dos se encuentra en su propósito previsto: como se usa comúnmente el término, un componente es un control visual independiente, como un pulsador o campo de texto. Un contenedor contiene un grupo de componentes. Así, un contenedor es un tipo especial de

componente que está diseñado para contener otros componentes. Además, para que un componente

se muestre, debe mantenerse dentro de un contenedor. Por lo tanto, todas las GUI de Swing tendrán al menos un contenedor.

Dado que los contenedores son componentes, un contenedor también puede contener otros contenedores. Esto permite a Swing defina lo que se llama una jerarquía de contención, en la parte superior de la cual debe haber un contenedor de nivel superior.

javax.swing.BorderFactory;

Esta clase tiene métodos estáticos que nos ayudan a crear los bordes de una forma sencilla.

Lo usamos para crear los bordes del panel, las casillas de minimizar y cerrar, etc. Esto en el login.

```
        this.setLocationRelativeTo(null);
//        creamos un borde amarillo para el titulo del panel
//        los cuatro números son los bordes de la casilla, cuanto mas altos, mayor grosor de la linea
//         con el 0, en la parte superior del titulo del panel no ponemos borde
        Border jpanel_titulo_borde = BorderFactory.createMatteBorder(0, 1, 1, 1, Color.YELLOW);
//        fijamos el borde al jpanel del titulo
      jPanel_titulo.setBorder(jpanel_titulo_borde);
```

javax.swing.ButtonGroup;

Se utiliza para presentar al usuario un conjunto de opciones excluyentes entre sí, es decir si el usuario selecciona un componente RadioButton todos los demás componentes RadioButton en la forma, se desmarcan solos.

Lo usamos en el formulario de registro para seleccionar el género.

```
//creamos un boton de grupo para la selección del género
ButtonGroup bg = new ButtonGroup();
bg.add(jRadioButton_Masculino);
bg.add(jRadioButton_Femenino);
bg.add(jRadioButton_Otros);
```

javax.swing.ImageIcon;

Una implementación de la interfaz Icon que pinta iconos a partir de imágenes. Las imágenes que se crean a partir de una URL, un nombre de archivo o una matriz de

bytes se precargan mediante MediaTracker para controlar el estado de carga de la imagen.

Lo usamos para añadir una imagen/icono en los jlabels de usuario y contraseña del login.

```
jLabel_user.setIcon(new ImageIcon("img/password.png"));

jLabel_password.setIcon(new ImageIcon("img/user.png"));
```

javax.swing.JFileChooser;

El objeto de la clase JFileChooser representa una ventana de diálogo desde la que el usuario puede seleccionar el archivo. Hereda la clase JComponent.

Lo usamos en el formulario de registro que está conectado a la base de datos donde seleccionamos una imagen y la cargamos al usuario que se registra en la base de datos.

```
//Seleccionar una imagen y fijarla en la jlabel del image path
String path = null;
JFileChooser chooser = new JFileChooser();
chooser.setCurrentDirectory(new File(System.getProperty("user.home")));

//extensión archivo
FileNameExtensionFilter extension = new FileNameExtensionFilter("*Images", "jpg", "png", "jpeg");
chooser.addChoosableFileFilter(extension);

int filestate = chooser.showSaveDialog(null);
//comprobar si el usuario selecciona una imagen
if (filestate == JFileChooser.APPROVE_OPTION) {

    File selectedImage = chooser.getSelectedFile();
    path = selectedImage.getAbsolutePath();
    jLabel_imgpath.setText(path);

    image_path = path;
```

javax.swing.JFrame;

La clase javax.swing.JFrame es un tipo de contenedor que hereda la clase java.awt.Frame. JFrame funciona como la ventana principal donde se agregan componentes como etiquetas, botones, campos de texto para crear una GUI. A diferencia de Frame, JFrame tiene la opción de ocultar o cerrar la ventana con la ayuda del método setDefaultCloseOperation(int).

```
public class Register extends javax.swing.JFrame {
```

javax.swing.JTextArea

El objeto de una clase JTextArea es una región de varias líneas que muestra texto. Permite la edición de texto de múltiples líneas. Hereda la clase JTextComponent. JTextArea() Crea un área de texto que no muestra texto inicialmente.

Lo usamos en el chat

```
  private final JTextField userinput = new JTextField(38);
  private final JTextArea chatxt = new JTextArea(20,50);

  private JTextArea userInfo;
public Chat() {

//POP-UP textarea
    userInfo = new JTextArea(7,20);
    userInfo.setEditable(false);
```

javax.swing.Timer;

Permite la realización de tareas periódicas. Los objetos Timer pueden usarse de dos maneras:

- Ejecutar una tarea después de que haya transcurrido cierto tiempo
- Ejecutar una tarea repetidamente

La clase Timer tiene únicamente el siguiente constructor: **Timer**(int delay, ActionListener listener). El primer argumento es el tiempo entre cada tarea y el segundo un objeto que implemente la interfaz ActionListener cuyo método actionPerformed estará encargado de ejecutar las tareas.

Activa uno o más ActionEvents a intervalos específicos. Configurar un temporizador implica crear un objeto Timer, registrar uno o más detectores de acción en él e iniciar el temporizador usando el método de inicio.

```
            java.util.Timer timer = new java.util.Timer();
            timer.schedule(new KillSearchThread(t, timer), 100);
            t.start();
        }
        userInfo.append("   Waiting response....\n");
    }
    //Set message in case it takes too much
    new Timer(15000, (ActionEvent e) -> {
        if(!serverIP.equals("")){((Timer)e.getSource()).stop();}
        userInfo.append("   Maybe your Internet is down?\n");
        userInfo.append("   Or our server is fucked...\n");
    }).start();
`
   if(serverIP.equals(""))
       new Timer(20000, (ActionEvent e) -> {
           if(!serverIP.equals("")){((Timer)e.getSource()).stop();}
           else{getServerIP();}
       }).start();
```

javax.swing.border.Border;

Interfaz que describe un objeto capaz de representar un borde alrededor de los bordes de un componente oscilante.

javax.swing.filechooser.FileNameExtensionFilter;

Una implementación de FileFilter que filtra utilizando un conjunto especificado de extensiones. La extensión de un archivo es la parte del nombre del archivo después del último ".". Archivos cuyo nombre no contiene un "." no tienen extensión de nombre de archivo. Las comparaciones de extensión de nombre de archivo no distinguen entre mayúsculas y minúsculas.

javax.swing.table.DefaultTableModel;Esta es una implementación de TableModel que usa un Vector de vectores para almacenar los objetos de valor de celda.

```
public class Register extends javax.swing.JFrame {

    DefaultTableModel model = null;
    String driver = "com.mysql.jdbc.Driver";
```

7. Copias de Seguridad

La aplicación consta de tres partes que deberán ser salvaguardadas. Dos de ellas serán competencia del desarrollador, mientras que la restante será responsabilidad del administrador de sistemas del usuario final, sea este el mismo o una tercera entidad.

7.1 - La aplicación

Los datos que componen la aplicación y todos sus componentes externos necesarios (como librerías desarrolladas por terceros) están siempre guardados en formato físico por los desarrolladores, así como diversas versiones anteriores de la misma aplicación en caso de que hiciera falta un rollback.

Así mismo, se mantendrá constantemente actualizada una copia de los datos de la aplicación en la nube, actualmente en el servicio otorgado por GitHub. Esta copia es de acceso público y los usuarios finales deberán obtener su copia de la aplicación desde esta ubicación.

7.2 - La estructura de la BBDD

La estructura de la base de datos será competencia del desarrollador. Se utilizará el mismo procedimiento que con los datos de la aplicación, copia actualizada en físico con versiones anteriores como método de recuperación y copia en la nube.

El administrador de sistemas del usuario final podrá obtener la versión actualizada de la estructura directamente de GitHub. Esta estructura se proveerá en forma de archivo SQL con todo lo necesario para importarla.

7.3 - Datos generados por el uso de la aplicación

Los datos que se generen durante el uso de la aplicación y que se guardan en la base de datos, por ejemplo usuarios y mensajes, serán competencia del administrador de sistemas del usuario final, siendo este el mismo o una tercera entidad. Será discreción del administrador decidir con qué frecuencia guardará los datos y en qué formato. Por nuestra experiencia, recomendamos realizar una exportación acumulativa diaria en un formato plano no propietario para fomentar la mayor compatibilidad posible y la menor pérdida de información.